Approximate Solutions to Abstract Argumentation Problems Using Graph Neural Networks

Lars Malmqvist

Doctor of Philosophy

University of York
Computer Science

July 2022

Dedication

To Ada, Pino, and Damiana

Abstract

This thesis explores a new approach to approximating decision problems in abstract argumentation using Graph Convolutional Networks (GCN). It demonstrates that such an approach can reach well-balanced accuracy levels above 90 % across a range of different decision problems, argumentation semantics, and benchmarks.

This thesis develops a new Deep Neural Network (DNN) architecture adapted from the classic GCN that better addresses the specific issues found in abstract argumentation. Likewise, it develops a training approach that produces superior results for abstract argumentation data sets by introducing structured randomness and dynamic adaptation to the training data.

Then, the thesis systematically applies this architecture to a large argumentation dataset across the main argumentation semantics used in the biannual ICCMA competition. It evaluates the performance of the model in a variety of different settings and across benchmarks, size bands, and model variants. The main models show good performance in the majority of cases, although there is some variation.

Having created the core model, the thesis goes on to explore additional extensions of the core work. This first focuses on combining the approximate approach with exact approaches using a deterministic algorithm and a SAT solver, showing an improvement by solving six additional hard instances relative to existing solvers.

Second, we explore a visualisation approach that can give new insights into argumentation graphs by applying a dimensionality reduction technique to weights from the trained GCN models, showing new insights in explaining benchmark performance.

Finally, we explore using the same basic architecture to address another problem that can be structured using abstract argumentation. In this case, we apply the approach to the prediction of misinformation in tweets and achieve good performance on a key dataset.

Contents

Abstract **iii**

List of figures **ix**

List of tables **xiii**

Acknowledgements **xix**

Declaration **xxi**

Abbreviations **xxiii**

1 Introduction **1**

1.1 Background and rationale 1

1.2 Aim and objectives 3

1.3 Key contributions 4

1.4 Structure of the thesis 5

2 Background and Related Work **7**

2.1 Computational argumentation 7

2.2 Abstract argumentation 9

2.2.1 Definitions 10

2.2.2 Argumentation Semantics 12

2.2.2.1 Complete Semantics 12

2.2.2.2 Grounded Semantics 13

2.2.2.3 Preferred Semantics 13

2.2.2.4 Ideal Semantics 13

2.2.2.5 Stable Semantics 13

2.2.2.6 Semi-Stable and Stage Semantics 14
2.2.2.7 Summary of semantics 14
2.3 Deep learning 15
2.3.1 Basic concepts 15
2.3.1.1 Multi Layer Perceptron 15
2.3.1.2 Other basic Deep Neural Network (DNN) concepts 16
2.3.1.3 Different types of neural networks 17
2.3.1.4 Convolutional graph neural networks 21
2.4 Graph embeddings 24
2.4.1 Relevant work in other areas 25
2.4.1.1 Automated reasoning 25
2.4.1.2 Boolean Satisfiability 27
2.5 Solution approaches for problems in abstract argumentation 28
2.5.1 Exact approaches 28
2.5.1.1 SAT-Based Approaches 29
2.5.1.2 Answer-Set Programming 29
2.5.1.3 Labelling-based approaches 30
2.5.1.4 Dialogue Games 30
2.5.1.5 Dynamic Programming 31
2.5.2 Approximate approaches 31
2.5.2.1 Stochastic Search 31
2.5.2.2 Grounded Reasoning 31
2.5.2.3 Deep Learning 32
2.6 Summary 33

3 Approximating Acceptability in Abstract Argumentation Frameworks with Graph Convolutional Networks 35
3.1 AFGCN: an GCN-based approximate solver for abstract argumentation . . 37
3.1.1 Neural network architecture 37
3.1.1.1 Deep residual connections 38
3.1.2 Input features 39
3.1.2.1 Standard input features 39
3.1.2.2 Grounded reasoning as an input feature 40
3.1.3 Training regime 40

3.1.3.1 Randomised training batches 40
3.1.3.2 Dynamic balancing and outlier exclusion 41
3.1.4 Runtime implementation 42
3.2 Experimental results 43
3.2.1 SAFA 2020 results 43
3.2.1.1 Dataset and experimental setup 43
3.2.1.2 Credulous acceptability results 44
3.2.1.3 Sceptical acceptability results 44
3.2.1.4 Ablation studies 45
3.2.2 AFGCN results 47
3.2.2.1 Dataset and experimental setup 47
3.2.2.2 Results for credulous acceptance 51
3.2.2.3 Results for sceptical acceptance 61
3.2.2.4 Cross-cutting results 72
3.2.2.5 Runtime performance 76
3.3 Summary 79

4 Using Co-Admissibility to Predict Admissible Sets and as a SAT Heuristic 101

4.1 Predicting co-admissibility 102
4.1.1 Co-admissibility neural network training 103
4.1.2 Co-admissibility network architecture 104
4.1.3 Finding an admissible set from a set of predicted co-admissible arguments 105
4.2 Using co-admissibility as a SAT heuristic 107
4.3 Experimental results 107
4.3.1 Predicting co-admissibility 108
4.3.1.1 Experimental setup 108
4.3.1.2 Results 109
4.3.2 Co-admissibility as a SAT heuristic 109
4.3.2.1 Experimental setup 109
4.3.2.2 Results 110
4.4 Summary 111

5 Visualising Argumentation Graphs with Graph Embeddings and t-SNE 113

5.1 Visualising embeddings using t-SNE 114

5.2 Visualising argumentation graphs 116

5.2.1 Introduction 116

5.2.2 Node-level visualisation 117

5.2.3 Graph-level visualisation 118

5.3 Application to embeddings from AFGCN 120

5.4 Summary 123

6 Improving Misinformation Detection in Tweets with Abstract Argumentation 127

6.1 Misinformation detection on social networks 128

6.2 Method 129

6.2.1 Constructing the argumentation framework 129

6.2.2 Linguistic features 132

6.2.3 Argumentative features 132

6.2.4 GCN architecture 132

6.3 Experimental results 133

6.4 Summary 136

7 Conclusion 137

7.1 Conclusion 137

7.2 Key contributions 138

7.3 Future work 139

7.3.1 Further improvements to accuracy 139

7.3.2 Extension to other argumentation formalisms 140

7.3.3 Adaptation to additional problems 140

7.3.4 Deepen neuro-symbolic linkages 140

7.4 Broader knowledge contributions 141

7.4.1 Neuro-symbolic AI 141

7.4.2 Argumentation and machine learning 141

Appendix 143

A Additional Results Tables 143

References **153**

List of Figures

1.1 Example of a simple abstract argumentation framework. 2

2.1 Example of an Argumentation Framework. Nodes represent arguments in the argumentation framework. Edges represent attacks between arguments. [5]. 8

2.2 Example argumentation framework annotated according to Grounded (on the left) and Stable (on the right) semantics. Green represents arguments to be accepted. Red arguments to be rejected. Grey arguments whose status cannot be decided. 9

2.3 Three different extensions that are all acceptable in an example argumentation framework. Acceptable arguments in green. Red indicate arguments that are not acceptable. Grey arguments cannot be determined. 11

2.4 This framework has the following extensions by semantics. Grounded and ID={(a)}, Complete={(a),(a,c),(a,d)}, Stable and Semi-Stable={(a,d)}, Preferred={(a,c),(a,d))} 14

2.5 Example of adjacencey matrix based on Figure 2.1 23

2.6 Results of Various DNN Architectures on Premise Selection Task, reproduced from [59] 26

3.1 Overview of the GCN Architecture used for the experiments in the SAFA 2020 version. 81

3.2 Overview of how input features are combined during training of the AFGCN solver 82

3.3 Overall process for data processing and training of the AFGCN solver . . . 82

3.4 Overview of the process for generating random training batches. 83

3.5 Illustration of the dynamic balancing process used by AFGCN 84

3.6 Runtime solver architecture for the solver used in the AFGCN experiments 85

(a) Relationship between Runtime Solver, Grounded Solver and GCN . . 85
(b) Runtime solver system architecture 85
3.7 Network architecture for the solver used in the results section for AFGCN . 85
3.8 Overview of AFGCN approximation results for DC-PR 86
3.9 Overview of AFGCN approximation results for DC-CO 87
3.10 Overview of AFGCN approximation results for DC-ST 88
3.11 Overview of AFGCN approximation results for DC-SST 89
3.12 Overview of AFGCN approximation results for DC-STG 90
3.13 Overview of AFGCN approximation results for DS-PR 91
3.14 Overview of AFGCN approximation results for DS-CO 92
3.15 Overview of AFGCN approximation results for DS-ST 93
3.16 Overview of AFGCN approximation results for DS-SST 94
3.17 Overview of AFGCN approximation results for DS-STG 95
3.18 Overview of AFGCN approximation results for DS-ID 96
3.19 Runtime distribution by semantics for the AFGCN solver experiments . . . 97
(a) w/GR 97
(b) No GR 97
3.20 Runtime distribution by benchmark for the AFGCN solver experiments . . 98
(a) w/GR 98
(b) No GR 98
3.21 Runtime distribution by size for the AFGCN solver experiments 99
(a) w/GR 99
(b) No GR 99

4.1 Network architecture for co-admissibility prediction 105
4.2 Solver architecture for predicting co-admissible sets 108
4.3 Processing pipeline for co-admissibility prediction 109
4.4 Results for the co-admissibility driven solver on ICCMA 2017 test cases . . 111

5.1 Examples of using t-SNE visualisation on three commonly examined datasets.115
(a) MNIST with t-SNE 115
(b) CORA with t-SNE 115
(c) Word2Vec with t-SNE 115
5.2 Visaulization of the Sembuster Scheme, reproduced from [12] 117

5.3 T-SNE Visualisation of Sembuster Graphs. Cyan represents partition A, Red partition B, and Blue partition C. Partitions are indicated in figure 5.1, k references the definition from figure 5.1 118
(a) k=600 118
(b) k=1200 118
(c) k=1800 118
5.4 Graph-level Visualisation using t-SNE across benchmarks in table 5.1, grouped by similarity of embedding 120
5.5 Overview of visualizations of AFGCN embeddings. Logic Based Argumentation highlighted in red 122
5.6 Overview of visualizations of AFGCN embeddings. ABA2AF highlighted in red 122
5.7 Overview of visualizations of AFGCN embeddings. AFGEN highlighted in red 123
5.8 Overview of visualizations of AFGCN embeddings. High performing benchmarks on top vs low performing benchmarks on bottom. Relevant group highlighted in red 124

6.1 Methodology for the misinformation detection task 129
6.2 Scheme diagrams, supporter = positive stance, commenter/querier = neutral stance, denier = negative stance 131
(a) Scheme 1 131
(b) Scheme 2 131
(c) Scheme 3 131
(d) Scheme 4 131
(e) Scheme 5 131
(f) Scheme 6 131
(g) Scheme 7 131
6.3 GCN architecture used in misinformation detection experiments. 133
6.4 Evaluation of experiments on RumourEval dataset using GCN based misinformation detection 135
(a) Results graph 135
(b) Results table 135

List of Tables

2.1 Decision problems in argumentation frameworks. Reproduced from [83] . . 28
2.2 Main complexity results for abstract argumentation reproduced from [36] . 28

3.1 Abbreviations of problems and semantics used extensively in chapter 3 . . . 37
3.2 Models trained for the SAFA 2020 experiments broken down by key features. 44
3.3 Approximation results for the SAFA 2020 model on the credulous acceptability problem for DC-PR 45
3.4 Approximation results for the SAFA 2020 model on the credulous acceptability problem for DS-PR 46
3.5 Approximation results for the SAFA 2020 model on the acceptability problem ordered by network depth 46
3.6 Approximation results for the SAFA 2020 model on the acceptability problem ordered by training regime 47
3.7 Description of benchmarks used for the AFGCN solver evaluation 48
3.8 Characteristics of the test set used for AFGCN evaluation 49
3.9 Overview of AFGCN approximation results for DC-PR ordered by benchmark 54
3.10 Overview of AFGCN approximation results for DC-PR ordered by band . . 54
3.11 Overview of AFGCN approximation results for DC-CO ordered by benchmark 56
3.12 Overview of AFGCN approximation results for DC-CO ordered by band . . 56
3.13 Overview of AFGCN approximation results for DC-ST ordered by benchmark 58
3.14 Overview of AFGCN approximation results for DC-ST ordered by band . . 58
3.15 Overview of AFGCN approximation results for DC-SST ordered by benchmark 59
3.16 Overview of AFGCN approximation results for DC-SST ordered by band . 60
3.17 Overview of AFGCN approximation results for DC-STG ordered by benchmark 62

3.18 Overview of AFGCN approximation results for DC-STG ordered by band . 62
3.19 Overview of AFGCN approximation results for DS-PR ordered by benchmark 63
3.20 Overview of AFGCN approximation results for DS-PR ordered by band . . 64
3.21 Overview of AFGCN approximation results for DS-CO ordered by benchmark 66
3.22 Overview of AFGCN approximation results for DS-CO ordered by band . . 66
3.23 Overview of AFGCN approximation results for DS-ST ordered by benchmark 67
3.24 Overview of AFGCN approximation results for DS-ST ordered by band . . 67
3.25 Overview of AFGCN approximation results DS-SST ordered by benchmark 69
3.26 Overview of AFGCN approximation results for DS-SST ordered by band . . 69
3.27 Overview of AFGCN approximation results for DS-STG ordered by benchmark 71
3.28 Overview of AFGCN approximation results for DS-STG ordered by band . 71
3.29 Overview of AFGCN approximation results DS-ID ordered by benchmark . 73
3.30 Overview of AFGCN approximation results for DS-ID ordered by band . . . 73
3.31 Overview of AFGCN approximation results compared across semantics . . . 74
3.32 Overview of AFGCN approximation results compared across benchmarks for all semantics 75
3.33 Overview of AFGCN approximation results compared across size bands for all semantics 76
3.34 Overview of AFGCN runtime results, key statistics 76
3.35 Overview of AFGCN runtime results ordered by semantics. Median runtime given. Results in seconds. 77
3.36 Overview of AFGCN runtime results ordered by benchmark. Median given. Results in seconds. 78
3.37 Overview of AFGCN runtime results ordered by size. Median given. Results in seconds. 78

4.1 Overview of results from predicting co-admissibility with adapted AFGCN . 109
4.2 Overview of using co-admissibility as a SAT heuristic to guide the solution of abstract argumentation problems 110

5.1 Argumentation problem domains for graph-level visualisation experiment . 119

6.1 Key statistics of the RumourEval dataset used for the experiments in Chapter 6, reproduced from [39] 134

A.1 Results DC-PR - equal weighting 143
A.2 Results DC-PR - complete balanced 143
A.3 Results DC-PR - reduced balanced 144
A.4 Results DS-PR - equal weighting 144
A.5 Results DS-PR - complete balanced 144
A.6 Results DS-PR - reduced balanced 144
A.7 Results DC-CO - equal weighting 145
A.8 Results DC-CO - complete balanced 145
A.9 Results DC-CO - reduced balanced 145
A.10 Results DS-CO - equal weighting 145
A.11 Results DS-CO - complete balanced 146
A.12 Results DS-CO - reduced balanced 146
A.13 Results DC-ST - equal weighting 146
A.14 Results DC-ST - complete balanced 146
A.15 Results DC-ST - reduced balanced 147
A.16 Results DS-ST - equal weighting 147
A.17 Results DS-ST - complete balanced 147
A.18 Results DS-ST - reduced balanced 147
A.19 Results DC-SST - equal weighting 148
A.20 Results DC-SST - complete balanced 148
A.21 Results DC-SST - reduced balanced 148
A.22 Results DS-SST - equal weighting 148
A.23 Results DS-SST - complete balanced 149
A.24 Results DS-SST - reduced balanced 149
A.25 Results DC-STG - equal weighting 149
A.26 Results DC-STG - complete balanced 149
A.27 Results DC-STG - reduced balanced 150
A.28 Results DS-STG - equal weighting 150
A.29 Results DS-STG - complete balanced 150
A.30 Results DS-STG - reduced balanced 150
A.31 Results DS-ID - equal weighting 151
A.32 Results DS-ID - complete balanced 151
A.33 Results DS-ID - reduced balanced 151

Acknowledgements

First, I would like to acknowledge the unfailing support of my wife Damiana and the inspiration of my children Ada and Pino. Thank you for putting up with me throughout these past years.

Second, I want to thank my supervisors Dr. Tommy Yuan and Dr. Peter Nightingale for their exceptional support throughout the research and thesis writing process. Between the two of you, I could not have asked for better guidance.

Finally, I also want to thank my first supervisor Dr. Suresh Manandhar for his invaluable support during the early part of the research.

Declaration

I declare that the research described in this thesis is original work, which I undertook at the University of York during 2018 - 2022. Except where stated, all of the work contained within this thesis represents the original contribution of the author.

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References.Some parts of this thesis have been published in conference proceedings; where items were published jointly with collaborators, the author of this thesis is responsible for the material presented here. For each published item the primary author is the first listed author.

- Malmqvist, L. (2021). AFGCN: An Approximate Abstract Argumentation Solver. *International Competition in Computational Models of Argumentation (ICCMA) 2021.* http://argumentationcompetition.org/2021/downloads/afgcn.pdf [63]

- Malmqvist, L. (2021). Approximate Solutions to Argumentation Frameworks with Graph Neural Networks. *Online Handbook of Argumentation for AI*, 32-36. [64]

- Malmqvist, L., Yuan, T., & Manandhar, S. (2021). Visualising Argumentation Graphs with Graph Embeddings and t-SNE. *COMMA Workshop on Argument Visualization*, online. https://argvis-workshop.lingvis.io/pdfs/ArgVis2020_paper_1.pdf [65]

- Malmqvist, L., Yuan, T., & Nightingale, P. (2021). Improving misinformation detection in tweets with abstract argumentation. *Computational Models of Natural Argument 2021, CEUR Workshop Proceedings, 2937*, online, pp. 40–46. [66]

- Malmqvist, L., Yuan, T., Nightingale, P., & Manandhar, S. (2020). Determining the acceptability of abstract arguments with graph convolutional networks. *System and*

Algorithms for Formal Argumentation 2020, CEUR Workshop Proceedings, 2672, online, pp. 47–56. [67]

Abbreviations

AFGCN	Argumentation framework graph convolutional network
ASP	Answer set programming
CNN	Convolutional Neural Network
CO	Complete semantics
DC	The credulous decision problem
DNN	Deep Neural Network
DS	The sceptical decision problem
GCN	Graph Convolutional Network
GNN	Graph Neural Network
ICCMA	International Competition in Computational Models of Argument
ST	Grounded semantics
LSTM	Long Short-Term Memory
PR	Preferred semantics
RNN	Recurrent Neural Network
SAT	The boolean satisfiabiltiy problem
SST	Semi stable semantics
ST	Stable semantics

STG	Stage semantics

Chapter 1

Introduction

1.1 Background and rationale

Abstract argumentation is a research field that formalises certain notions in the wider field of argumentation that makes certain notions about the acceptability of arguments under conflict precise and amenable to algorithmic determination. It has been used in a wide number of fields to reason about the acceptability of argumentative structures and holds great promise as a formal approach to certain reasoning tasks in Artificial Intellligence (AI). Examples include legal argumentation [82], where abstract argumentation can help model reasoning under disagreement, the structure of argument systems, and the development of dispute tactics. Applications have also been also found in Intelligence Analysis [74] to determine what intelligence is internally consistent, in multi-agent systems as a part of communication protocols [102], and even in maritime safety [75] to determine consistency of sensor readings.

This thesis presents a state-of-the-art method for approximating problems in abstract argumentation using Graph Neural Networks (GNNs). Fundamentally, abstract argumentation is a formalism for non-monotonic reasoning that bases its representation on the modelling of conflict. It is typically represented in the form of a directed graph in which vertices represent arguments that can represent anything that can stand in a relationship of conflict and edges that represent a relation of attack between these arguments. See figure 1.1 for an example. We will cover the detail of this formalism in Chapter 2, but, importantly, this representation gives rise to a range of reasoning problems which determine the acceptability of arguments or the joint acceptability of sets of arguments. The majority of these reasoning problems are known to be NP-hard [21, 101].

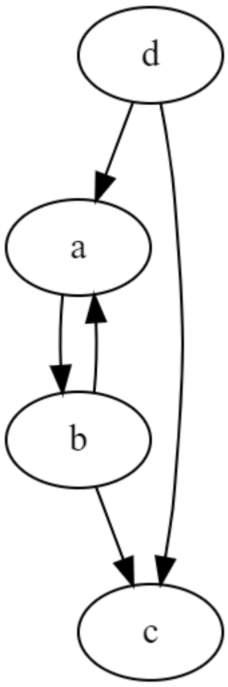


Figure 1.1: Example of a simple abstract argumentation framework.

Our core research method consists in merging research strands within deep learning with research on abstract argumentation. Our starting point is that Graph Neural Networks [103] of various types (e.g. Graph Convolutional Networks [51]) are well suited to addressing the graph-structured problems encountered in abstract argumentation. That means finding ways of approximating solutions to tasks involving argumentation frameworks and queries that because of their size or graph topology are challenging for existing solvers to address.

The vast majority of solution approaches in abstract argumentation are exact and complete, often using reduction to some other formalism such as SAT [16] or ASP [17] to solve the problem. In general, SAT-based approaches have achieved the best solver performance, but this type of approach does not necessarily provide the easiest fit with real-world deployment scenarios and very large datasets.

There are, therefore, major benefits to approximate approaches such as one using Graph Neural Networks in this area that makes them worth considering. In particular, they potentially have greatly superior run-time performance and are more easily integrated into parallelised Cloud-based architectures.

In general, if any of the following properties are true of a system that deploys abstract argumentation as a formalism for reasoning about some problem, then an approximate approach such as ours may be useful:

- The system needs to solve argumentation problems, reliably but not perfectly, in near real-time based on complex and/or large graphs. In this scenario, approximation helps address the gaps that exist in current solvers, none of which perform perfectly across different benchmarks.

- The system needs to be able to quickly generate answers for all arguments in one or more argumentation frameworks, reliably but not perfectly, in near real-time. In this scenario, approximation helps improve on the lack of parallel processing in most current solvers that tend to provide answers for a single argument only through the ability of the GNN to provide approximate answers across all arguments in an argumentation framework in a single inference step.

- The system can use a preliminary answer in many circumstances and only requires exact answers in certain cases. In this case, approximation complements exact approaches and can also be used as a heuristic to guide their execution.

Our general contention is that approximation extends the range of use cases that can be addressed using abstract argumentation by guaranteeing high runtime performance with good, although not perfect, accuracy. We will see one example of such a use case, when considering Twitter data in Chapter 6, but any system that matches any of the three characteristics above would benefit.

1.2 Aim and objectives

The overarching aim of the thesis is to develop an approximate approach to solving abstract argumentation problems and to extend that approach in a number of directions that enrich and inform the research and its applications. We entered into this research with a hypotheses that Graph Neural Networks (GNNs) could provide a good method and architecture for approximating solutions to argumentation problems.

We then framed these aims as a core research questions:

RQ: Can you improve on current solution methods in abstract argumentation using Deep Neural Networks (DNNs) in particular GNNs.

Based on that overarching question, seven sub-questions (SQ) were formulated during the research to explore different areas of the wider problem. These sub-questiosn, we will refer back to in Chapters 3-6.

SQ 1 Is it possible to develop a high-performing approximate solver for abstract argumentation using GNN methods?

SQ 2 What neural network architecture and training methods work most effectively for the purposes of approximating abstract argumentation problems?

SQ 3 Given the polynomial solvability of the grounded extension, can we use that as a starting point to improve approximation?

SQ 4 Are there significant differences in the approximability of argumentation frameworks based on semantics, benchmark type or size?

SQ 5 Can the output of an approximate solver be used as a heuristic to improve the performance of exact solution methods?

SQ 6 Does visualising the structure of the GCN model used for approximation give us additional insight into the way it functions?

SQ 7 Can we extend the basic architecture of the approximate solver to other related problems with similar structure?

These research questions will be addressed in the following chapters.

1. SQ 1-4 will be addressed in Chapter 3.
2. SQ 5 will be addressed in Chapter 4.
3. SQ 6 will be addressed in Chapter 5.
4. SQ 7 will be addressed in Chapter 6.

1.3 Key contributions

Throughout this thesis, we will highlight specific contributions that we will be able to make as part of the research presented. A list of these key contributions can be found below:

- We develop a new approach to approximating solutions to problems in abstract argumentation, using GNNs. This approach set a new state-of-the-art and won 4 out of 6 categories in the ICCMA 21 competition for approximate argumentation solvers
- We systematically investigate the properties of this approach across argumentation semantics and benchmarks, and also conducted ablation studies to determine the key parameters for the neural network architecture
- We link the approximate approach described with exact approaches by using the approximate solutions as an input to a SAT based solution approach and an experimental, but exact, algorithm
- We present a new visualisation approach for argumentation frameworks, leveraging the GNN architecture that we developed for approximation, and used it to derive additional insights about our neural network model
- We present an application of the GNN model that we developed for approximation to the classification of misinformation in Twitter data. Although we didn't set a new state-of-the-art on the key datasets, we demonstrated that the approach we use for approximating acceptability in argumentation frameworks has wider applicability

1.4 Structure of the thesis

The remainder of the thesis is structured as follows:

- In Chapter 2, we present the necessary background that will enable the reader to follow the rest of the material. We also place our research in the context of current research trends to clarify our points of departure.
- Chapter 3 presents the core contribution of this thesis, a state-of-the-art approximation approach for abstract argumentation. This chapter describes the overall architecture of our approach and then systematically goes through the experimental results that we have obtained from it.
- Chapter 4 presents the link between approximate and exact approaches by using approximate solutions as input to a SAT solver, using a polarity heuristic, and to an exact search algorithm as a branching heuristic.

- In Chapter 5, we deepen our understanding of our approximation approach by applying a new visualisation method to the underlying weights of the neural network model, which gives us new insight into the different results obtained for certain benchmarks.
- Chapter 6 provides an application of our approach. Here, we adapt our GNN architecture to be able to predict misinformation in tweets, using a constructed argumentation framework and a large scale language model in tandem.
- The final chapter, Chapter 7, is the conclusion.

Chapter 2

Background and Related Work

This chapter will review the background material and related work necessary to understand the context for the contributions that will follow in later chapters. We will start by surveying the general field of computational argumentation, then zoom into the specific subfield of abstract argumentation that is most relevant to the work in this thesis. Then we move on to the second major strand of background material, where we start by covering general topics relevant to deep learning. Then we cover graph convolutional networks, the key area of relevance for this thesis in more detail. Finally, we combine the two strands when looking at current solution approaches for problems in abstract argumentation.

2.1 Computational argumentation

Computational argumentation is a major sub-field of AI research that seeks to provide techniques for evaluating the claims or conclusions that can be drawn from a representation of elements usually set up in some relationship of conflict.

In human argumentation this amounts to reasons for and against a given proposition, but in a computational context it is more profitable to view the field as studying abstract systems of conflict with elements not necessarily representing arguments in a traditional sense, but rather conflictables that stand in some defined relationship to one another and that in total define a logical framework from which some conclusions may be drawn dependant on a set of rules of interpretation [5]. This also expands the fields of potential applications beyond traditional areas such as the analysis of legal cases to any situation that can be represented as atoms in conflict, for instance social media or conflicting sensor data.

Originally, the problems of argumentation were studied within philosophy departments and much of the research that has gone into computational argumentation has come from attempts to operationalise such principles [97]. However, since Dung's seminal 1995 paper [29], one of the most important computational paradigms has been that of abstract argumentation, a minimal formalism where the only elements to consider are a set of argument and a relation of attacks between these arguments, defining a digraph as shown in figure 2.2 below.

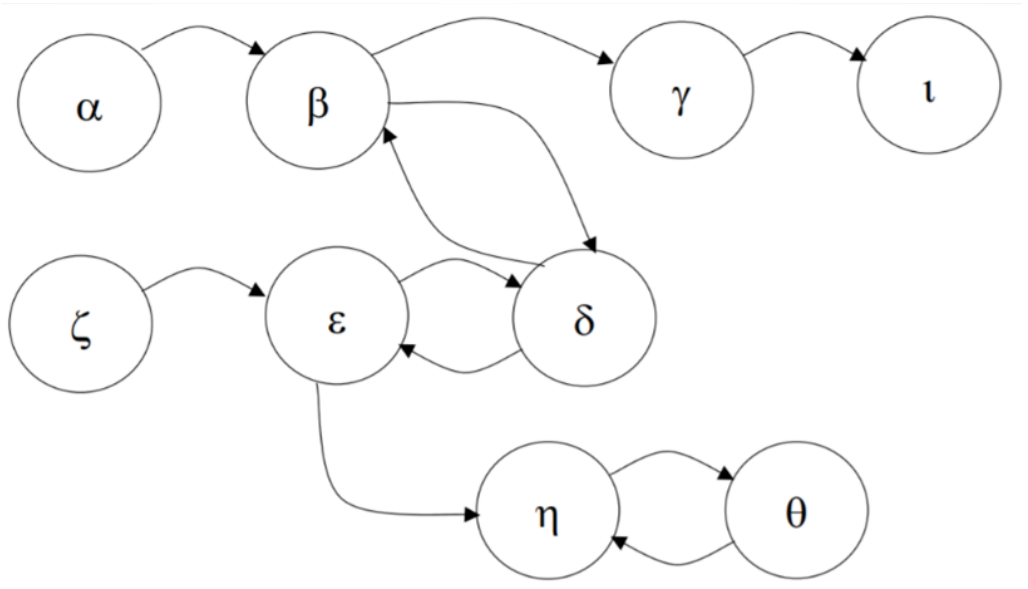


Figure 2.1: Example of an Argumentation Framework. Nodes represent arguments in the argumentation framework. Edges represent attacks between arguments. [5].

This formalism can be interpreted according to different sets of rules, called semantics within the field, that potentially lead to different conclusions to be drawn from the frameworks. For instance, under the Grounded semantic one accepts only arguments that are absolutely certain given an argumentation framework, whereas under the Stable semantics one accepts only sets of arguments that collectively attack all other arguments in that framework that aren't part of the solution, an approach summed up by the phrase "you're either with us or against us".

Abstract argumentation is at this point a well-studied field [101], although it is also true that because of the intractable nature of many core problems in the field, there is still scope for improvement of the techniques used to solve argumentation frameworks under this formalism [21, 36].

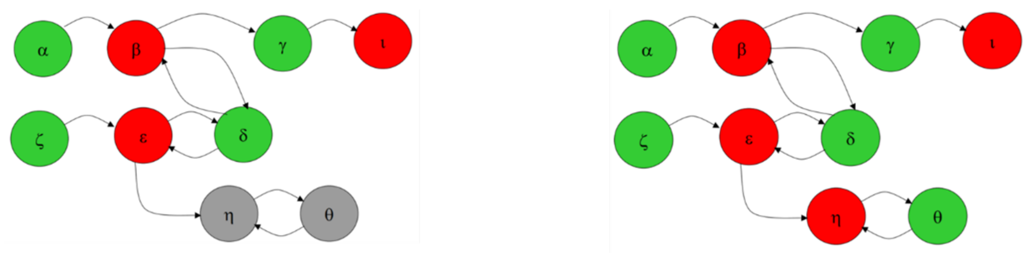


Figure 2.2: Example argumentation framework annotated according to Grounded (on the left) and Stable (on the right) semantics. Green represents arguments to be accepted. Red arguments to be rejected. Grey arguments whose status cannot be decided.

Many extensions have been proposed to the basic formulation of abstract argumentation including extending it with rankings [10], probabilities [93], and differentiated notions of support and attack [8, 15]. There are also more elaborate argumentation systems such as ASPIC+ [71] that include several of these elements but retain a certain compatibility with the basic abstract argumentation formalism.

Few of these approaches have gathered as much interest as Dung's original formulation, but that does not imply that such approaches aren't required or that all relevant argumentation scenarios can be accurately modelled using Dung's approach. It is perhaps more a question of the suitability of methods to solve such frameworks that provides the limiting factor at present.

Abstract argumentation has the useful feature of being readily reducible to well-known problems such as SAT or Answer Set Programming [21], where existing solvers exist with many years of supporting research. This is not always true for more elaborate argumentation systems, which can make their solution harder.

We will, therefore, proceed to discuss the properties of abstract argumentation in greater detail.

2.2 Abstract argumentation

Abstract argumentation is a way of formalising the representation of conflicting claims [5] using an intentionally minimalist approach. In abstract argumentation, (abstract) arguments are composed into argumentation frameworks that contain only the arguments themselves and the relationships of conflict between them. The "abstract" in the title should be taken quite seriously. While the formalism has its origin in the study of argumentation, it can and has been used to represent a variety of different situations where conflict is of the essence including, for instance maritime safety [75] and intelligence analysis [74].

The origin of abstract argumentation as mentioned is in a seminal paper by Dung [29] that presented the general theory of argumentation frameworks and related them to a number of other logical formalisms. In the following part of this section, I will go through some of the key definitions from this paper in order to cover the necessary background for the subsequent discussions of the rules of interpretation to which one can subject argumentation frameworks and the approaches one can take to solving them. Definitions here are given using an extension based approach, but an equivalent labelling based approach is equally common in the literature [80].

2.2.1 Definitions

Definition 2.2.1 (Argumentation framework) *An argumentation framework is a tuple, $F = \langle args, atts \rangle$ in which args is a finite set of arguments and $atts \subseteq args \times args$ defines a relation of attack.*

To say that A1 attacks A2 is hence the same as saying that $(A1, A2) \in atts$. If $S \subseteq args$ and $A \in args$ we can extend this nomenclature by saying that A attacks S iff there exists $B \in S$ such that $(A, B) \in atts.$. In a parallel manner we can say that S attacks A iff there exists $B \in S$ such that $(B, A) \in atts$.

We can also define a similar notion of defence.

Definition 2.2.2 (Defence) *An argument $A \in args$ is defended by a set $S \subseteq args$ if, for each $B \in args$ such that $(B, A) \in atts$, there exists a $C \in S$ such that $(C, B) \in atts$.*

Definition 2.2.3 (Attacking and attacked) *The set of all attacking arguments of a subset of an argumentation framework can be written as $S^- = \{B \mid \exists A \in S : (B, A) \in atts\}$. The set of all attacked arguments can be written as $S^+ = \{B \mid \exists A \in S : (A, B) \in atts\}$*

This notation is convenient in that it also lets us define notions of range and negative range.

Definition 2.2.4 (Range and negative range) *The range of S can be defined as $S \cup S^+$ The negative range of S can be defined as $S \cup S^-$.*

The range is thus the union of a set and those arguments attacked by that set, whereas the negative range is a set and all arguments that attack the set.

Definition 2.2.5 (Characteristic function) *Given an argumentation framework $F = \langle args, atts \rangle$, the characteristic function $F : 2^{args} \rightarrow 2^{args}$ of F is defined as $F_F(S) = \{x \in args \mid \text{x is defended by S}\}$.*

The characteristic function returns the set of arguments defended by a given subset of the argumentation framework. The last basic concepts required before we can move on from this section are those of conflict-freeness, acceptability, and admissibility.

The notion of acceptability is key to abstract argumentation.

Definition 2.2.6 (Acceptability) *An argument $A \in args$ is called acceptable with respect to an extension $ext \subseteq args$ iff for every $B \in args$ with B attacks A there is an argument $A' \in ext$ with A' attacks B.*

That means there can be multiple acceptable extensions in an argumentation framework. See figure 2.3 for an example.

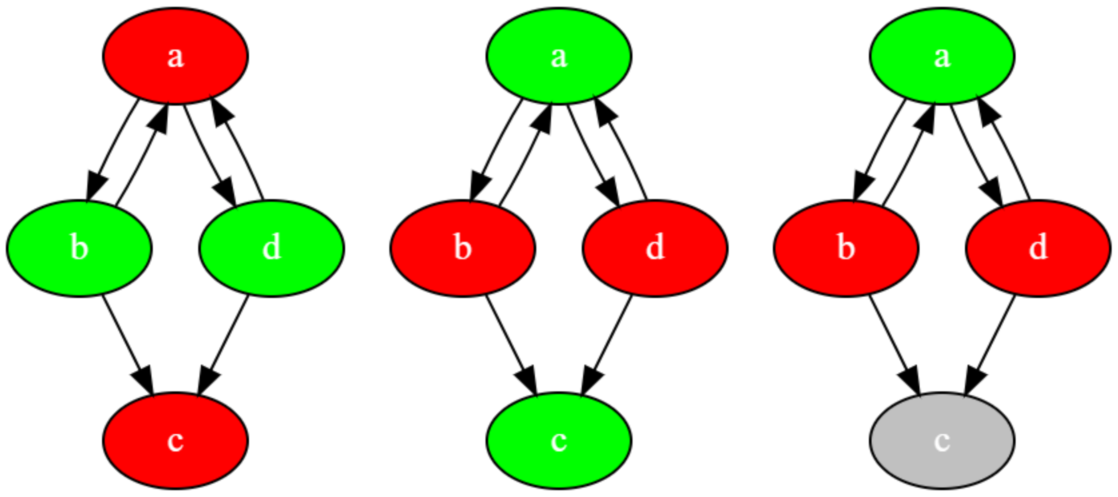


Figure 2.3: Three different extensions that are all acceptable in an example argumentation framework. Acceptable arguments in green. Red indicate arguments that are not acceptable. Grey arguments cannot be determined.

Extensions are subsets of arguments that are collectively acceptable. Extensions are evaluated based on semantics that define rules for which sets of arguments can be accepted together. Almost all semantics require the property of conflict-freeness.

Definition 2.2.7 (Conflict-freeness) *Given an argumentation framework $F = \langle args, atts \rangle$, a given subset, S, of this argumentation framework, is said to be conflict-free iff there does not exist $(A, B) \in atts$ with A, $B \in S$.*

The notion of conflict-freeness implies that there are no internal conflicts within acceptable solutions and is a building block of all semantics.

Definition 2.2.8 (Admissibility) *A subset, S, of an argumentation framework $F = \langle args, atts \rangle$ is admissible if it is conflict-free and $S \subseteq F_F(S)$.*

That definition states that an admissible set is a set that defends itself from all attacks given the definition of defence in definition 2.2.2.

2.2.2 Argumentation Semantics

The semantics of an argumentation framework define the rules under which a set of arguments can be accepted. That is to say which sets of arguments contained in an argumentation framework can be said to constitute acceptable solutions under the given semantic. The original paper by Dung (1995) defined the four "classic" semantics: Grounded, Complete, Preferred, and Stable.

However, a number of additional semantics have been proposed. In the following, we will cover the four "classic" semantics and three additional semantics: Ideal [28], Stage [96], and Semi-stable [14] that all feature prominently in the literature. This will be done based on the definitions covered above, but I will also attempt to give a more "qualitative" account of what each semantic encapsulates and how they differ in the kinds of solutions they allow. The derivations of each semantic is kept to a minimal, but complete level. More elaborate derivations exist in the papers cited above.

2.2.2.1 Complete Semantics

The most basic semantics for most purposes are Complete semantics. Many other common semantics are special cases of Complete semantics. Complete semantics can be defined as a fix point of a conflict-free subset of an argumentation framework. That is to say for a subset S of an argumentation framework, F, the subset forms a Complete extension iff $F_f(S) = S$ and S is conflict-free. That means in practical terms that a Complete extension is an extension that defends itself and also contains all the elements defended by the extension. Qualitatively, one can think of a Complete extension as a reasonable or at least defensible position given the evidence.

2.2.2.2 Grounded Semantics

The Grounded extension is the subset-minimal complete extension. That is to say that if S is a Complete extension then it is also grounded iff there does not exist another complete extension, $C \mid C \subset S$. Qualitatively, one can think of the grounded extension as the most sceptical position one can take vis-à-vis the evidence.

2.2.2.3 Preferred Semantics

A Preferred extension in contrast is a subset-maximal Complete extension. That is to say that if S is a Complete extension then it is also preferred iff there does not exist another complete extension, $C \mid S \subset C$. A preferred extension can be thought of as a position that tries to incorporate as much as possible of the available evidence in formulating a defensible position.

2.2.2.4 Ideal Semantics

In many cases, the scepticism of the Grounded extension proves too severe for practical applications and a slightly less severe form of scepticism is called for. This is provided by the Ideal semantic, which can be defined as the largest admissible subset of an argumentation framework in which all the elements are members of every preferred extension. This is still a sceptical position, but can vary from the Grounded extension.

2.2.2.5 Stable Semantics

While the Stable extension can also be shown to be a Complete extension it is not usually defined as such. Instead, a Stable extension, which may or may not exists for a given argumentation framework, is defined as a conflict-free extension whose range is equal to the total set of arguments in the argumentation framework. That is to say, the stable extension takes the "if you're not with us you're against us" approach by ensuring that every argument is either a member of the extension or attacked by the extension.

2.2.2.6 Semi-Stable and Stage Semantics

The completeness of the Stable extension's binary division of the argumentation framework is a desirable feature in some applications, where undecidability is an issue. However, the Stable extension does not exist in all argumentation frameworks. Therefore, two alternatives have been proposed that try to maximise the range of extensions, but can be shown to exist for all argumentation frameworks.

The first of these, Stage semantics, can be defined formally as the conflict-free set that maximises range. That is to say S is a Stage extension [12] iff S is conflict-free and there does not exist a conflict-free extension C such that $S^+ \subset C^+$. The semi-stable semantic is defined similarly only it starts from an admissible set rather than a conflict-free one [14].

The difference between the evaluation rules leads to significant differences in the computational approaches that are taken towards them. While many solvers deploy a similar starting point for generating solutions across semantics, there is substantial differences in how they are computed leading to high variability in performance. The complexity of reasoning tasks equally vary substantially by semantic [32,36] and it is not always the case that similar semantics have similar computational complexity, which we'll discuss later in this chapter.

2.2.2.7 Summary of semantics

These seven semantics can be summarized as per the table below [80]. Some examples are shown in figure 2.4:

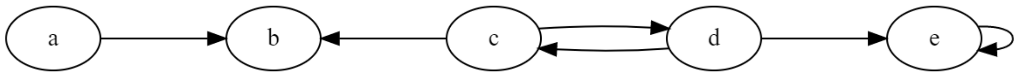


Figure 2.4: This framework has the following extensions by semantics. Grounded and ID={(a)}, Complete={(a),(a,c),(a,d)}, Stable and Semi-Stable={(a,d)}, Preferred={(a,c),(a,d))}

These are all the semantics we shall concern ourselves with in this thesis and are also the ones used for recent ICCMA competitions (2017, 2019, 2021).

Semantics	**Definition**
Complete	An extension S is Complete iff it is admissible and it includes all arguments that it defends.
Grounded	An extension S is Grounded iff it is complete and subset minimal.
Preferred	An extension S is Preferred iff it is complete and subset maximal.
Stable	An extension S is Stable iff it is conflict-free and $S \bigcup S^+$ contains all arguments in the argumentation framework.
Semi-Stable	An extension S is Semi-stable if it is complete and $S \bigcup S^+$ is subset maximal.
Stage	An extension S is Stage iff it is conflict-free and $S \bigcup S^+$ is subset maximal.
Id	An extension S is Id iff it is an admissible subset of all preferred extensions and is subset maximal.

2.3 Deep learning

2.3.1 Basic concepts

In this section, we will go through the basics of Artificial Neural Networks (ANN) that base themselves on artifical neurons layered together to learn a function from data.

2.3.1.1 Multi Layer Perceptron

The basic model for deep learning is the Multi Layer Perceptron. A Multi Layer Perceptron (MLP) [25] is a feedforward artificial neural network model that maps sets of input data onto a set of outputs based on a learned function. An MLP consists of multiple layers of nodes, where each layer is fully connected to the next one in a computational graph. Excepting input nodes, all nodes use an activation function to compute its output.

An activation function of an artificial neuron is a function that maps the net input of the neuron to its output. Typically, the activation function of an artificial neuron is chosen to be some sort of continuous mathematical function like the sigmoid function, the hyperbolic tangent, or the rectified linear unit [38]. For these functions, the output is typically seen to be the firing tendency, or activation of the neuron.

MLPs use a supervised learning method called backpropagation to train based on input data. Backpropagation in neural network is a short form for backward propagation of errors. Overall, it is the standard method of training neural networks. Given an artificial neural network and a loss function, the method calculates the gradient of the loss function with respect to all the weights in the network. This information is then used to calculate changes to the weights in order to optimize the loss function further. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron and it can distinguish data that is not linearly separable [38].

2.3.1.2 Other basic Deep Neural Network (DNN) concepts

There are a variety of DNN specific terminology that relates to the architecture and training of DNNs, which are relevant to the understanding of the material development in this thesis. Below we define these key terms.

RELU

The Rectified Linear Unit (RELU) is a nonlinear function that is used in neural networks to introduce nonlinearity into the network. It is defined as:

$f(x) = max(0, x)$

RELU has been shown to improve the convergence of neural networks and can help prevent overfitting [38].

Dropout

Dropout is a statistical technique for reducing the amount of variance in a data set. It is typically used in machine learning and can be applied to data sets with a high degree of collinearity. Dropout is used to prevent overfitting by randomly dropping some data points from the data set. This reduces the chance of the model being too closely fitted to the data, and results in a more generalisable model [38].

Repeating blocks

A repeating block in a DNN architecture is a sequence of layers that is repeated as a block multiple times over in the neural network architecture. This often involves the combination of a computational layer and layers that either use droupout or some form of normalisation [38].

Residual connection

A residual connection is a type of skip connection in which the input to a layer is directly connected to the output of that layer, without passing through any intermediate layers. This type of connection can be useful in preventing the vanishing gradient problem and can improve the training speed of deep neural networks [38].

Loss function

A loss function is a mathematical function that calculates the error between predicted values and actual values. The loss function is used to minimize the error in order to improve the accuracy of the predictions [38].

Learning rate

The learning rate is the rate at which a neural network learns from training data. It is a parameter that controls how much the weights of the network are updated in response to the error gradient [38].

ADAM

ADAM (Adaptive Moment Estimation) is a method for training neural networks that is based on gradient descent. The main difference between ADAM and other methods is that it uses a different learning rate for each parameter, which is adapted based on the parameter's gradient. This allows the learning rate to be automatically adjusted as the training progresses, which can lead to faster and more efficient training [38].

2.3.1.3 Different types of neural networks

The following sections explain some of the different types of neural network architectures referenced in this thesis. As this section concerns network architectures that aren't central to the main thrust of the thesis, the explanations will be relatively brief. However, they are necessary to completely understand the content to follow in this and later chapters.

RNN

RNNs are a type of neural network that can process sequential data. This makes them well-suited for tasks such as text classification and language translation.

RNNs work by propagating information through time. That is, they take as input a sequence of vectors, and produce a corresponding sequence of vectors as output. The vectors can represent anything, such as words in a sentence or pixels in an image.

To compute the output of an RNN, we first need to specify a set of weights. This set of weights will be shared by all time steps, and will be updated during training. The weights are used to compute a new vector at each time step, which is then propagated to the next time step.

At each time step, the RNN takes as input the current input vector and the previous output vector. The output vector is then computed by combining the input vector and the previous output vector using the weights. This process is repeated for each time step [38, 85].

Historically, RNNs have been used mainly for tasks in Natural Language Processing (NLP). However, they are not used commonly in this field anymore, due to the dominance of large-scale language models.

CNN

CNNs are a type of neural network that are used to process data that has a grid-like structure, such as images. CNNs are made up of a series of layers, where each layer is made up of a series of neurons. The first layer of a CNN is the input layer, which is where the data is fed into the network. The next layer is the hidden layer, which is where the data is processed by the neurons. The last layer is the output layer, which is where the processed data is outputted.

CNNs are very effective at processing images, as they are able to extract features from the data that is fed into them. CNNs are also very efficient, as they only need to process the data that is fed into them, and do not need to store the data in memory. This makes CNNs very fast, and able to process large amounts of data [38, 85].

LSTM

LSTM is a type of recurrent neural network that is well-suited to modeling data that is in the form of a sequence. This is because LSTM is able to remember information for long periods of time, and can therefore keep track of patterns in data over extended periods of time.

LSTM networks are composed of a number of LSTM cells, which are themselves composed of a number of neurons. Each cell takes as input a vector of data, and outputs a new vector. The cells are interconnected such that the output of each cell is fed as input into the next cell in the network.

LSTM networks are trained using a variation of the backpropagation algorithm. In each training iteration, the network is presented with a sequence of data. The network then produces an output sequence, which is compared to the desired output sequence. The error is then backpropagated through the network, and the weights of the cells are updated in order to minimize the error [38, 85].

WaveNet

WaveNet is a neural network that is used to generate raw audio waveforms. It is composed of a series of 1-D convolutional layers, which are used to model the raw audio waveforms. The WaveNet model is trained by using a set of audio recordings, which are then used to generate new audio waveforms. The generated audio waveforms can be used to synthesize new sounds, or to improve the quality of existing sounds.

WaveNet has been shown to generate audio waveforms that are of high quality, and that are natural-sounding. In addition, WaveNet can be used to generate audio waveforms that are realistic-sounding, and that match the characteristics of the training data. WaveNet has also been used to improve the quality of automatic speech recognition, and to generate realistic-sounding speech synthesis [38, 85].

While the original architecture and main applications have been in generating audio, the architecture has been adapted to other problem areas, although not with overwhelming success.

Transformer models

Transformer models are a type of neural network that is designed to handle sequential data. This means that it can take in a series of data points, such as a sequence of words, and learn to predict the next data point in the sequence. Transformer models are often used for tasks such as language translation and image captioning.

One of the key features of transformer models is that they use self-attention. This means that they can learn to attend to different parts of the input sequence simultaneously. This allows them to capture long-range dependencies in the data, which is difficult for other types of neural networks.

Transformer models have been shown to be very successful on a variety of tasks. They are often used in conjunction with other types of neural networks, such as convolutional neural networks, to provide a powerful model that can learn from data in a variety of ways [38, 85].

Large-scale language models

Large-scale language models are neural networks that are trained on a large corpus of text in order to learn the syntactic and semantic properties of language. These models are used in a variety of tasks, such as machine translation, question answering, and text generation.

Large-scale language models have a number of advantages over traditional statistical models. First, they are able to capture the long-range dependencies in language, which are difficult to model with traditional methods. Second, they can be trained on much larger corpora, which allows them to learn more about the statistical properties of language. Finally, they can be used for a variety of tasks, including machine translation, question answering, and text generation.

Despite these advantages, large-scale language models have a number of disadvantages. First, they are much more expensive to train than traditional statistical models. Second, they require a large amount of data in order to learn the syntactic and semantic properties of language. Third, they are often difficult to interpret, due to the complex nature of neural networks.

Despite these disadvantages, large-scale language models have become increasingly popular in recent years, as they have shown to be successful in a number of tasks. As the cost of training these models decreases and the amount of data available increases, it is likely that they will become even more popular in the future [38, 85].

Graph Neural Networks

A GNN is a neural network that can learn to recognize patterns in graphs. Graphs are a way of representing data that can be very useful for analyzing relationships between data points. GNNs can be used to find patterns in data sets that are too large to be processed by traditional methods.

GNNs are similar to other neural networks in that they have nodes that are connected by weights. The nodes in a GNN represent data points, and the weights represent the relationships between them. The nodes are arranged in layers, and each layer is connected to the next layer by a set of weights.

The GNN can learn to recognize patterns by training on a set of data. The training data is fed into the GNN, and the GNN adjusts the weights between the nodes to try to find patterns in the data. The GNN is then tested on a new set of data, and the results are used to improve the GNN's ability to find patterns.

GNNs have been used to find patterns in data sets that are too large to be processed by traditional methods. GNNs can also be used to find patterns in data sets that are not well-defined. For example, GNNs have been used to find patterns in social networks [38,85].

Three key types of graph neural networks include the Graph Convolutional Network (GCN) [51], the key type used in this thesis, GraphSage [41], and Graph Attention Networks (GAT) [95]. GCNs will be covered in detail below, but we will cover the other two next as the nearest alternatives to the approach taken in this thesis.

A Graph Attention Network (GAT) is a neural network that operates on graphs. It is an extension of the traditional convolutional neural network (CNN) that is designed to better handle the problems posed by irregular, non-Euclidean data such as graphs.

GATs are similar to CNNs in that they learn to map input data to a desired output through a series of layers. However, unlike CNNs, which operate on regular grids of data, GATs can operate on any kind of graph. This makes them well-suited for tasks such as link prediction and node classification, which require the ability to learn from data that is not necessarily structured in a regular way.

GATs are composed of two main types of layers: an attention layer and a graph convolution layer. The attention layer is responsible for learning the importance of each node in the graph, while the graph convolution layer is responsible for propagating information through the graph.

GraphSage is a neural network architecture for inductive learning on graph-structured data. It is based on the idea of learning continuous latent representations for nodes in a graph, and then using these representations to generate predictions for new nodes. GraphSage is particularly well-suited for learning on large-scale graph-structured data, as it can be efficiently trained on graphs with millions of nodes and edges.

Having now discussed some key types of neural networks, we will go on to discuss the key type of neural network architecture, we use in this thesis: the convolutional graph neural network.

2.3.1.4 Convolutional graph neural networks

Convolutional graph neural networks (CGNNs) draw on the popularity and success of traditional Convolutional Neural Networks's (CNN) in particular in computer vision. There are, however, different ways of defining convolution, in this case understood as a method of transformation based on a filter applied to a matrix representation of input data, when it is applied to graphs, which gives rise to different types of CGNNs. The most common approach bases itself on digital signal processing where convolution is seen effectively as a noise removal operation.

The difference between most variants in this approach including the seminal GCN architecture by Kipf and Welling [51], ChebNet [72], and CaleyNet [58] consists mainly in how they represent, approximate, and simplify the filter operations used in the convolution of the graph to achieve computational improvements.

The second main approach to CGNNs stays closer to the conventional CNN definition by considering convolution based on a node's spatial relationships [103]. That means spatial based methods in some way aggregates information from a node's neighbourhood. This can be seen for instance in the Message Passing Neural Network [84] that explicitly defines a framework for looking at graph convolution as a message-passing process.

The most relevant model for this thesis is the Graph Convolutional Network created by Kipf and Welling [51]. This model seeks to learn a function on a graph given a set of node features and a representation of graph structure.

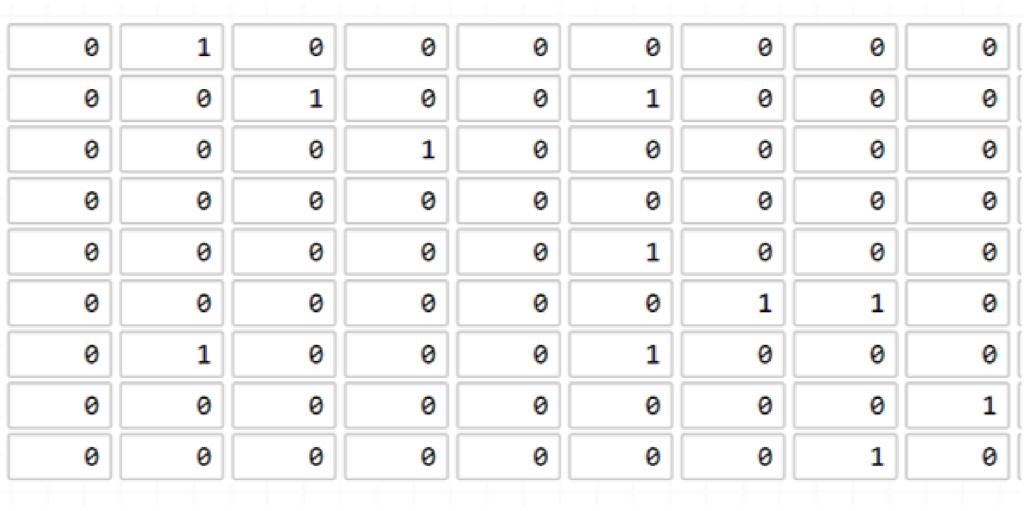
In several areas, particularly computer vision, transformer based architectures [43] have become dominant during the creation period of this thesis. There have also been research into graph transformers [104] that has achieved a level of success. Why then have we not considered this architecture in our work?

First, the success of the transformer architecture is much less marked in graph applications than it is for some other areas (e.g. computer vision). Second, the flexibility of the basic GCN architecture means that it is easily adaptable to special cases such as ours for argumentation. This easy adaptability to particular use cases is no doubt why it is still commonly used in practical applications, despite the availability of other architectures.

In addition, the basic GCN model works well in cases where graph structure rather than node features is the predominant carrier of information, which is why we prefer it to more elaborate models within the same family such as Graph Attention Networks (GAT) [95] or GraphSage [42].

We cover the key areas of the GCN architecture in the following definitions.

Definition 2.3.1 (Adjacency Matrix) *An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. In other words, the matrix has a row and column for every* *vertex in the graph, and the entry in row i and column j is* 1 *if there is an edge from vertex i to vertex j, and* 0 *otherwise.*



0	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	1	0
0	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0

Figure 2.5: Example of adjacencey matrix based on Figure 2.1

Definition 2.3.2 (Graph Convolutional Network) *A Graph Convolutional Network is a model that learns a function f on a graph $G(V, E)$, using inputs X, a matrix representation of node features, and A, the adjacency matrix of G.*

This function produces a node-level output with an arbitrary number of output features. This can be represented in line with other neural networks as a non-linear function.

Definition 2.3.3 (Layer-wise propagation) *Each layer in a GCN can be written as a non-linear function $H^{l+1} = f(H^{(l)}, A)$, where $H^{(0)} = X$ and the output of the final layer is the output of the GCN.*

Each layer of the GCN follows a propagation rule that maps an input representation to an output representation following a given rule. The propagation rule used by GCN is as follows.

Definition 2.3.4 (GCN propagation rule)
$f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$
$\hat{A} = A + I$, *where I is the identity matrix*
$\hat{D} =$ *Diagonal node degree matrix of $\hat{A}$*
$W^{(l)} =$ *The weight matrix for layer l*
$\sigma =$ *Any non-linear activation function*

This propagation rule uses two key tricks to improve on a naive update rule that would simply multiply the adjacency matrix with the weights and layer-wise representations. First, self-loops are added by adding the identity matrix to the adjacency matrix. This ensures that a node's own information can propagate to itself. Without this, the node would receive only information from its neighbours. Second, the adjacency matrix is normalized to avoid changing the scale of the feature vectors. This helps ensure numerical stability. Overall, the GCN model provides a remarkably simple, flexible model that has proven effective in many practical applications.

There has been several previous papers that has applied GCN methods to abstract argumentation [24, 52] with limited success. We shall consider these approaches, below. Graph Neural Networks have also been used more successfully in the related fields of Automated Theorem Proving [98] and the Graph Colouring Problem [57].

2.4 Graph embeddings

A graph embedding is a mathematical mapping of the vertices of a graph into a Euclidean space, such that the distances between the vertices in the graph correspond in some way to the distances between the points in the Euclidean space. This can be thought of as a way of representing the structure of a graph in a way that is amenable to mathematical analysis.

Graph Embeddings are used in a range of graph analysis applications including node classification, link prediction, clustering, and visualisation either directly or as additional input features to machine learning algorithms. There are several different algorithmic approaches to generating graph embeddings that are suited to different use cases. Goyal and Ferrara define three main approaches in their 2018 survey of the field [40].

The first category is factorisation-based approaches that share the property of working with a matrix representation (e.g. adjacency or Laplacian matrix) and a proximity measure to calculate the node embeddings. The next category, including the prominent Deepwalk algorithm, is based on random walks through the graph to generate the embedding. The last major category of approaches is based on Deep Neural Networks. For instance, Graph Convolutional Networks (GCN) generate an embedding by iteratively aggregating neighbourhood embeddings [51] based on graph structure and arbitrary features. In this thesis, we will use HOPE [76], a factorisation based approach, and a GCN based neural network approach to obtain the embeddings that we will visualise for argumentation graphs.

2.4.1 Relevant work in other areas

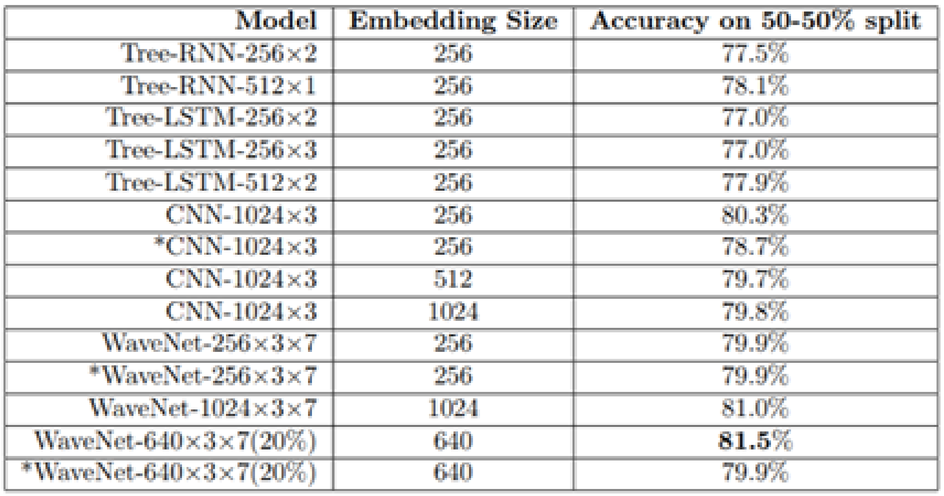
It is no secret that in recent years DNN-based methods have become dominant in a range of different AI subfields [25]. However, there has not so far been much interest from the argumentation community. This is somewhat surprising, given the fact that related subfields whose problems share many of the general characteristics of abstract argumentation have seen increasing amounts of research in this direction [11, 26, 59, 98].

As noted, an argumentation framework can be thought of as simultaneously a logical structure and a directed graph. This points to some obvious places to start looking for potential similar applications that may provide useful starting points for further research. In the following, we will review some of the relevant literature from related work applying DNNs to Automated Reasoning tasks and the solution of logic problems on the one hand and in a parallel manner we will equally review relevant work applying DNN methods to problems with graph structure.

The last few years have seen an increased interest in applying deep learning approaches to traditional problems in automated reasoning. The two main areas of exploration that are particularly relevant to the solution of argumentation problems are automated theorem proving, because it shares features of the dual graph and logical structure that also characterise argumentation frameworks. That is to say that both of these problems can be represented either in graph form or in a logical form. The other area is boolean satisfiability as this is the most common target for encoding/reduction of problems in argumentation frameworks [11, 59].

2.4.1.1 Automated reasoning

The current wave of research owes much to the release of the Holstep dataset [48], a dataset containing millions of intermediate steps in logical proofs as well as whether they were useful in the final found proof of some relevant theorem. By applying standard neural network architectures including tree based RNNs and LSTMs, classic CNNs, and WaveNet variants researchers from Google [59] achieved between 77.5% and 81.5% accuracy on the task of determining whether a given premise would be relevant to the proof of a given theorem.



Model	Embedding Size	Accuracy on 50-50% split
Tree-RNN-256×2	256	77.5%
Tree-RNN-512×1	256	78.1%
Tree-LSTM-256×2	256	77.0%
Tree-LSTM-256×3	256	77.0%
Tree-LSTM-512×2	256	77.9%
CNN-1024×3	256	80.3%
*CNN-1024×3	256	78.7%
CNN-1024×3	512	79.7%
CNN-1024×3	1024	79.8%
WaveNet-256×3×7	256	79.9%
*WaveNet-256×3×7	256	79.9%
WaveNet-1024×3×7	1024	81.0%
WaveNet-640×3×7(20%)	640	**81.5%**
*WaveNet-640×3×7(20%)	640	79.9%

Figure 2.6: Results of Various DNN Architectures on Premise Selection Task, reproduced from [59]

This premise selection task is one of the standards for determining performance of automated theorem proving algorithms and at the time set a new state-of-the-art:

"We have demonstrated the feasibility of guiding first-order logic proof search by augmenting the given clause selection method with ranking by deep neural networks. With a properly engineered mixture of neural guidance and hand-crafted search strategies, we get significant improvements in first-order logic prover performance, especially for theorems that are harder and require deeper search." [59, p. 14]

This performance, however, was subsequently improved by a team of researchers using a deep order preserving graph embedding that trained a set of functions to take into account the positioning of elements in a graph representation of proof steps [49]. This approach allows maintaining information from a node's immediate neighbourhood while simultaneously taking into account the ordering of elements in the logical statement under consideration.

The authors were able to achieve a 90.3% accuracy rate in the conditional premise selection task and 90.0% in the unconditional premise selection task setting a new state-of-the-art.

Reinforcement Learning (RL) approaches have also been applied successfully to automated theorem proving. Reinforcement learning is a type of machine learning where agents are trained to maximize a Reward by performing specific actions in an environment. The agent receives feedback in the form of positive or negative reinforcement after each action, which is used to modify the agent's behavior.

The following quote sets out the one of the most significant contribution in this area: "We have developed a theorem proving algorithm that uses practically no domain engineering and instead relies on Monte-Carlo simulations guided by reinforcement learning from previous proof searches. We have shown that when trained on a large corpus of general mathematical problems, the resulting system is more than 40% stronger than the baseline system in terms of solving nontrivial new problems. We believe that this is a landmark in the field of automated reasoning, demonstrating that building general problem solvers for mathematics, verification and hard sciences by reinforcement learning is a very viable approach." [49, p. 24]

In addition, there is work on using RL to teach agents a dialogue game for abstract argumentation by researchers at the University of York [1–3]. However, this work is not aimed specifically at the solution of argumentation frameworks according to specific semantics, but more at learning to argue generally.

2.4.1.2 Boolean Satisfiability

Boolean satisfiability also known by its acronym SAT is a key NP-complete problem related to abstract argumentation in so far as many solution approaches in abstract argumentation work by reducing the argumentation problem to a boolean satisfiability problem [101]. SAT is the problem of determining whether a given Boolean formula can be satisfied by a given assignment of truth values to its variables. A formula is satisfiable if it can be made true by some assignment of truth values to its variables. This is a surprisingly versatile tool for modelling a range of different problems.

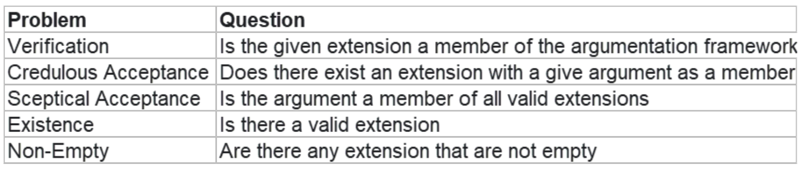
In the field of boolean satisfiability, there have been several reasonably successful attempts at using neural networks to train SAT solvers. In the paper "Learning a SAT solver from single-bit supervision" [88], researchers from Microsoft and Stanford successfully trained a simple DNN to solve small SAT problems using basic supervised techniques. Separately, a team from University College Cork also reported similar success rates by treating SAT solving as a supervised classification problem [26], work further extended using Graph Neural Networks by another team [11].

Another relevant approach, however, is the SATNet paper [100] that shows a method for implementing boolean satisfiability as a layer that can be embedded in a wider neural network architecture. That allows the SATNet layer to be integrated with the neural network training process directly and its outputs sent directly to further layers for processing.

2.5 Solution approaches for problems in abstract argumentation

2.5.1 Exact approaches

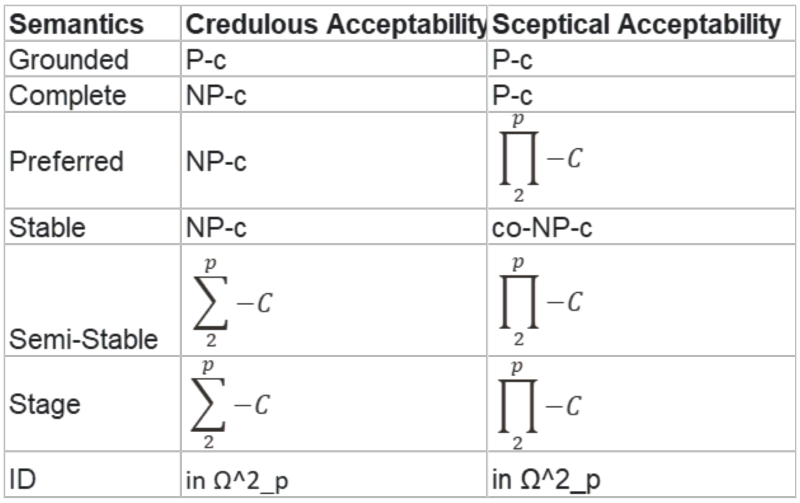
Abstract argumentation contains five decision problems [83], defined in figure 2.1.



Problem	Question
Verification	Is the given extension a member of the argumentation framework
Credulous Acceptance	Does there exist an extension with a give argument as a member
Sceptical Acceptance	Is the argument a member of all valid extensions
Existence	Is there a valid extension
Non-Empty	Are there any extension that are not empty

Table 2.1: Decision problems in argumentation frameworks. Reproduced from [83]

Many abstract argumentation decision problems are known to be intractable. Gaggl [36] gives an overview for sceptical and credulous acceptance, which can be found in figure 2.2.



Semantics	Credulous Acceptability	Sceptical Acceptability
Grounded	P-c	P-c
Complete	NP-c	P-c
Preferred	NP-c	$\prod_2^p -C$
Stable	NP-c	co-NP-c
Semi-Stable	$\sum_2^p -C$	$\prod_2^p -C$
Stage	$\sum_2^p -C$	$\prod_2^p -C$
ID	in Ω^2_p	in Ω^2_p

Table 2.2: Main complexity results for abstract argumentation reproduced from [36]

This state of affairs has led to a variety of approaches, none of which perform perfectly across benchmarks [36]. In fact, it has been demonstrated [6] that every abstract argumentation solver submitted to the 2017 International Competition on Computational Models of Argument (ICCMA) [94] is sensitive to the choice of the benchmark set of frameworks against which it is tested. For instance, it is possible to change the outcomes by manipulating the mix of graph topologies that the benchmark argumentation frameworks present. Nor is there any consistency in the ability of solvers to perform well across all semantics.

Broadly speaking, approaches to abstract argumentation systems can be grouped into those that utilise reduction to a separate formalism and those that deal with the argumentation framework directly [21]. Reduction based approaches utilise existing solvers for areas such as Boolean satisfiability (SAT) or Answer-Set Programming (ASP) and work by reducing decision problems in abstract argumentation to these forms. Direct approaches in contrast take a variety of approaches but share the property that they work directly on the argumentation graph defined by the argumentation framework. In the following paragraphs, I will cover the main variants of these approaches, starting with the reduction-based ones, including a brief discussion of their strengths and weaknesses.

2.5.1.1 SAT-Based Approaches

SAT-based approaches have dominated recent argumentation solver competitions (ICCMA 2017, 2019, 2021). This is to some extent not surprising given the long-standing work in the SAT-community to create highly optimised solvers and the existence of equally durable competitions for testing them against each other [47]. Charwat et al. [21] cover three approaches that SAT-based solvers have taken to reduce argumentation frameworks to a format suitable for feeding to a mature SAT-solver (e.g. Glucose).

The first is to reduce to unquantified propositional logic formulae. This is straightforward, but does not allow the encoding of all semantics. The second is to encode to Quantified Boolean Formulas (QBF). This allows the encoding of all semantics, but is considerably more complex. Finally, there is the iterative approach, which uses repeated calls to a SAT solver using simple propositional formulae to get the answers required for the semantics that cannot be encoded using such formulae. For instance, you can find all preferred extensions by traversing the complete search space for admissible sets or complete extensions. This last approach is used in practice by several leading argumentation solvers such as ArgSemSAT [16, 18] and CEGARTIX [33].

2.5.1.2 Answer-Set Programming

Answer-Set Programming (ASP) is another common target for argumentation solvers, the most prominent being ASPARTIX [34]. Encoding to ASP means providing a set of queries that given an input database and an answer set provides an answer that is in a one to one correspondence with the sought extensions. Charwat et al. [21] gives a set of basic rules to encode this procedure across a set of common semantics. More advanced encodings including saturation encodings and metasp encodings are covered by the same article and allow for the computation of all the common reasoning problems in argumentation. ASP based approaches have the benefit of being based on the well-known and understood domain of logic programming. However, in practice, ASP based approaches have not been able to compete with SAT-based ones in terms of efficiency.

Other reductions including to Constraint Programming or monadic second order logic also exist, but the SAT and ASP-based approaches are currently the dominant paradigms for solving argumentation frameworks efficiently using reduction.

2.5.1.3 Labelling-based approaches

The three main direct approaches are labelling-based algorithms [80], dialogue games [68], and dynamic programming [7]. They do not share much except that they work directly with the argumentation framework rather than reducing it. Direct approaches have not so far been able to compete with reduction based approaches in terms of efficiency, but one could argue that they are more likely to be the path to major advances as they allow the consideration of the unique characteristics of argumentation frameworks and argumentation related reasoning tasks without needing to worry about the target platform.

Algorithms that use a labelling based approach work by assigning a set of labels, most commonly the three values IN, OUT, and UNDEC, to the nodes in an argumentation framework and then using some strategy to propagate information across the graph in order to arrive at a labelling that is complete and correct. There are many variants of labelling algorithms in the literature that mostly fall into two categories: those that seek to enumerate all extensions under a given semantic, which mainly use backtracking strategies with clever heuristics to prune the search tree, and those that seek to solve particular reasoning problems (e.g. is argument X acceptable under semantic Y) using a strategy tailored for the specific reasoning problem [80].

2.5.1.4 Dialogue Games

Dialogue games instead model argumentation as as process of answer and reply and use that to reach a conclusion about the status of arguments under consideration in a given reasoning problem. The dialogue game is played by two players a proponent and an opponent of an argument who take turns attacking or defending the argument under consideration until a conclusion is reached. The specific rules of the game (e.g. whether players can repeat moves) lead to solutions that conform to different semantics. Such dialogue games can be represented by the resulting game tree of the permissible moves by each player at each turn and the implementation of dialogue game solver generally involve a search in the space of this game tree various methods [68].

2.5.1.5 Dynamic Programming

The last approach to consider is the dynamic programming approach. In this approach a tree decomposition of the given argumentation framework is traversed bottom-up generating partial solutions to the overall framework. As the algorithm works it way up it generates partial solutions that then using an appropriate data structure (e.g. a set of valid colourings for every node) can be combined into a general solution [7].

2.5.2 Approximate approaches

While the majority of the literature on solving abstract argumentation problems relates to exact approaches, there have been some other work that uses an approximate approach in particular in the last few years.

2.5.2.1 Stochastic Search

First, Thimm developed a local stochastic search algorithm inspired by WalkSAT, to find stable extensions [90, 91]. The algorithm searches for a single stable extension of a given input argumentation framework using a stochastic walk over arguments in the framework that randomly flips assignments based either pure chance or a greedy heuristic. This algorithm may obtain an answer or may terminate without having found one after reaching a maximum runtime in which case there may or may not be an answer.

2.5.2.2 Grounded Reasoning

Thimm has also provided a second approximate solver in the form of Harper++ [92]. This solver uses grounded reasoning exclusively to make approximations incorporating the set of arguments not attacked by the grounded extension for credulous acceptance, the problem of determining if an argument belongs to any valid extension. This is done on the basis that the grounded extension has been found empirically to be a good approximator for a range of abstract argumentation tasks in previous research [19].

2.5.2.3 Deep Learning

There have been previous papers that have applied deep learning methods to abstract argumentation [24, 52]. Kuhlmann and Thimm conducted a feasibility study by applying Kipf's approach to a range of datasets but using random and real world argumentation frameworks with some success.

They implemented a GCN model based closely on Kipf and Welling [51] recasting acceptability as a classification problem. The output of the network represented an estimate of the acceptability of the argument.

They, then, evaluated the GCN against synthetically generated argumentation frameworks and a sample taken from the ICCMA 2017 competition. Their best reported performance was 80.5% on one of the synthetic datasets. On their benchmark dataset, which closely resembles the one we will be considering in chapter 3, their overall accuracy was 63%.

Kuhlmann and Thimm, therefore, showed that a GCN approach was possible, but did not achieve the levels of accuracy that would have made it useful in practice. However, they pointed to an interesting direction of research that we are partially following in this thesis.

A second strand has been explored by Bex and Craandijk [24] who developed an Argumentation Graph Neural Network (AGNN) that learns a message passing function to approximate the acceptability status of arguments in an argumentation framework.

In this approach, an argumentation framework is mapped to a graph representation and each node is given its own node level embedding. These embeddings are then iteratively updated by exchanging messages with a node's neighbours. They show that their approach converges to near perfect accuracy on a set of relatively small test instances ($<= 200$ arguments).

The approach was not evaluated for larger argumentation frameworks, so it isn't directly comparable to the work in Chapter 3. However, it represents an interesting alternative graph-based approach, not based on a GCN.

2.6 Summary

This chapter has indicated the background material needed to understand the material covered in this thesis with a few minor exceptions, where short amounts of background material are used in one specific locaction in the thesis and it is therefore easier to present it immediately where it is used.

Throughout this chapter, we have focused on two main strands of material: one relating to computational argumentation, specifically abstract argumentation, and one relating to deep learning, specifically Convolutional Graph Neural Networks. We have covered these strands paying most attention to our specific areas, but also covering enough general background to allow the necessary understanding of the upcoming discussions in subsequent chapters. We, then, placed our work within the broader spectrum of solution approaches that are commonly applied to abstract argumentation problems. Having completed this task, we will now proceed to discuss the first and most significant set of contributions in this thesis as we embark on our discussion of how to approximate problems in abstract argumentation in the following chapter.

Chapter 3

Approximating Acceptability in Abstract Argumentation Frameworks with Graph Convolutional Networks

Acceptability in abstract argumentation concerns the question of whether a given argument is a member of certain extensions of an argumentation framework or not. It is found in a credulous version, which asks whether an argument is a member of any extension belonging to the argumentation framework and a sceptical version, which asks whether it belongs to all of them.

The decision problems associated with acceptability are all NP-hard with the single exception of acceptance under the grounded semantic, which can be computed polynomially [31]. As the grounded extension is unique, sceptical and credulous acceptance are identical in this case. The lack of a general polynomial time solution has led to an interest in approximate approaches for runtime critical applications as covered in the previous chapter.

In this chapter, we develop a new approximation approach based on a customised version of the GCN architecture developed by Kipf and Welling [51] as discussed in the background chapter, combined with a training approach tailored to abstract argumentation. We then conduct a substantial number of experiments to test the capabilities and limitations of this approach. This allows us to explore research questions 1-4 as defined in section 1.2.

During this chapter, we make the following contributions:

- We present the first systematic results for approximating abstract argumentation tasks across all the current ICCMA semantics.

- We set a new of state of the art for performance on the previously studied DC-PR and DS-PR abstract argumentation tasks. [52, 67]

- We propose an improved Graph Convolutional Network architecture and runtime implementation for this purpose.

- We demonstrate two different ways for grounded reasoning to be combined with a neural network model for the purposes of approximation.

- We provide a detailed analysis of approximation performance across the 11 benchmarks that formed part of ICCMA 19.

- We provide a detailed analysis of runtime performance of this approximation approach.

- We present a new scheme for evaluation for the approximation of abstract argumentation tasks.

By doing so, we advance the state-of-art for approximating abstract argumentation frameworks and make theoretical and empirical advances in understanding the limits and opportunities for approximation in this field.

We will start this chapter by reviewing the network architecture and the specific changes that enable us to achieve strong approximation performance on abstract argumentation tasks. We do this using two variants: a proof-of-concept version that was originally submitted to SAFA 2020 and a full version that was submitted to ICCMA 21. This allows us to understand the progression and design choices made for the final version better than would otherwise be possible. We go through the results for first the proof-of-concept (SAFA 2020) version and then more systematically for the final model.

Note that the following abbreviations are used extensively in this chapter:

DC-CO	The credulous acceptability problem under complete semantics.
DC-PR	The credulous acceptability problem under preferred semantics.
DC-ST	The credulous acceptability problem under stable semantics.
DC-SST	The credulous acceptability problem under semi-stable semantics.
DC-STG	The credulous acceptability problem under stage semantics.
DS-CO	The sceptical acceptability problem under grounded semantics.
DS-PR	The sceptical acceptability problem under preferred semantics.
DS-ST	The sceptical acceptability problem under stable semantics.
DS-SST	The sceptical acceptability problem under semi-stable semantics.
DS-STG	The sceptical acceptability problem under stage semantics.
DS-ID	The sceptical acceptability problem under ID semantics.

Table 3.1: Abbreviations of problems and semantics used extensively in chapter 3

3.1 AFGCN: an GCN-based approximate solver for abstract argumentation

3.1.1 Neural network architecture

The architecture used in this chapter builds on the seminal approach introduced by Kipf and Welling [51], but extends it in a number of areas. In the original formulation, the GCN consisted of an input layer, two hidden layers with RELU nonlinearities inserted in between, and ending with an output layer. Node embeddings were generated using a propagation rule following a first-order approximation of spectral graph convolutions.

We follow the same basic pattern, but add a number of features to allow for greater depth and to tailor the approach to abstract argumentation graphs that do not intrinsically have node-level features.

The core GCN architecture has been extended using deep residual connections between layers, input features based on the grounded extension, and a randomised training regime that shuffles both the frameworks to predict and what values within those frameworks on a continuous basis to improve generalisation.

We present two versions of the architecture in this chapter, one derived from the work presented at SAFA 2020 [67] and one from the AFGCN version that won 4 out of 6 tracks in the approximate track at ICCMA 2021. The SAFA 2020 can be viewed as a proof-of-concept model used to develop the initial approach and establish that the general approximation approach could work. It also served as a way to do experimentation with some of the critical parameters such as depth and batch randomisation that remained fixed for the AFGCN version. Therefore, the information presented relative to SAFA 2020 enhances the presentation of the final AFGCN model and has been included.

The core components of the GCN architecture used includes the following elements:

1. AFGCN has randomised input features combined with input features generated from the grounded extension of the argumentation framework. In the SAFA 2020 version, input features were generated by using DeepWalk [78].

2. An input layer receiving these inputs.

3. The AFGCN version has 4 repeating blocks of a GCN layer [51] and a Dropout layer [89]. The SAFA 2020 version experimented with layer depths of 4 to 6.

4. Residual connections feeding the original features and the normalised adjacency matrix as additional input at each block.

5. A Sigmoid output layer generating an estimate for the acceptability of each arguments on a continuous [0..1] scale.

The model was trained using Adam [50] with Binary Cross-Entropy as the loss function. The learning rate was set to $1e^{-3}$ for two hours and then dropped to $1e^{-6}$ for an additional six hours of training. These rates were identified by manual inspection of the training process. Details on the training regime are described in subsequent sections. All hyperparameters were manually optimised, using a process of starting learning observing results and adjusting.

3.1.1.1 Deep residual connections

The original formulation of Graph Convolutional Networks suffers from major performance degradation with an increase of depth beyond a certain limit. Kipf and Wellings' [51] original GCN, for instance, used only 2-layers in the model. In practice, as the depth of the GCN increases beyond this limit the model stops responding to training data and instead converges to a fix-point. This problem is known as the suspended animation problem [105] and the limit as the suspended animation limit.

Several approaches have been applied to overcome this limit and allow greater depth in GCN architectures. Among the most fruitful approaches have been those that adapt the notion of residual connections to the GCN context [105] by feeding in the graph structure and node features across layers in a variety of ways.

In this chapter, we follow a similar approach by adapting the graph-raw residual defined by Zhang and Meng [105]. They define the residual term as the multiplication of the normalised adjacency matrix and the raw input features. This residual term is fed as input to each layer in the model, which achieves the aim of extending the suspended animation limit.

The only difference in our approach is that the normalised adjacency matrix and raw input features are fed to each layer separately rather than as a unit, largely for reasons related to the implementation approach.

Definition 3.1.1 (Deep residuals) *By deep residuals, we mean layer-wise terms, R, that are added to the hidden state at each layer according to the following equation:*
$R(H^{(l-1)}, X; G) = \hat{A}X$

3.1.2 Input features

The input features can be divided into standard input features and the extended features generated by grounded reasoning. An overview of how the input features are combined can be found in figure 3.2.

3.1.2.1 Standard input features

The first important input feature is the adjacency matrix of the argumentation graph. This is generated dynamically from the input file and preprocessed in accordance with the normalization in definition 2.3.4. That is to say the identity matrix is added to the adjacency matrix.

To preserve the directionality of the attack relationships, only incoming links are included in the adjacency matrix. These are normalised using the function $\hat{A}\hat{D}^{-1}$. The normalisation is done dynamically on initialization of the training function.

For the SAFA version a DeepWalk graph embedding of dimensionality 64 was generated by running a preprocessing script on the input dataset using the standard implementation of DeepWalk. This was then included as node features in the feature vector, X.

In the AFGCN version, these were replaced by 64 random features initialized using Xavier initialization [37].

3.1.2.2 Grounded reasoning as an input feature

We incorporate grounded reasoning into AFGCN using input features that correspond to the binary status of whether an argument is included in the grounded extension or not. This is done to leverage the polynomial time solution to the grounded extension as a starting point for further approximation to speed up training and accuracy.

This results in a vector of 1s and 0s with a length equal to the number of arguments in the argumentation framework. We incorporate these as standard node features in the GCN model by precalculating them with a grounded solver and use them along with standard other features and the normalised adjacency matrix as inputs to the GCN.

For training purposes, we have two modes: one that includes the elements of the grounded extension in a mask of elements to be predicted as explained under the training regime and one that does not. This is because, we have two different modes of incorporating grounded reasoning at runtime. In one mode, we incorporate values in training only and in the other, we use a grounded solver to give an answer directly when an element is in the grounded extension. In the second case, there would never be a need for predicting the acceptability status of the arguments in the grounded extension using the neural network and the elements are therefore excluded from training.

3.1.3 Training regime

The overall process for generating data and training the model for use in the solver can be seen in figure 4.3. In the following sections, we will cover the relevant details.

3.1.3.1 Randomised training batches

Real-world abstract argumentation frameworks tend to have a skewed distribution between acceptable and non-acceptable arguments both for credulous and sceptical acceptance. In particular, there tends to be a large preponderance of non-acceptable arguments. In the argumentation frameworks used for the experiments in this chapter, the percentage of non-acceptable arguments ranges from 69.5% to 99.95%.

This affects GCN training as the neural net will by default learn to predict a negative outcome even in cases where it is incorrect. This problem was also noted by Kuhlmann and Thimm [52], who generated balanced training data to attempt to address it, but this approach does not seem to have generalised well.

To overcome these limitations, we have devised a randomised training scheme that generates random training batches at the start of each training iteration. These are generated by sampling the total dataset and selecting n argumentation frameworks at random with uniform probability.

The overall training scheme feeds multiple argumentation frameworks to the neural network as a single graph with each argumentation framework represented by a single connected component. That enables effective batch processing of multiple argumentation frameworks that can be treated as a single graph for learning purposes. This processing happens dynamically in the training function.

To preserve the maximal amount of structural information the entire graph is fed to each network layer as a normalised adjacency matrix. The output layer of the neural network is a prediction of the acceptability of the argument.

The randomised training operates by generating a new mask of outcomes to be predicted at the start of each new epoch, essentially asking the neural net to fill in the blanks. This mask is a binary vector with a length equal to the size of all the graphs in the training batch. The value in the binary vector indicates whether the prediction in the given spot is included in the loss calculation used for network learning.

This is done to encourage the network to learn to generalise based on structural properties of the graphs. By continuously randomising both the composition of what graphs are predicted together and what arguments in those graphs are predicted, the ability of the neural network to generalise to unseen graphs improved dramatically. See figure 3.4 for an overview of batch generation.

3.1.3.2 Dynamic balancing and outlier exclusion

Two additional measures were taken to address the problems related to imbalanced training data and poor generalisation performance. First, the training mask created to facilitate the generalisation of the neural network was developed to have the option of dynamically balancing the training mask to include equal amounts of acceptable and non-acceptable arguments.

That is accomplished by programmatically adjusting the mask during training so the target vector contains similar amounts of acceptable and non-acceptable arguments. The algorithm to implement this simply replaces a given number of arguments of one class with arguments of another class in order to achieve the balance.

This has the intention to avoid the skew caused by unbalanced training data, but also has the unfortunate side effect of reducing the amount of data used for training. This mode is therefore not used in all experiments described in the results section.

A second optimisation was added to handle extremely skewed argumentation frameworks. Some frameworks have no or almost no acceptable arguments and when included tend to skew the training disproportionately. These frameworks have been excluded from the training set using a z-score test with a threshold of 3.5 [46]. See figure 3.5 for an illlustration.

3.1.4 Runtime implementation

The model chosen for the final solver runtime is a 4-layer model with 128 features per layer. This was selected based on the experimentation done for the original SAFA 2020 model and the need for a fast model to optimize time for the competition. The solver has been built using the Python programming language, utilising the Pytorch framework for training and modelling [77], the Deep Graph Library for graph representation [99], and Numpy [44] for numerical computation.

At runtime the solver is called using a shell script wrapper that conforms to the specifications of ICCMA 2021. This shell script calls a Python script that loads the relevant parameters into the GCN model based on the semantic in question. It then pre-computes the grounded extension using a Numpy-based grounded solver and passes this information along with a random input feature to the GCN model for inference.

The output of the inference step is then passed to a threshold function, which applies a threshold for acceptance that is adapted to the size of the argumentation framework and the semantic under consideration. The solver approximates the acceptability status of all arguments in the argumentation framework in parallel during the inference step, using a single step of the GCN, but to conform with the ICCMA 2021 solver specification it only outputs the predicted status for the particular argument under consideration. See figure 3.6 for more detail on the runtime architecture.

3.2 Experimental results

3.2.1 SAFA 2020 results

3.2.1.1 Dataset and experimental setup

The experiments were run against a dataset consisting of 900 argumentation frameworks selected from the ICCMA 2017[1] Benchmark datasets. The selection includes frameworks from benchmark sets A, B, and C including all 5 difficulty categories. Except for the exclusion of a small number of very large argumentation frameworks that could not be processed in the computational environment used for the experiment because of memory limitations, no systematic exclusions were made. Input frameworks contained up to 15,605 arguments and 6,250,500 attacks with an average number of arguments of 1,595 and average number of attacks of 187,542. We divided the dataset into 9 folds to facilitate the training process and the neural net was trained sequentially on each fold. Training and validation sets were generated using a random mask as described above and 10% of graphs were held out as a test set for final evaluation.

The GCN model has been evaluated using two separate tasks: credulous acceptability and sceptical acceptability under the preferred semantic. For an argument to be credulously acceptable in the context of a given argumentation framework it must belong to one of the extensions of that argumentation framework; to be sceptically acceptable, it must belong to all. To be able to frame the problem for supervised learning, we generated ground truth solutions using the Pyglaf argumentation solver [4] for all argumentation frameworks in the dataset.

The experiments were run on a single K80 GPU instance with 12GB ram. Each model took between 4 and 12 hours to train. Separate models were trained for each experiment listed in the results below. Table 3.2 details what elements were used for each model.

The headings have the following meanings:

- **Layers**. How many GCN layers were included in the model.
- **Balanced**. Whether the training mask was actively balanced.
- **Randomised**. Whether the random training mask was used.
- **Residuals**. Whether deep residuals were added.

[1]http://argumentationcompetition.org/2017/

Table 3.2: Models trained for the SAFA 2020 experiments broken down by key features.

Model	**Layers**	**Balanced**	**Randomised**	**Residuals**
4-Layers Modified GCN	4	No	Yes	Yes
5-Layers Modified GCN	5	No	Yes	Yes
6-Layers Modified GCN	6	No	Yes	Yes
Mod GCN with Balanced Data	4	Yes	Yes	Yes
Mod GCN with Fixed Batches	4	No	No	Yes

3.2.1.2 Credulous acceptability results

The results for credulous acceptability improve by 30 percentage points on the previous baseline set by Kuhlmann and Thimm. While the best performing model in terms of overall accuracy is the 4-Layer Modified GCN, the better performance on positive acceptability coupled with the very slight decrease in overall accuracy means that the 5-Layer Modified GCN is the overall best performing model.

Excepting the Modified GCN with Balanced Data, all the other models have a disparity in favour of negative acceptability, which is consistent with the previous findings by Kuhlmann and Thimm. The Modified GCN with Balanced Data inverts the pattern with a similar disparity between positive and negative in favour of the positive side, which demonstrates that there is nothing inherent about negative acceptability that makes it easier to learn. If anything, the opposite would seem to be the case as the Modified GCN with Balanced Data will see roughly equal amounts of positive and negative training instances. The poorer overall performance of this model could be due to the lower amounts of training instances seen because of dynamic balancing.

The extremely poor performance when predicting positive acceptability of the Modified GCN with Fixed Batches demonstrates the key contribution of the randomised training regime in obtaining good results in this domain.

3.2.1.3 Sceptical acceptability results

The results for Sceptical Acceptability seem to reflect the much starker imbalance between negative and positive instances in this setting. By the nature of the problem there will tend to be much fewer positive instances in the sceptical setting. This is reflected in all the models having a large discrepancy between positive and negative accuracy.

Table 3.3: Approximation results for the SAFA 2020 model on the credulous acceptability problem for DC-PR

Model	**Accuracy**		
	Overall	**Yes**	**No**
4-Layers Modified GCN	92,68%	69,33%	93,54%
5-Layers Modified GCN	92,26%	73,56%	92,95%
6-Layers Modified GCN	91,63%	71,81%	92,37%
Modified GCN with Balanced Data	81,20%	91,20%	71,00%
Modified GCN with Fixed Batches	96,40%	7,00%	99,70%
Kuhlmann and Thimm 2019 - Balanced	63,00%	17,00%	93,00%
Kuhlmann and Thimm 2019 - Unbalanced	62,00%	10,00%	97,00%

Similar to the credulous results, the Modified GCN with Fixed Batches is the worst performer. It is, however, even more extreme for the sceptical case, possibly for the same reason as before. For all intents and purposes the model is incapable of correctly predicting positive acceptability and only gets good overall accuracy from the extreme skew of the underlying dataset.

In contrast to the credulous setting, the Modified GCN with Balanced Data is the clear overall winner under the sceptical setting. It has both the best overall accuracy and is vastly superior in predicting positive acceptability with a relatively minor hit to negative accuracy. Note that there is no baseline in this case as this was the first work carried out to approximate this dataset.

3.2.1.4 Ablation studies

Effects of depth For both credulous and sceptical acceptability, three separate models were trained differing only in the number of layers they possess[2]. Based on these results the effect of depth is inconclusive, but at best minor. Under the credulous setting the 5-Layer Modified GCN outperforms the other two slightly. Under the sceptical setting the results are not different enough to assess which model is superior.

The reasons why greater depth does not seem to lead to better performance are unclear

[2]Note that the tables for the ablation studies do not contain additional results, but are constructed based on the previous models.

Table 3.4: Approximation results for the SAFA 2020 model on the credulous acceptability problem for DS-PR

Model	**Accuracy**		
	Overall	**Yes**	**No**
4-Layers Modified GCN	96,21%	24,04%	97,10%
5-Layers Modified GCN	96,20%	22,92%	97,11%
6-Layers Modified GCN	96,24%	22,69%	97,15%
Modified GCN with Balanced Data	97,15%	46,35%	94,39%
Modified GCN with Fixed Batches	98,44%	0,33%	99,66%

at this stage and will require further research. Four layers may already be adequate to capture what can be learned from the training data. It may also be related to the well known issues experienced by other GCN models that have tried to increase depth. However, unlike some of these models, we are not seeing a decline in performance with the addition of layers, but a more or less stationary set of results.

Table 3.5: Approximation results for the SAFA 2020 model on the acceptability problem ordered by network depth

Model	**Accuracy**		
	Overall	**Yes**	**No**
4-Layers Modified GCN (Credulous)	92,68%	69,33%	93,54%
5-Layers Modified GCN (Credulous)	92,26%	73,56%	92,95%
6-Layers Modified GCN (Credulous)	91,63%	71,81%	92,37%
4-Layers Modified GCN (Sceptical)	96,21%	24,04%	97,10%
5-Layers Modified GCN (Sceptical)	96,20%	22,92%	97,11%
6-Layers Modified GCN (Sceptical)	96,24%	22,69%	97,15%

Effects of randomisation and balancing The largest and most substantial difference in this study is between the models that include the randomised training regime and those that do not, including both models from Kuhlmann and Thimm and the Modified GCN with Fixed Batches for both the credulous and the sceptical case. That points towards this training regime being the largest contributor to increased model performance.

The effects of balancing are more mixed, but also substantial. Kuhlmann and Thimm found a small improvement from using a balanced dataset. In our model under the credulous setting, the Model GCN with Balanced Data performed substantially worse than the other models that also use the randomised training regime. However, for the sceptical setting it was hands-down the best performing model. This may be due to a relationship with the extent to which the underlying training data is skewed. One could speculate that the more skewed the training data, the greater the value of dynamic balancing.

Table 3.6: Approximation results for the SAFA 2020 model on the acceptability problem ordered by training regime

Model	**Accuracy**		
	Overall	**Yes**	**No**
Modified GCN with Balanced Data (Credulous)	81,20%	91,20%	71,00%
Modified GCN with Fixed Batches (Credulous)	96,40%	7,00%	99,70%
Modified GCN with Balanced Data (Sceptical)	97,15%	46,35%	94,39%
Modified GCN with Fixed Batches (Sceptical)	98,44%	0,33%	99,66%
Kuhlmann and Thimm 2019 - Unbalanced	62,00%	10,00%	97,00%
Kuhlmann and Thimm 2019 - Balanced	63,00%	17,00%	93,00%

3.2.2 AFGCN results

The AFGCN solver, both in the version submitted for ICCMA21 and in the version further developed for this chapter, builds on and systematises the results from the SAFA 2020 version. We will start by presenting the experimental setup for the chapter and then review the results first by semantics and then in a cross-cutting way. An overview of the AFGCN network architecture can be found in

3.2.2.1 Dataset and experimental setup

We train our models on a dataset of 792 graphs taken from past ICCMA competitions. The ICCMA competitions provide a comprehensive set of benchmark problems that provide good comparability to historical results and therefore provides a good source for dataset creation. The graphs range in size from 2 to 100,000 arguments. The test set consists of 99 graphs constructed to contain a fairly even split of graphs between the benchmarks present at the ICCMA 19 competition. Table 3.7 contains a description of the benchmarks under consideration. More information can be found by consulting the relevant benchmark descriptions [86].

Table 3.7: Description of benchmarks used for the AFGCN solver evaluation

Benchmark	Description
ABA2AF	Assumption-Based Argumentation translated to abstract argumentation frameworks
AFGen	Based on a general model for producing random digraphs with differing properties
Barabasi-Albert	Barabasi-Albert graphs, randomly generated
Erdős–Rényi (ER)	Erdős-Rényi graphs, randomly generated
Grounded	Randomly generated argumentation frameworks containing only a large grounded extension
Logic Based Argumentation (LBA)	Argumentation graphs based on knowledge bases
Planning2AF	Planning problems transformed to abstract argumentation problems
Stable	Graphs generated to have a high number of stable extensions
Traffic	Traffic networks converted to abstract argumentation frameworks
Watts-Strogatz	Watts-Strogatz graphs, randomly generated
admbuster	admbuster graphs, based on Caminada and Podlaszewski [13], designed to foil certain types of solvers

Table 3.8 contains the characteristics of the test set relative to the benchmarks defined above. Max, mean, and min refers to the number of arguments.

This division into benchmarks allows us to evaluate the relative performance of the solver on different graph structures. The size differentials also allows us to see whether there are differences in approximation performance related to size. Both of these help us answer our fourth research question as defined in section 1.2.

We trained three different models for comparison using a variety of features described in the previous sections. In addition, we also used a deterministic grounded solver as a

Table 3.8: Characteristics of the test set used for AFGCN evaluation

Benchmark	Number	Max	Mean	Min
ABA2AF	10.0	848.0	611.7	443.0
AFGen	10.0	320.0	189.6	100.0
Barabasi-Albert	10.0	201.0	111.0	21.0
Erdős–Rényi	3.0	102.0	101.7	101.0
Grounded	10.0	8020.0	3942.7	1697.0
LBA	10.0	103.0	58.0	6.0
Planning2AF	10.0	1992.0	627.4	86.0
Stable	10.0	767.0	562.7	265.0
Traffic	10.0	35.0	21.1	7.0
Watts-Strogatz	6.0	300.0	266.7	200.0
admbuster	10.0	10000.0	7000.0	4000.0

fourth option. For convenience, we also refer to this as a model in our results presentation, although strictly speaking it does not rely on any machine learning components. The division into these four models help us answer our research questions 2 and 3, evaluating the best architecture for the solver and the value of the grounded extension.

The four models are characterised below, the abbreviation after each model denotes how it is referred to in the results tables:

- **GR-ONLY (GR)**. This model uses only the deterministic grounded solver.
- **GCN-NO-GR (NO-GR)**. This model uses a 4-layer GCN model using the randomised training regime, input feature initialisation, thresholding, and residual connections, but no grounded features.
- **GCN-WITH-GR (W/GR)**. This model uses everything discussed under the GCN-NO-GR model, but also takes the grounded extension as an input feature both during training and inference.
- **HYBRID-GCN-GR (HYBR)**. This model uses everything included in the GCN-WITH-GR model. However, it does not train on elements of the grounded extension. Instead, it incorporates a grounded solver during the inference stage and always trusts a positive answer from that solver. For negative cases, it applies the neural network for inference as in the previously discussed model.

We evaluate our models in five different ways for all semantics under consideration. This allows us to see whether there are any systematic differences in approximation performance across semantics, which helps us answer research question 4 as defined in section 1.2. The evaluation settings are summarised in the following list:

- **Equally weighted**. The equally weighted setting weighs each argumentation framework equally regardless of its size. This is equivalent to the score one would expect when picking a single argument to classify from each of a number of different argumentation frameworks as for instance in the ICCMA competitions.
- **Complete balanced**. This setting classifies all arguments across all argumentation frameworks and gives weight according to the size of the framework. So performance on a 1,000 argument framework is weighted 10x as highly as on a 100 argument one. This setting is included as it is the one, which has been used in previous work on approximating argumentation frameworks [52, 67].
- **Reduced balanced**. This setting is equivalent to complete balanced, but excludes the benchmarks Grounded and admbuster. As can be seen from Table 3.8, these two benchmarks dominate in terms of size and also share the characteristic of being fully solvable using only grounded reasoning. Therefore, including these on an equal basis makes the results hard to interpret. The reduced balanced setting corrects for this problem.
- **By benchmark**. This setting compares performance across the benchmarks targetted for evaluation.
- **By size band**. This setting compares performance across 10 size bands that splits the test set into 10 roughly equal parts by number of arguments.

We report a number of different evaluation metrics for the sake of completeness, but rely mostly on accuracy, which has been the key metric in past research, and Matthews Correlation Coefficient (MCC), which gives the best view of an estimator's overall performance taking into consideration all classes and class imbalances.

These are defined as follows:

$TP = True\ Positive$

$FP = False\ Positive$

$TN = True\ Negative$

$FN = False\ Negative$

$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

$Precision = \frac{TP}{TP+FP}$

$Recall = \frac{TP}{TP+FN}$

$F1 = \frac{2*Precision*Recall}{Precision+Recall} = \frac{2*TP}{2*TP+FP+FN}$

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

The models were tuned using the hyperparameters for the original AFGCN model submitted to ICCMA 21. As the purpose here has been to evaluate systematically, we have not sought to wring out every last little bit of performance from the various models. We tuned the configurable thresholds on the complete training set, assigning an optimal set of thresholds based on the 10 size band used for evaluation for all models and semantics.

There is considerable variability across the different semantics that have been evaluated as part of this chapter. In the following sections, we will systematically go through the results for each of the six included semantics before proceeding to the additional cross-cutting analyses.

3.2.2.2 Results for credulous acceptance

Preferred semantics

We will first consider the results of running the experiment on credulous acceptability. We refer readers to results tables that can be found in the appendix for detailed presentation of the metrics. But for a simpler overview while reading the chapter, we include summary diagrams that show the headline results and allows for easier digestion of the trends.

Figure 3.8 shows the results of running our model against the test set for DC-PR.

Overall the best performing model under equal weighting is the one combining GCN and grounded reasoning in a hybrid mode, although the difference in performance to the GCN-WITH-GR model, incorporating grounded reasoning only through input features is minimal. For positive accuracy and precision the grounded reasoner achieves a perfect score, which is unsurprising considering that all preferred extension are also complete extensions and the grounded extension is a subset of any complete extension.

It is perhaps surprising that the HYBRID-GCN-GR model does not achieve a higher boost in positive accuracy by applying grounded reasoning. While the configurable threshold helps performance here, it may not be sufficient to compensate for the tendency towards false positives exhibited by all the neural network models. However, all of the performance boost seen relative to the GCN-WITHOUT-GR model that doesn't incorporate grounded reasoning does come from an increase in positive accuracy as the simplified GCN model actually performs slightly better on negative accuracy.

The F1 and MCC scores both indicate that all the GCN models are strong positive predictors for credulous acceptability under the preferred semantics. The GR-ONLY model exhibits only a moderate positive relationship as a predictor in comparison.

Moving on to the results for the complete balanced setting, we can see that all models have very strong performance, which is attributable to the dominance of grounded reasoning in this evaluation setting. Unsurprisingly, the best performing model both in terms of accuracy and MCC is HYBRID-GCN-GR.

On the other hand in the reduced balanced setting, excluding the two large grounded-focused benchmarks, we see marginally better performance from the GC-WITH-GR model, largely attributable to this model having slightly better performing thresholds for targeting benchmarks that do not rely exclusively on grounded reasoning. It is also worth noting that there is poor recall performance of the GCN-NO-GR model in this evaluation setting, which indicates that the threshold configuration has not been enough on its own to achieve a low rate of false negatives.

If we turn to the benchmark results in Table 3.9, there is a marked difference in performance between benchmark types. All models achieve comparable results on the ABA2AF framework, but the model without grounded features achieved a much lower MCC indicating a weaker balance in the results. For the AFGen benchmark, no model is strong enough to have real predictive power according to MCC, although the grounded GCN variants perform slightly better.

The three GCN models have equivalent correlation scores for BA graphs, but the GCN-NO-GR model has highest accuracy indicating that for this graph type, grounded reasoning is not an important factor. ER graphs exhibit identical accuracy and for all three models that apply grounded reasoning, outperforming the GCN-NO-GR model.

This is an interesting phenomenon worthy of a separate analysis, but outside the scope of the current chapter. As expected the GR-ONLY model performs best on the Grounded benchmark with the GCN-NO-GR model performing the worst. This also shows the hybrid approach to work better as a way of incorporating grounded reasoning than just features on tasks that are slanted heavily towards this mode.

All the GCN models achieve perfect scores on the Logic Based Argumentation benchmark, outperforming the grounded reasoner.The fairly well-structured information derived from knowledge base information would seem to be a good fit for approximation with GCNs.

On the Planning2AF benchmark, the hybrid model achieves slightly higher accuracy than and equal MCC to the GCN-WITH-GR model, outperforming the other two, again showing the value of combining GCNs with grounded reasoning. The hybrid model also outperforms slightly on the Stable benchmark, although none of the models do particularly well.

Traffic network data is another area, where grounded reasoning does not seem to play a part and in fact seems harmful, given the much stronger performance of the GCN-NO-GR model. Watts-Strogatz graphs as with AFGen previously seems basically unapproximable using either a GCN or grounded reasoning approach. Finally, we can note that none of the models are fooled by the admbuster benchmark.

Finally, we can consider the breakdown by size band as shown in Table 3.10. Performance of the two upper size bands, 8 and 9, can be explained simply by the dominance of Grounded and admbuster benchmarks in those bands. The same explain the overall performance of the GR-only solver as that only does very well on these benchmarks and hence only logs good performance scores at the top end of the table. It is harder to see clear size based patterns for the other bands, excepting that the two feature based GCN models, GCN-WITH-GR and GCN-NO-GR, seem to do better at the low and high ends, while the hybrid model has slightly more consistent performance across sizes.

Table 3.9: Overview of AFGCN approximation results for DC-PR ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	98.53%	99.07%	98.97%	98.96%	0.53	0.73	0.71	0.7
AFGen	58.23%	55.36%	54.60%	58.16%	0.094	0.13	-0.1	0.11
Barabasi-Albert	91.55%	86.52%	49.93%	90.70%	0.72	0.71	0.31	0.73
Erdős–Rényi	63.03%	96.04%	96.04%	96.04%	0.32	0.67	0.67	0.67
Grounded	97.94%	98.48%	100.00%	98.98%	0.69	0.77	1.0	0.94
LBA	100.00%	100.00%	0.79%	100.00%	1.0	1.0	-0.4	1.0
Planning2AF	65.18%	74.91%	59.66%	76.45%	0.34	0.52	0.4	0.52
Stable	65.59%	67.24%	62.53%	68.03%	0.27	0.3	0.2	0.32
Traffic	81.40%	75.67%	32.95%	73.19%	0.51	0.31	0.029	0.27
Watts-Strogatz	75.97%	75.92%	75.25%	75.25%	0.1	0.16	0.0	0.0
admbuster	99.75%	99.57%	100.00%	100.00%	0.99	0.99	1.0	1.0

Table 3.10: Overview of AFGCN approximation results for DC-PR ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	80.45%	76.24%	36.29%	73.76%	0.47	0.41	0.14	0.32
1	93.96%	93.72%	34.75%	92.38%	0.8	0.71	0.0086	0.73
2	89.51%	89.72%	17.08%	90.42%	0.81	0.83	-0.21	0.85
3	62.37%	60.86%	67.65%	73.74%	0.26	0.23	0.34	0.41
4	61.30%	71.46%	65.56%	73.04%	0.026	0.22	0.13	0.21
5	68.91%	69.33%	70.33%	70.33%	0.17	0.3	0.16	0.16
6	83.46%	85.68%	69.92%	81.88%	0.54	0.63	0.32	0.52
7	93.49%	94.39%	89.87%	93.02%	0.58	0.73	0.67	0.68
8	98.92%	98.97%	100.00%	100.00%	0.82	0.86	1.0	1.0
9	99.03%	99.17%	100.00%	100.00%	0.88	0.91	1.0	1.0

Complete semantics

The results for the complete semantics have many similarities with those for the preferred semantics, which is unsurprising as all preferred extensions are also complete extensions. There are, however, a number of salient differences that we shall point out as we go through.

Starting with the equally weighted results in figure 3.9, we see the GCN-WITH-GR model edge ahead of the HYBRID-GCN-GR model on accuracy, while maintaining equal MCC. As expected for credulous reasoning the grounded reasoner does relatively poorly.

This changes, however, when we get to the complete balanced setting, dominated by the large Grounded benchmarks. As DS-CO is equal to the grounded extension, we would expect the GR-ONLY model to have near perfect performance in this case. Here it is matched by the HYBRID-GCN-GR model that even slightly outperforms it by having a better class balance between positive and negative cases.

Removing these large grounded-focused frameworks, the overall pattern seen in the equally weighted setting reasserts itself.

ABA2AF, AFGen, Grounded, Planning2AF, Stable and admbuster benchmarks have effectively the same behaviour seen in the preferred case albeit with some variation in performance. However, the other five benchmarks do not follow the same pattern. Barabasi-Albert, ER, and Traffic benchmarks all seem to benefit from a level of grounded reasoning that they did not under the preferred semantics. The LBA benchmark that was perfectly predictable by the GCN models under preferred semantics is only partially predictable under complete semantics and interestingly has a perfect negative correlation with grounded reasoning. Watts-Strogatz graphs are slightly more predictable by a GCN under complete than preferred semantics, although still only weakly so.

The size band patterns shown in Table 3.12 are very similar to the ones for preferred semantics, although the GCN-WITH-GR model has better performance in the middle bands, which will help explain its better performance overall.

Stable semantics

The results for stable semantics has much in common with the ones we have just seen for complete semantics, more so than it shares with preferred semantics, which is somewhat strange, considering that every stable extension is a preferred extension.

In the equally weighted setting, shown in figure 3.10 again the GCN-WITH-GR model

Table 3.11: Overview of AFGCN approximation results for DC-CO ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	98.74%	99.10%	99.03%	99.21%	0.64	0.77	0.77	0.82
AFGen	54.66%	65.93%	54.60%	62.43%	-0.099	0.068	-0.1	0.058
Barabasi-Albert	93.88%	95.61%	49.93%	95.91%	0.78	0.84	0.31	0.85
Erdős–Rényi	91.75%	88.80%	96.04%	81.90%	0.75	0.061	0.67	0.1
Grounded	97.95%	98.51%	100.00%	99.78%	0.69	0.79	1.0	0.98
LBA	67.76%	95.73%	0.00%	96.11%	0.3	0.3	-1.0	0.3
Planning2AF	72.48%	83.09%	59.66%	77.34%	0.36	0.64	0.4	0.54
Stable	67.49%	71.36%	62.04%	69.30%	0.34	0.43	0.19	0.4
Traffic	81.04%	89.79%	32.95%	85.59%	0.27	0.72	0.029	0.64
Watts-Strogatz	73.42%	76.64%	75.25%	78.14%	0.13	0.27	0.0	0.21
admbuster	99.38%	99.67%	100.00%	100.00%	0.99	0.99	1.0	1.0

Table 3.12: Overview of AFGCN approximation results for DC-CO ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	64.12%	89.75%	19.33%	87.50%	0.4	0.43	-0.49	0.48
1	87.87%	95.91%	30.69%	95.43%	0.44	0.8	-0.055	0.72
2	90.96%	92.86%	33.45%	93.52%	0.39	0.5	-0.12	0.5
3	66.31%	78.23%	62.44%	71.61%	0.25	0.28	0.19	0.25
4	68.41%	74.06%	65.56%	73.87%	0.16	0.27	0.13	0.22
5	70.48%	74.12%	70.38%	75.08%	0.14	0.33	0.15	0.39
6	87.09%	89.27%	81.96%	85.04%	0.49	0.64	0.49	0.56
7	87.27%	88.82%	83.19%	86.93%	0.71	0.74	0.66	0.72
8	98.67%	99.05%	100.00%	100.00%	0.81	0.87	1.0	1.0
9	98.85%	99.23%	100.00%	100.00%	0.88	0.92	1.0	1.0

is the overall best performing model. Interestingly, the HYBRID-GCN-GR model and the GCN-NO-GR model have near identical performance, which might indicate that there is less space for improving performance with grounded reasoning under stable semantics.

In the complete balanced setting, we see the same pattern as under complete semantics, where the HYBRID-GCN-GR model performs the best, due to the dominance of the large grounded frameworks.

However, when excluding these frameworks the trend reverses again. This is consistent with the other findings we have seen so far.

Again, it is in the benchmark specific performance, shown in Table 3.13, we find the most interesting variation due to the semantics. For AFGen, Barabasi-Albert, Watts-Strogatz, Grounded, admbuster, and Stable benchmarks, the pattern is the same as we saw under complete semantics. One might have expected that a model trained on stable extension would perform better on the Stable benchmark, but this is not reflected in the data. This would seem to indicate that the GCN has not learned any semantics specific representations for these semantics. The approximation of ER graphs is for some reason easier with the GCN-WITH-GR model under stable semantics than under complete semantics where this model completely failed. The GCN-NO-GR model is still the best performing model here, which was true for complete, but not preferred semantics. Approximation of LBA frameworks is somewhere in the middle between those of complete and preferred semantics. Planning2AF problems have slightly better results under stable semantics across the board than under complete or preferred semantics, excepting a slight drop for the GCN-WITH-GR model relative to complete semantics, the same is true for the Traffic benchmark.

The size specific evaluation, shown in Table 3.14 follows the pattern we have seen before for credulous acceptance with increases in performance at the low and high ends for the GCN models, although it is less pronounced for stable semantics.

Semi-stable semantics

We would intuitively expect the results for semi-stable semantics to most closely resemble those of the stable and preferred semantics. That intuition is not entirely borne out in practice, as the results for these semantics are quite distinctive.

Looking first at the equally weighted setting in figure 3.11, we note a much reduced performance for the GCN-NO-GR model. It would seem that semi-stable semantics presents

Table 3.13: Overview of AFGCN approximation results for DC-ST ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	98.97%	98.90%	98.53%	98.57%	0.049	0.12	0.07	0.085
AFGen	54.53%	54.83%	54.60%	59.70%	-0.087	0.0016	-0.1	0.045
Barabasi-Albert	94.24%	92.81%	49.93%	82.25%	0.78	0.77	0.31	0.61
Erdős–Rényi	95.38%	93.73%	96.04%	80.93%	0.78	0.73	0.67	0.12
Grounded	97.75%	98.50%	100.00%	99.97%	0.64	0.77	1.0	1.0
LBA	97.05%	97.81%	0.35%	97.81%	0.7	0.5	-0.7	0.5
Planning2AF	80.76%	84.55%	62.06%	84.73%	0.58	0.67	0.42	0.68
Stable	68.35%	72.36%	64.49%	66.55%	0.32	0.43	0.23	0.27
Traffic	81.49%	88.58%	31.52%	80.18%	0.58	0.56	-0.071	0.47
Watts-Strogatz	76.78%	76.14%	75.25%	73.56%	0.15	0.15	0.0	0.23
admbuster	99.79%	99.73%	100.00%	100.00%	1.0	0.99	1.0	1.0

Table 3.14: Overview of AFGCN approximation results for DC-ST ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	86.37%	89.67%	24.05%	82.83%	0.76	0.41	-0.41	0.45
1	91.25%	91.95%	35.69%	82.41%	0.68	0.68	0.1	0.45
2	93.01%	95.91%	20.42%	92.86%	0.57	0.69	-0.17	0.64
3	68.06%	69.15%	62.44%	68.69%	0.29	0.39	0.19	0.24
4	74.68%	76.74%	72.22%	74.92%	0.16	0.23	0.13	0.22
5	75.31%	75.72%	69.75%	75.64%	0.22	0.22	0.092	0.26
6	83.70%	85.51%	78.62%	82.74%	0.25	0.38	0.2	0.26
7	91.73%	92.65%	87.62%	92.08%	0.4	0.47	0.41	0.47
8	97.43%	98.31%	98.31%	99.22%	0.77	0.83	0.96	0.98
9	99.07%	99.37%	100.00%	100.00%	0.89	0.93	1.0	1.0

a harder approximation problem for a GCN than some of the others we have considered. The overall best performing model is the HYBRID-GCN-GR model, not surprising considering the previous observation.

The complete balanced setting, doesn't change the picture as much as it has in some other semantics. The GR-ONLY model increases performance as expected, but the ordering among the GCN-based models remain constant.

The reduced balanced setting reverts the picture to one fairly close to the equally weighted one. Overall, the HYBRID-GCN-GR model is the clear winner in terms of performance for credulous acceptance under semi-stable semantics.

On the benchmark side, shown in Table 3.15 we can note a similar pattern to stable semantics for AFGen, Barabasi-Albert, Planning2AF, Traffic, Watts-Strogatz, and admbuster graphs. ABA2AF is approximable under semi-stable semantics as it is under preferred semantics. ER graphs prove overall somewhat easier to approximate under semi-stable semantics that we've seen previously, whereas Stable and Traffic benchmarks have some reduced performance. The performance on the LBA benchmark is a bit lower than for stable semantics, but not as bad as for complete semantics. Finally, we can note that the reason the GCN-NO-GR model does poorly under these semantics is mainly attributable to a bad performance on the Grounded benchmark.

Table 3.15: Overview of AFGCN approximation results for DC-SST ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	98.55%	98.91%	99.12%	99.27%	0.53	0.65	0.81	0.83
AFGen	54.60%	55.73%	54.60%	54.60%	-0.1	0.021	-0.1	-0.1
Barabasi-Albert	88.27%	92.21%	49.93%	90.56%	0.71	0.75	0.31	0.72
Erdős–Rényi	93.07%	94.72%	96.04%	96.04%	0.74	0.71	0.67	0.67
Grounded	96.84%	97.99%	100.00%	99.87%	0.38	0.69	1.0	0.99
LBA	29.69%	97.40%	1.48%	96.82%	0.2	0.5	-0.7	0.5
Planning2AF	74.36%	82.69%	62.06%	87.74%	0.46	0.62	0.42	0.75
Stable	64.67%	70.95%	63.55%	70.90%	0.2	0.38	0.2	0.37
Traffic	83.63%	80.65%	32.95%	82.36%	0.46	0.48	0.029	0.51
Watts-Strogatz	75.86%	77.81%	75.25%	78.00%	0.12	0.2	0.0	0.23
admbuster	98.64%	99.75%	100.00%	100.00%	0.97	1.0	1.0	1.0

As can be seen in Table 3.16, the general pattern for size related performance holds for the GR-ONLY, HYBRID-GCN-GR, and GCN-WITH-GR models under semi-stable semantics. However, it breaks for the GCN-NO-GR model as the performance at the low end is much worse that has been seen for other semantics. This demonstrates that while much of the bad performance of this model under semi-stable semantics is attributable to inferior grounded reasoning that is not the whole story.

Table 3.16: Overview of AFGCN approximation results for DC-SST ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	48.41%	87.99%	21.85%	86.12%	0.25	0.61	-0.33	0.51
1	87.44%	88.61%	33.25%	91.00%	0.6	0.58	0.023	0.62
2	57.31%	92.93%	26.20%	90.90%	0.44	0.46	-0.091	0.55
3	67.09%	69.39%	62.44%	70.58%	0.26	0.38	0.19	0.31
4	70.63%	74.92%	66.29%	76.47%	0.12	0.2	0.1	0.25
5	70.64%	73.49%	71.06%	74.34%	0.17	0.31	0.15	0.34
6	80.19%	84.45%	76.76%	84.43%	0.37	0.5	0.42	0.58
7	91.43%	94.17%	90.47%	95.25%	0.56	0.72	0.81	0.85
8	95.53%	97.55%	98.31%	99.21%	0.71	0.82	0.96	0.98
9	98.18%	99.23%	100.00%	100.00%	0.66	0.88	1.0	1.0

Stage semantics

Stage semantics are the only semantics not based on admissible sets of the ones considered in this chapter. One might therefore expect a significantly different results than for the other semantics based on the different way extensions are created. However, we don't see any such radical departure from the patterns we have seen, although as in other cases, we see interesting variation in benchmark specific performance.

Looking first at the equally weighted results for credulous acceptance in figure 3.12, we find the GCN-WITH-GR model performing best both in accuracy and MCC terms. The GR-ONLY model performs relatively poorly under this semantic, which also implies a slight dip in performance for the HYBRID-GCN-GR model.

The complete balanced setting shows the now familiar increase in accuracy and the HYBRID-GCN-GR model performing the best followed by the GR-ONLY model. Once again, we see the pattern revert to one closer to the equally weighted setting once we remove the two large grounded-focused benchmarks. This is consistent with what we have seen for other semantics.

We see benchmark specific behaviour that in many ways is familiar from other semantics, especially semi-stable ones. This is true for ABA2AF, AFGen, Barabasi-Albert, Grounded, Planning2AF, Traffic, and admbuster benchmarks. However, we can note that the GR-ONLY model performs unusually poorly on ER graphs under these semantic. There is slightly better performance from GCN-models on Stable and Watts-Strogatz models and slightly worse performance from all models on the LBA benchmark compared to credulous semi-stable semantics.

When we turn to the analysis based on size bands in Table 3.18 we see the usual patterns of peaks at small and large bands for the GCN models, while the GR-ONLY model has an especially pronounced dip at the lowest band for these semantics.

3.2.2.3 Results for sceptical acceptance

Preferred semantics

Now we turn attention to sceptical acceptance under the preferred semantics.

We start again with the equally weighted setting (refer to figure 3.13). On an equally weighted basis results are slightly better overall than for credulous acceptance. Unsurprisingly, it is much better for the GR-ONLY model that actually is the second best performing on an MCC basis. The best performing model is the GCN-WITH-GR model, which is somewhat surprising given the better performance we saw from the hybrid model on the Grounded benchmark for credulous acceptance. However, the GCN-WITH-GR model would seem to have better ability to generalise sceptical acceptance across benchmarks leading to the overall higher score.

Looking instead at the complete balanced setting, the GR-ONLY model is the overall winner followed by GCN-WITH-GR both in terms of accuracy and MCC, largely reflecting its superior performance on the Grounded benchmark. The hybrid model does particularly poorly in this evaluation, which reflects an overoptimism in the configured thresholds leading to low precision.

Excluding the Grounded and admbuster benchmarks, the GCN-WITH-GR model edges ahead of pure grounded reasoning in both accuracy and MCC, reflecting this models better ability to generalise across benchmarks. The GCN-NO-GR model performs the worst under this setting, although the HYBRID-GCN-GR model is still underperforming due to low precision.

Table 3.17: Overview of AFGCN approximation results for DC-STG ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	97.47%	98.03%	97.43%	97.87%	0.48	0.66	0.52	0.63
AFGen	58.36%	58.03%	54.60%	61.28%	0.018	0.051	-0.1	0.1
Barabasi-Albert	88.82%	93.64%	49.93%	93.77%	0.69	0.78	0.31	0.79
Erdős–Rényi	72.94%	95.05%	29.37%	70.96%	0.37	0.82	-0.67	0.68
Grounded	98.13%	98.46%	100.00%	99.71%	0.71	0.77	1.0	0.97
LBA	84.12%	96.77%	0.90%	94.74%	-0.1	0.4	-0.8	0.1
Planning2AF	82.29%	82.98%	62.06%	81.62%	0.62	0.65	0.42	0.63
Stable	70.77%	71.71%	63.70%	70.23%	0.38	0.43	0.22	0.38
Traffic	83.55%	85.35%	25.81%	83.30%	0.46	0.55	-0.054	0.52
Watts-Strogatz	78.19%	77.75%	75.25%	75.28%	0.27	0.2	0.0	0.024
admbuster	99.78%	99.62%	100.00%	100.00%	1.0	0.99	1.0	1.0

Table 3.18: Overview of AFGCN approximation results for DC-STG ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	82.21%	93.17%	14.12%	93.32%	0.23	0.58	-0.52	0.44
1	83.88%	88.88%	33.25%	85.49%	0.45	0.61	0.023	0.54
2	90.23%	93.94%	26.20%	92.96%	0.3	0.51	-0.091	0.35
3	66.02%	72.77%	44.26%	68.06%	0.28	0.41	-0.17	0.46
4	74.71%	74.40%	67.24%	74.81%	0.27	0.21	0.14	0.24
5	73.73%	71.81%	70.70%	72.08%	0.31	0.37	0.14	0.18
6	85.28%	87.25%	79.85%	84.26%	0.38	0.56	0.36	0.48
7	92.58%	91.99%	87.13%	92.41%	0.74	0.75	0.63	0.75
8	97.69%	98.10%	98.31%	99.12%	0.8	0.83	0.96	0.98
9	99.18%	99.23%	100.00%	100.00%	0.91	0.92	1.0	1.0

Considering the benchmark evaluation in Table 3.19, the ABA2AF benchmark shows a common phenomenon when dealing with sceptical acceptance, which is very high accuracy, but significantly lower MCC, which is due to a large imbalance in favour of negative cases in the data. This is demonstrated perfectly by the AFGen benchmark, where 93.96% accuracy is revealed to have no predictive power by the MCC score. Watts-Strogatz graphs reveal similar behaviour, but less strongly. For Barabasi-Albert graphs and ER graphs all models perform effectively on par, excepting a small dip for the hybrid model on BA graphs. Grounded graphs are solved perfectly by the grounded model, but only well by the various GCN model. On the other hand the grounded model does not have any predictive power for LBA graphs, while the scores for the GCN models are much lower than in the credulous setting. Planning2AF graph performance is on par between the GR-ONLY and the GCN-WITH-GR models, indicating that in this case the GCN simply applies grounded reasoning via the approximation. For the Stable benchmark, the GCN-WITH-GR model has best performance. In contrast, it has worst performance on the Traffic benchmark, while other models are approximately on par. Neither of these phenomena are readily explainable. Admbuster graphs are once again solved near-perfectly in the sceptical setting.

Table 3.19: Overview of AFGCN approximation results for DS-PR ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	99.21%	99.55%	99.52%	99.52%	0.68	0.81	0.84	0.84
AFGen	93.96%	93.96%	93.96%	93.96%	0.0	0.0	0.0	0.0
Barabasi-Albert	84.22%	84.98%	82.86%	81.08%	0.69	0.71	0.71	0.61
Erdős–Rényi	96.04%	96.04%	96.04%	96.04%	0.67	0.67	0.67	0.67
Grounded	98.04%	98.64%	100.00%	92.37%	0.73	0.79	1.0	0.68
LBA	68.72%	81.40%	50.50%	68.92%	0.4	0.62	0.045	0.43
Planning2AF	81.76%	88.29%	88.20%	84.99%	0.56	0.72	0.72	0.65
Stable	79.27%	81.36%	79.12%	75.68%	0.29	0.4	0.27	0.21
Traffic	70.77%	61.35%	69.26%	68.50%	0.36	0.23	0.36	0.34
Watts-Strogatz	82.08%	82.64%	82.03%	81.69%	0.02	0.15	0.0	0.011
admbuster	99.73%	99.92%	100.00%	100.00%	0.99	1.0	1.0	1.0

The size based performance evaluation shown in Table 3.20 has similar patterns to the credulous case for the GR-ONLY model presumably for much the same reasons. The general trend for sceptical acceptance under the preferred semantics seems to be the larger the framework, the better the performance, which may have something to do with the large class imbalance seen in sceptical acceptance in favour of negative cases.

Table 3.20: Overview of AFGCN approximation results for DS-PR ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	76.88%	70.97%	62.85%	68.28%	0.52	0.48	0.27	0.37
1	75.05%	76.05%	69.90%	75.19%	0.42	0.44	0.34	0.46
2	68.65%	79.02%	69.07%	74.29%	0.27	0.45	0.32	0.38
3	91.90%	91.58%	92.20%	91.86%	0.49	0.49	0.5	0.49
4	87.22%	89.97%	89.58%	88.56%	0.16	0.25	0.24	0.24
5	86.20%	87.76%	86.26%	85.01%	0.21	0.4	0.23	0.2
6	85.47%	87.63%	84.39%	81.60%	0.43	0.56	0.45	0.41
7	95.96%	96.91%	97.37%	96.24%	0.8	0.85	0.92	0.87
8	97.39%	98.72%	99.86%	93.75%	0.84	0.87	0.99	0.79
9	99.01%	99.33%	100.00%	97.20%	0.86	0.9	1.0	0.86

Complete semantics

The sceptical setting for the complete semantics is equal to the grounded semantics and computable in polynomial time. You would therefore never in practice want to approximate this task. However, for the sake of completeness, we will still run through the results.

The equally weighted setting shown in figure 3.14, unsurprisingly shows perfect performance for the GR-ONLY model with near-perfect performance for the HYBRID-GCN-GR model, indicating that it is using a .99 threshold for all size bands and therefore almost always use a grounded reasoner to answer. The other GCN models do not reach the same level of performance, indicating that they have not learnt pure grounded reasoning.

The picture is identical for the complete balanced setting, although the underperformance of the GCN models is less marked.

The reduced balanced setting is somewhere in between the two other evaluation settings, but shows the same overall pattern.

The benchmark level view in Table 3.21 shows us where the difficulties are in approximation. The hardest to approximate frameworks in this setting are the Planning2AF and

Traffic benchmarks, accounting for most of the reduced performance in the GCN models. It would be worth a separate investigation to see why these are hard to approximate.

The size based analysis in this setting, shown in Table 3.22 does not seem to reflect any patterns beyond those that are related to the benchmarks.

Stable semantics

Moving on to the sceptical setting, we see a drop in performance for the GR-ONLY model, relative to the other semantics we have considered when considering the equally weighted performance in figure 3.15. In contrast, the three GCN models are within the same performance envelope, once again reinforcing the view that grounded reasoning doesn't add as much to an approximation attempt under stable semantics as it does under other semantics we have seen.

The complete balanced setting, as with past cases, accentuates the performance of the pure grounded elements in the GR-ONLY and HYBRID-GCN-GR models. In contrast, the GCN-WITH-GR model goes to becoming the worst performing in this setting, because it has learned to generalise more across benchmarks at the cost of underperforming on the Grounded one.

In the reduced setting, the trend again reverses and the three GCN-based models are once again within the same performance envelope. The GCN-NO-GR model is marginally ahead as it was for the equally weighted setting, but not enough to be noteworthy.

The benchmark specific analysis for the stable semantics can be seen in Table 3.23. Comparing to the results for sceptical acceptance under the preferred semantics, we find rough equivalence of results for AFGen, ER, Grounded, Planning2AF, Stable, Watt-Strogatz, and admbuster. As was the case with credulous acceptance, the ABA2AF frameworks are not approximable under stable semantics. There are notable performance drops for Barabasi-Albert and Traffic benchmarks and a large increase in performance for LBA frameworks.

As we've seen before, the analysis of size bands reveals a generally increasing trend in accuracy and MCC for sceptical acceptance. This can be seen in Table 3.24.

Semi-stable semantics

Considering sceptical acceptance under the semi-stable semantics, the results are less divergent than for credulous acceptance. The equally weighted setting in figure 3.16 shows the HYBRID-GCN-GR model having the best performance on MCC and effectively equal

Table 3.21: Overview of AFGCN approximation results for DS-CO ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	99.76%	99.83%	100.00%	100.00%	0.84	0.85	1.0	1.0
AFGen	100.00%	100.00%	100.00%	100.00%	1.0	1.0	1.0	1.0
Barabasi-Albert	97.29%	97.25%	100.00%	99.76%	0.95	0.94	1.0	1.0
Erdős–Rényi	100.00%	100.00%	100.00%	100.00%	1.0	1.0	1.0	1.0
Grounded	98.49%	98.28%	100.00%	99.99%	0.78	0.75	1.0	1.0
LBA	99.97%	100.00%	100.00%	100.00%	0.9	1.0	1.0	1.0
Planning2AF	90.75%	92.07%	100.00%	99.86%	0.71	0.75	1.0	1.0
Stable	99.93%	100.00%	100.00%	100.00%	0.97	1.0	1.0	1.0
Traffic	84.20%	88.38%	100.00%	96.79%	0.2	0.53	1.0	0.94
Watts-Strogatz	100.00%	100.00%	100.00%	100.00%	1.0	1.0	1.0	1.0
admbuster	99.94%	99.83%	100.00%	100.00%	1.0	1.0	1.0	1.0

Table 3.22: Overview of AFGCN approximation results for DS-CO ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	87.12%	89.87%	100.00%	97.08%	0.35	0.67	1.0	0.95
1	93.44%	95.67%	100.00%	99.71%	0.77	0.79	1.0	0.99
2	98.63%	99.26%	100.00%	99.82%	0.97	0.99	1.0	1.0
3	98.51%	97.96%	100.00%	99.94%	0.96	0.94	1.0	1.0
4	98.09%	98.80%	100.00%	100.00%	0.92	0.95	1.0	1.0
5	99.91%	99.95%	100.00%	100.00%	0.84	0.97	1.0	1.0
6	97.81%	97.66%	100.00%	100.00%	0.85	0.87	1.0	1.0
7	98.75%	98.63%	100.00%	99.86%	0.88	0.89	1.0	1.0
8	98.99%	98.69%	100.00%	99.99%	0.9	0.87	1.0	1.0
9	99.35%	99.15%	100.00%	100.00%	0.9	0.88	1.0	1.0

Table 3.23: Overview of AFGCN approximation results for DS-ST ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	99.46%	99.38%	98.80%	98.80%	0.035	0.05	0.05	0.05
AFGen	94.04%	93.37%	94.04%	94.04%	0.0	0.0061	0.0	0.0
Barabasi-Albert	83.11%	81.41%	68.60%	84.45%	0.62	0.58	0.53	0.66
Erdős–Rényi	96.04%	92.75%	96.04%	96.04%	0.67	0.41	0.67	0.67
Grounded	98.11%	97.73%	100.00%	99.98%	0.71	0.65	1.0	1.0
LBA	98.12%	99.38%	10.25%	98.75%	0.8	0.9	0.0055	0.8
Planning2AF	86.51%	83.26%	78.43%	84.29%	0.71	0.64	0.57	0.67
Stable	82.08%	81.80%	81.56%	81.55%	0.3	0.29	0.27	0.27
Traffic	58.53%	61.25%	63.56%	55.72%	0.21	0.16	0.24	0.1
Watts-Strogatz	81.92%	82.00%	81.92%	81.92%	0.0	0.031	0.0	0.0
admbuster	99.74%	99.52%	100.00%	100.00%	0.99	0.99	1.0	1.0

Table 3.24: Overview of AFGCN approximation results for DS-ST ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	71.41%	77.67%	43.25%	74.99%	0.29	0.4	0.16	0.33
1	74.82%	70.83%	63.28%	69.79%	0.49	0.38	0.38	0.35
2	93.14%	93.33%	41.23%	93.63%	0.7	0.71	0.14	0.72
3	92.32%	91.63%	90.99%	92.95%	0.37	0.31	0.35	0.39
4	88.32%	84.46%	83.35%	86.59%	0.31	0.25	0.22	0.27
5	88.88%	88.82%	88.61%	88.64%	0.096	0.093	0.097	0.098
6	92.13%	91.36%	87.86%	89.85%	0.33	0.34	0.26	0.29
7	90.80%	90.86%	91.14%	90.97%	0.43	0.42	0.47	0.46
8	97.32%	96.70%	99.18%	98.99%	0.79	0.74	0.98	0.97
9	99.35%	99.03%	100.00%	100.00%	0.9	0.87	1.0	1.0

on accuracy with the two other GCN-based models. This is due to increased positive accuracy from the grounded reasoner, as both of the other models have better negative accuracy.

In the complete balanced setting, considering the large grounded frameworks, the performance of the GR-ONLY model and the HYBRID-GCN-GR model are effectively identical.

Removing the two large grounded benchmarks, leads to the HYBRID-GCN-GR model again coming out ahead. But under these semantics the GR-ONLY model remains very competitive. For both of the balanced settings the GCN-NO-GR model outperforms the GCN-WITH-GR model, indicating that for these semantics the model has not learnt to reason effectively with grounded features.

Relative to the stable semantics the results are mainly consistent across benchmarks as shown in Table 3.25. ABA2AF is approximable again for sceptical acceptance as well, which is consistent with preferred semantics. There is a drop in performance for LBA frameworks as there was for credulous acceptance. In contrast, there is a performance increase for the Traffic domain.

The performance based on size reveals overall lower performance in the smaller size bands as shown in Table 3.26. However, there is the same overall pattern that we have seen in general for sceptical acceptance that performance increases with size.

Stage semantics

Moving on to sceptical acceptance under stage semantics, we see in figure 3.17 that the GCN-WITH-GR model is the best performer overall in the equally weighted setting as it was for credulous acceptance. As expected the GR-ONLY model does much better in the sceptical context and is on par with the GCN-NO-GR model measured by MCC.

The HYBRID-GCN-GR model and the GR-ONLY model are indistinguishable in terms of performance in the complete balanced setting. The other models also retain good performance in this setting.

The reduced balanced setting has results closer in accuracy terms than we've seen previously, but taking MCC into account, the GCN-WITH-GR slightly outperforms the pack as in the equally weighted setting.

There is a fair degree of overlap with semi-stable semantics in the case of benchmark

Table 3.25: Overview of AFGCN approximation results DS-SST ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	99.10%	99.09%	99.43%	99.44%	0.65	0.63	0.82	0.81
AFGen	93.96%	93.96%	93.96%	93.96%	0.0	0.0	0.0	0.0
Barabasi-Albert	82.93%	83.10%	85.38%	84.43%	0.66	0.65	0.74	0.7
Erdős–Rényi	96.04%	96.04%	96.04%	96.04%	0.67	0.67	0.67	0.67
Grounded	98.14%	97.72%	100.00%	100.00%	0.72	0.64	1.0	1.0
LBA	71.23%	77.34%	53.34%	64.67%	0.49	0.57	0.11	0.3
Planning2AF	87.38%	82.11%	88.44%	85.84%	0.7	0.55	0.72	0.66
Stable	79.47%	79.78%	78.75%	80.32%	0.32	0.29	0.23	0.32
Traffic	68.46%	69.34%	69.26%	70.40%	0.42	0.4	0.36	0.36
Watts-Strogatz	82.50%	83.03%	82.03%	82.92%	0.14	0.16	0.0	0.15
admbuster	99.82%	99.68%	100.00%	100.00%	1.0	0.99	1.0	1.0

Table 3.26: Overview of AFGCN approximation results for DS-SST ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	69.87%	76.89%	60.54%	64.18%	0.45	0.56	0.24	0.3
1	69.55%	70.28%	65.09%	70.32%	0.44	0.38	0.26	0.33
2	86.37%	85.41%	88.12%	87.57%	0.52	0.49	0.56	0.55
3	86.28%	84.96%	86.01%	87.22%	0.44	0.43	0.46	0.46
4	89.42%	88.10%	89.37%	89.40%	0.22	0.18	0.22	0.22
5	87.00%	87.21%	86.44%	87.64%	0.28	0.26	0.21	0.32
6	89.99%	89.49%	87.87%	88.46%	0.58	0.56	0.58	0.61
7	93.93%	92.11%	94.86%	94.86%	0.71	0.65	0.83	0.83
8	97.93%	97.18%	99.87%	99.75%	0.84	0.76	1.0	0.99
9	99.46%	99.35%	100.00%	100.00%	0.89	0.87	1.0	1.0

specific performance. We can see from Table 3.27 that seven benchmarks have similar patterns including ABA2AF, AFGen, Grounded, Planning2AF, Stable, Watts-Strogatz, and admbuster benchmarks. There is increased performance for all models on Barabasi-Albert graphs, while ER graphs are unapproximable under these semantics. All GCN-based models perform slightly better on LBA frameworks and all models perform slightly worse on Traffic frameworks.

The size based results for stage semantics are shown in Table 3.28. They are consistent with what we have seen for previous semantics and do not present a distinctive pattern for consideration.

Id semantics

The Id semantics are defined by the largest admissible set that is a member of all preferred extension. As such it is related to the grounded extension and like the grounded extension one cannot distinguish between sceptical and credulous acceptance as the Id extension is unique.

Considering the results for sceptical acceptance using the equally weighted setting, shown in figure 3.18, we find that despite the conceptual similarity with grounded reasoning, the GR-ONLY model does not perform exceptionally well under these semantics. Instead, the HYBRID-GCN-GR model has overall best performance, followed by the GCN-WITH-GR model.

The picture for the complete balanced setting is the familiar one with the GR-ONLY and HYBRID-GCN-GR models performing more or less equivalently with the other GCN models following somewhat behind.

Unsurprisingly, this picture changes if we remove the two large grounded-focused frameworks from the equation. This evaluation setting results in the HYBRID-GCN-GR model outperforming the rest, which are relatively close in performance.

The benchmark specific performance is nearly indistinguishable from that under sceptical preferred semantics. All 11 benchmarks are sufficiently close that it is hard to ascribe the minor deviations to anything but chance, excepting a slight reduction across the board for the ABA2AF benchmark. This makes a certain amount of sense, as the overlap between the set of arguments contained in the largest admissible subset of all preferred extensions of an argumentation framework will share much with the set of arguments sceptically accepted for that framework under preferred semantics.

Table 3.27: Overview of AFGCN approximation results for DS-STG ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	98.36%	98.86%	98.72%	98.72%	0.49	0.66	0.67	0.67
AFGen	93.91%	93.97%	93.91%	93.91%	0.0	0.027	0.0	0.0
Barabasi-Albert	88.11%	89.35%	89.52%	90.30%	0.77	0.78	0.81	0.82
Erdős–Rényi	94.73%	94.73%	94.73%	94.73%	0.0	0.0	0.0	0.0
Grounded	97.92%	98.43%	100.00%	100.00%	0.73	0.78	1.0	1.0
LBA	81.96%	85.60%	57.96%	71.60%	0.61	0.69	0.046	0.33
Planning2AF	85.14%	86.41%	87.69%	87.14%	0.64	0.68	0.71	0.69
Stable	80.08%	81.14%	80.03%	80.26%	0.29	0.36	0.26	0.27
Traffic	63.27%	67.03%	64.97%	70.34%	0.21	0.35	0.29	0.35
Watts-Strogatz	81.92%	83.42%	81.92%	81.92%	0.0	0.23	0.0	0.0
admbuster	99.73%	99.95%	100.00%	100.00%	0.99	1.0	1.0	1.0

Table 3.28: Overview of AFGCN approximation results for DS-STG ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	69.06%	76.87%	56.77%	69.21%	0.41	0.57	0.2	0.41
1	74.84%	75.57%	67.02%	73.87%	0.41	0.47	0.19	0.34
2	89.59%	90.25%	89.55%	89.80%	0.39	0.41	0.38	0.37
3	90.52%	90.51%	91.30%	90.99%	0.41	0.44	0.43	0.42
4	86.30%	86.74%	87.31%	87.27%	0.26	0.28	0.29	0.28
5	86.50%	87.64%	86.59%	86.82%	0.14	0.35	0.2	0.21
6	90.59%	91.10%	89.70%	89.70%	0.49	0.56	0.56	0.56
7	92.74%	93.46%	93.88%	93.88%	0.65	0.76	0.7	0.7
8	97.42%	98.40%	99.87%	99.86%	0.83	0.86	1.0	0.99
9	99.42%	99.61%	100.00%	100.00%	0.89	0.9	1.0	1.0

The size band based analysis, shown in 3.30 shows the pattern of generally ascending performance with size that we are used to, but the performance at the low end is relatively good compared to some other semantics.

3.2.2.4 Cross-cutting results

In this section, we will look at the results across the various parameters that we have used for our analysis. This includes analyses across semantics, benchmarks, and sizes. We do this in order to show any general results that are not specific to individual semantics. We will start by presenting the cross-cutting analysis based on semantics.

By semantic

Based on the overview of results by semantics presented in Table 3.31, it is difficult to clearly identify whether there are significant differences in the ease at which semantics can be approximated using these models. Excluding the DS-CO task, which can be calculated in polynomial time, the spread between approximation performance in both accuracy and MCC terms is low. Overall, considering both factors, semi-stable semantics would seem to be the easiest to approximate and stable semantics the hardest. But the difference is not large enough to make a substantial point.

Identifying a best performing model is also difficult. The GR-ONLY model unsurprisingly wins the DS-CO task, but for the other task, they are very close between the HYBRID-GCN-GR model and the GCN-WITH-GR model. For DC-CO and DC-PR, they are close enough in performance to be indistinguishable. For the DC-SST, DS-ID, DS-SST, and DS-ST tasks, the HYBRID-GCN-GR models outperform the GCN-WITH-GR model. For DC-ST, DC-STG, DS-PR, and DS-STG tasks it is the other way around. That means that we cannot give a clear answer to whether it is better to incorporate grounded reasoning only using features fed to the neural network or whether there is a benefit to adding a grounded solver to the mix.

We can, however, note that incorporating grounded reasoning into a GCN model using either mechanism results in increased performance relative to a model that does not. The performance boost is modest, but consistent across semantics, considering both variants that include grounded reasoning.

By benchmark

The benchmark specific results, shown in Table 3.32 are more varied than was the case

Table 3.29: Overview of AFGCN approximation results DS-ID ordered by benchmark

Benchmark	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	99.31%	99.13%	99.36%	99.39%	0.7	0.7	0.77	0.78
AFGen	93.96%	93.96%	93.96%	93.96%	0.0	0.0	0.0	0.0
Barabasi-Albert	85.89%	86.10%	87.03%	86.96%	0.72	0.71	0.77	0.74
Erdős–Rényi	96.04%	96.04%	96.04%	95.38%	0.67	0.67	0.67	0.65
Grounded	97.41%	97.94%	100.00%	99.99%	0.69	0.69	1.0	1.0
LBA	65.74%	73.68%	49.72%	69.76%	0.39	0.53	0.12	0.5
Planning2AF	83.23%	87.22%	88.12%	91.60%	0.61	0.69	0.72	0.8
Stable	80.63%	80.41%	79.24%	80.02%	0.34	0.3	0.23	0.27
Traffic	68.69%	67.59%	69.26%	65.95%	0.4	0.32	0.36	0.3
Watts-Strogatz	82.42%	82.17%	82.03%	82.36%	0.11	0.15	0.0	0.08
admbuster	99.90%	99.90%	100.00%	100.00%	1.0	1.0	1.0	1.0

Table 3.30: Overview of AFGCN approximation results for DS-ID ordered by band

Band	**Accuracy**				**MCC**			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	69.57%	72.74%	66.60%	72.31%	0.42	0.44	0.37	0.46
1	67.53%	69.86%	64.37%	65.67%	0.38	0.4	0.3	0.34
2	81.04%	82.79%	74.53%	84.22%	0.41	0.46	0.33	0.49
3	91.21%	91.74%	91.61%	91.81%	0.37	0.4	0.39	0.39
4	86.74%	89.62%	89.70%	90.92%	0.36	0.4	0.41	0.44
5	87.71%	87.38%	86.89%	87.51%	0.35	0.33	0.22	0.29
6	91.95%	91.85%	90.28%	92.21%	0.62	0.61	0.61	0.66
7	89.05%	89.70%	90.77%	90.77%	0.57	0.58	0.68	0.68
8	97.85%	98.28%	99.89%	99.88%	0.84	0.83	1.0	1.0
9	99.46%	99.51%	100.00%	100.00%	0.87	0.88	1.0	1.0

Table 3.31: Overview of AFGCN approximation results compared across semantics

Semantics	Accuracy				MCC			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
DC-PR	83.10%	83.96%	63.98%	84.69%	0.54	0.58	0.34	0.58
DC-CO	81.31%	88.02%	63.86%	86.58%	0.46	0.58	0.28	0.58
DC-ST	85.62%	87.06%	64.19%	84.66%	0.49	0.52	0.24	0.49
DC-SST	77.04%	86.00%	64.41%	86.64%	0.42	0.55	0.32	0.6
DC-STG	84.05%	86.84%	61.45%	85.76%	0.46	0.57	0.23	0.54
DS-PR	86.24%	87.66%	84.99%	85.14%	0.5	0.56	0.52	0.5
DS-CO	97.00%	97.54%	100.00%	99.64%	0.83	0.88	1.0	0.99
DS-ST	88.65%	88.29%	78.10%	88.44%	0.46	0.45	0.39	0.48
DS-SST	86.75%	86.94%	85.51%	86.63%	0.53	0.51	0.52	0.55
DS-STG	87.48%	88.81%	85.90%	87.86%	0.48	0.55	0.48	0.52
DS-ID	86.16%	87.28%	85.33%	87.44%	0.52	0.53	0.52	0.57

for the analysis based on semantics. We can start with the admbuster benchmark, which is designed to foil certain types of solvers and note that it does not manage to do so for any of the models under consideration here.

The Grounded benchmark is a major factor in the evaluation, due to the large size of the frameworks and their focus on grounded reasoning. Here we see that the GR-ONLY model has its expected perfect performance, followed closely by the HYBRID-GCN-GR model for the simple reason that it will default to grounded reasoning in the majority of cases. The difference in performance to the GCN-WITH-GR model indicates that these two models have substantially different ways of incorporating grounded reasoning.

Four other benchmarks have a large component of grounded reasoning: ABA2AF, ER, Planning2AF, and Barabasi-Albert. Interestingly, for both ER and Barabasi-Albert graphs, the better performance for the GCN-models comes from the GCN part of the equation, despite the importance of grounded reasoning that can be seen from the performance of the GR-ONLY model. The reverse seems to be the case for ABA2AF. For Planning2AF there is a small boost from combining both GCN and grounded reasoning.

ER and LBA graphs both varied substantially in the performance seen across the semantics. In aggregate, they end up being fairly approximable, which is obviously misleading given the specific results we have seen.

Stable and Traffic benchmarks have similar and rather middling performance in aggregate, which is the result of the small but significant variations that were seen across semantics for these benchmarks.

AFGen and Watts-Strogatz graphs are some of the most consistent benchmarks in the set, given that they are weakly approximable or unapproximable by these models across semantics.

Table 3.32: Overview of AFGCN approximation results compared across benchmarks for all semantics

Benchmark	Accuracy				MCC			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
ABA2AF	98.81%	99.07%	98.96%	99.04%	0.49	0.59	0.63	0.64
AFGen	75.63%	76.51%	74.89%	77.20%	0.083	0.13	0.05	0.12
Barabasi-Albert	89.24%	89.69%	67.60%	89.32%	0.74	0.75	0.53	0.75
Erdős–Rényi	89.90%	94.79%	89.64%	90.87%	0.6	0.57	0.5	0.52
Grounded	97.93%	98.27%	100.00%	99.06%	0.68	0.74	1.0	0.95
LBA	79.86%	93.14%	27.56%	88.94%	0.53	0.65	-0.24	0.53
Planning2AF	80.66%	84.04%	74.83%	85.00%	0.57	0.64	0.58	0.68
Stable	75.77%	77.77%	73.58%	76.28%	0.37	0.43	0.31	0.38
Traffic	75.64%	76.74%	52.32%	76.64%	0.37	0.43	0.22	0.45
Watts-Strogatz	80.86%	81.53%	80.41%	80.87%	0.19	0.26	0.1	0.19
admbuster	99.63%	99.72%	100.00%	100.00%	0.99	0.99	1.0	1.0

By size

Size was less of a factor in the run-through of results than benchmarks, semantics, or evaluation setting. We saw two separate patterns for credulous and sceptical acceptance. The first had highs for small and large size bands, whereas the second had generally increasing performance by size, although the GCN-ONLY model maintained the second pattern also for credulous acceptance.

Looking at these results in aggregate, the GR-ONLY model retains the pattern seen during the past results presentation. It is simply a bit more smoothed. For the GCN-based models the pattern has smoothed so the highs at the bottom and the dip in the middle have reduced. Instead there is fairly consistent performance except for at the very high end.

We speculated earlier that there might be an increase in approximability, especially for sceptical acceptance. These results bear out that hypothesis, but it is not conclusive given the prevalence of easier to approximate benchmarks at the high end of the size scale. Other size effects observed in the specific semantics would seem to disappear when considered in aggregate.

Table 3.33: Overview of AFGCN approximation results compared across size bands for all semantics

Band	Accuracy				MCC			
	NO-GR	W/GR	GR	HYBR	NO-GR	W/GR	GR	HYBR
0	73.46%	82.92%	43.83%	79.65%	0.41	0.51	0.024	0.45
1	83.56%	85.35%	51.78%	84.20%	0.55	0.59	0.21	0.56
2	86.18%	91.04%	53.24%	90.39%	0.55	0.61	0.2	0.6
3	78.30%	80.32%	74.93%	80.90%	0.4	0.43	0.34	0.44
4	80.59%	82.59%	79.47%	83.15%	0.28	0.32	0.28	0.34
5	80.69%	81.52%	79.91%	81.46%	0.25	0.34	0.23	0.3
6	87.38%	88.79%	83.33%	86.59%	0.47	0.56	0.46	0.52
7	92.91%	93.45%	91.76%	93.60%	0.65	0.7	0.72	0.74
8	97.72%	98.17%	99.34%	99.01%	0.81	0.83	0.98	0.97
9	99.09%	99.27%	100.00%	99.72%	0.87	0.9	1.0	0.99

3.2.2.5 Runtime performance

Runtime performance is one of the major reasons one might consider using an approximate approach to solving abstract argumentation problems. Here we consider the runtime in a variety of contexts focusing on a comparison that either includes or excludes the time it takes to compute the grounded extension for an argumentation framework.

Table 3.34: Overview of AFGCN runtime results, key statistics

index	Runtime with GR	Runtime without GR
min	6.83ms	6.12ms
25%	12.44ms	10.55ms
50%	28.96ms	20.72ms
75%	810.58ms	242.72ms
max	21563.85ms	4922.45ms

The runtime statistics shown in Table 3.34 gives the results breakdown for classifying an entire argumentation framework with all its arguments. This table shows that while most frameworks can be fully classified in less than a second including the overheads needed to initialise the model, this can increase substantially for the worst case. It also shows that the cost of computing the grounded extension increases disproportionately with scale. This is as expected as the algorithm to compute the grounded extension has polynomial runtime in the number of arguments.

If we look at the median runtime to classify a single argument broken down by semantics shown in Table 3.35, we see that the difference per argument of including the grounded features is approximately 0.015ms. This may be within the acceptable boundary for many applications. We can also observe significant differences in the runtime for different semantics and a general tendency for runtime to be slightly slower for sceptical than for credulous acceptance. See figure 3.19 for the distribution.

Table 3.35: Overview of AFGCN runtime results ordered by semantics. Median runtime given. Results in seconds.

Semantics	Runtime w/GR	Runtime No GR
DC-CO	0,027	0,020
DC-PR	0,031	0,022
DC-SST	0,031	0,021
DC-ST	0,029	0,020
DC-STG	0,029	0,020
DS-CO	0,029	0,022
DS-ID	0,042	0,031
DS-PR	0,028	0,022
DS-SST	0,029	0,019
DS-ST	0,029	0,020
DS-STG	0,027	0,020

The difference in runtime by benchmark is nearly two orders of magnitude between the fastest, Planning2AF, and the slowest, ABA2AF. While some variation would be expected by benchmark, this is unexpectedly large. You can roughly group the benchmarks into three classes: Fast, Medium, Slow. Fast benchmarks include Planning2AF, Stable, admbuster, Watts-Strogatz, and Barabasi-Albert. Medium include AFGen, ER, Grounded, LBA, and Traffic. ABA2AF is in its own slow category. Interestingly, this partitioning does not straightforwardly map to the classification performance for these benchmarks. See figure 3.20 for an overview of the distribution.

Table 3.36: Overview of AFGCN runtime results ordered by benchmark. Median given. Results in seconds.

Benchmark	Runtime w/GR	Runtime No GR
ABA2AF	1,79	1,32
AFGen	0,06	0,05
Barabasi-Albert	0,01	0,01
Erdos-Renyi	0,03	0,03
Grounded	1,84	0,55
LBA	0,01	0,01
Planning2AF	0,02	0,01
Stable	0,04	0,02
Traffic	0,01	0,01
Watts-Strogatz	0,02	0,02
admbuster	2,61	0,10

Table 3.37: Overview of AFGCN runtime results ordered by size. Median given. Results in seconds.

Band	Runtime w/GR	Runtime No GR
(4.999, 19.0]	0,01	0,01
(19.0, 30.0]	0,01	0,01
(30.0, 51.0]	0,01	0,01
(51.0, 99.0]	0,02	0,02
(99.0, 195.5]	0,02	0,02
(195.5, 380.0]	0,03	0,02
(380.0, 547.0]	0,10	0,04
(547.0, 696.0]	1,85	1,35
(696.0, 1992.0]	1,02	0,41
(1992.0, 10000.0]	3,69	0,40

Turning to the results by size band in Table 3.37, we do see an obvious and expected pattern. Large frameworks as expected result in longer runtimes and this increases with size fairly reliably, although the band (547.0, 696.0] is an outlier in this regard presumably because it contains more samples from hard benchmarks than the other bands. See figure 3.21 for an overview of the distribution.

3.3 Summary

This chapter has presented the first systematic results from applying deep learning based approximation approaches to key problems in abstract argumentation. First, we can note that, in general, argumentation frameworks adhering to a variety of schemes can be approximated moderately well to very well by an approach that combines grounded reasoning with graph neural networks. This is true of both credulous and sceptical acceptance and across semantics.

There are cases that prove unapproximable or very hard to approximate and require further analysis such as the unapproximability of ER graphs under some but not all semantics and the general low approximability of Watts-Strogatz graphs.

However, the one benchmark that is generally unapproximable, AFGen, is likely so for the reason that it is a random graph model with very little structure in its generating function. This means it does not contain enough regularity for a neural network to learn anything.

It's worth noting that ER and Watts-Strogatz are also random graph models with different generating functions, so in general we can suggest that approximation for random graph models is problematic with our chosen approaches. However, differences in how the random graph model is generated does seem to matter in terms of learnability.

We can also conclude that while a GCN-based approach on its own is a good approximator, it is a better approximator when combined with grounded reasoning, although we cannot definitively conclude what is the best way to combine grounded reasoning with GCN-based approaches on the basis of these results.

A grounded reasoner is a good, but not perfect approximator for sceptical acceptance across semantics and in general the improvement made by adding a GCN model is small for sceptical acceptance. On the basis of these results, one might be tempted to conclude that it is not worth bothering with additional approximation approaches for sceptical acceptance unless one is dealing with problems of a scale where a marginal improvement is worth a substantial investment. The problem with this position, however, is that the grounded reasoner is fixed. It will never provide a better approximation than it already does, which is still no better than 80% accuracy on an equally weighted basis in most cases. While the current approach only improves marginally on this by adding a GCN, it at least shows it is possible to improve on this baseline. Further research may lead to greater improvements still.

Perhaps the most promising line of enquiry coming out of this research can be found by considering the considerable difference in performance found across benchmarks and semantics. For some benchmarks in some semantics, such as LBA frameworks under preferred semantics, the approximation performance is near perfect. This begs the question, whether the quest to create a general purpose approximator for abstract argumentation is actually a foolish one and whether the more profitable approach might not be to create task specific approximators depending on the problem at hand.

Referring back to our research questions from section 1.2, we have given answers to RQs 1-4. That is to say, we have confirmed that it is possible to create a high-performing approximation approach for abstract argumentation using GNNs. We developed a unique training approach, using a modified GCN architecture that works effectively in this context. We also discussed in detail the effects of bringing in the grounded extension as a starting point and demonstrated that it can provide a boost in approximation performance in many cases. Finally, we systematically evaluated differences in performance across semantics, benchmarks, and size bands, revealing much variability.

This chapter, then, has provided the answers to our basic research questions and we will now continue to explore ways of extending and applying this approach to other related areas. First, we will look at how the approximate approach can work together with exact solution approaches.

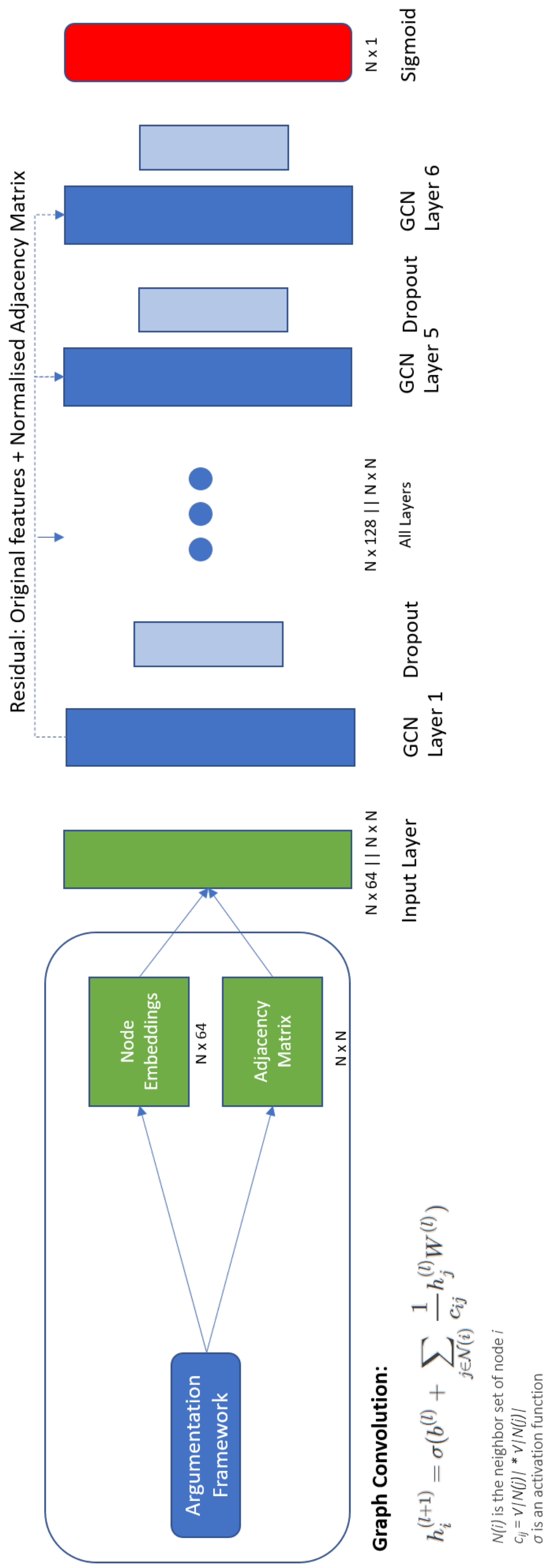

Residual: Original features + Normalised Adjacency Matrix
Argumentation Framework
Node Embeddings
N x 64
Adjacency Matrix
N x N
Input Layer
N x 64 || N x N
GCN Layer 1
Dropout
N x 128 || N x N
All Layers
GCN Layer 5
Dropout
GCN Layer 6
Sigmoid
N x 1
Graph Convolution:
$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} h_j^{(l)} W^{(l)})$
N(i) is the neighbor set of node i
$c_{ij} = \sqrt{|N(j)|} * \sqrt{|N(j)|}$
σ is an activation function

Figure 3.1: Overview of the GCN Architecture used for the experiments in the SAFA 2020 version.

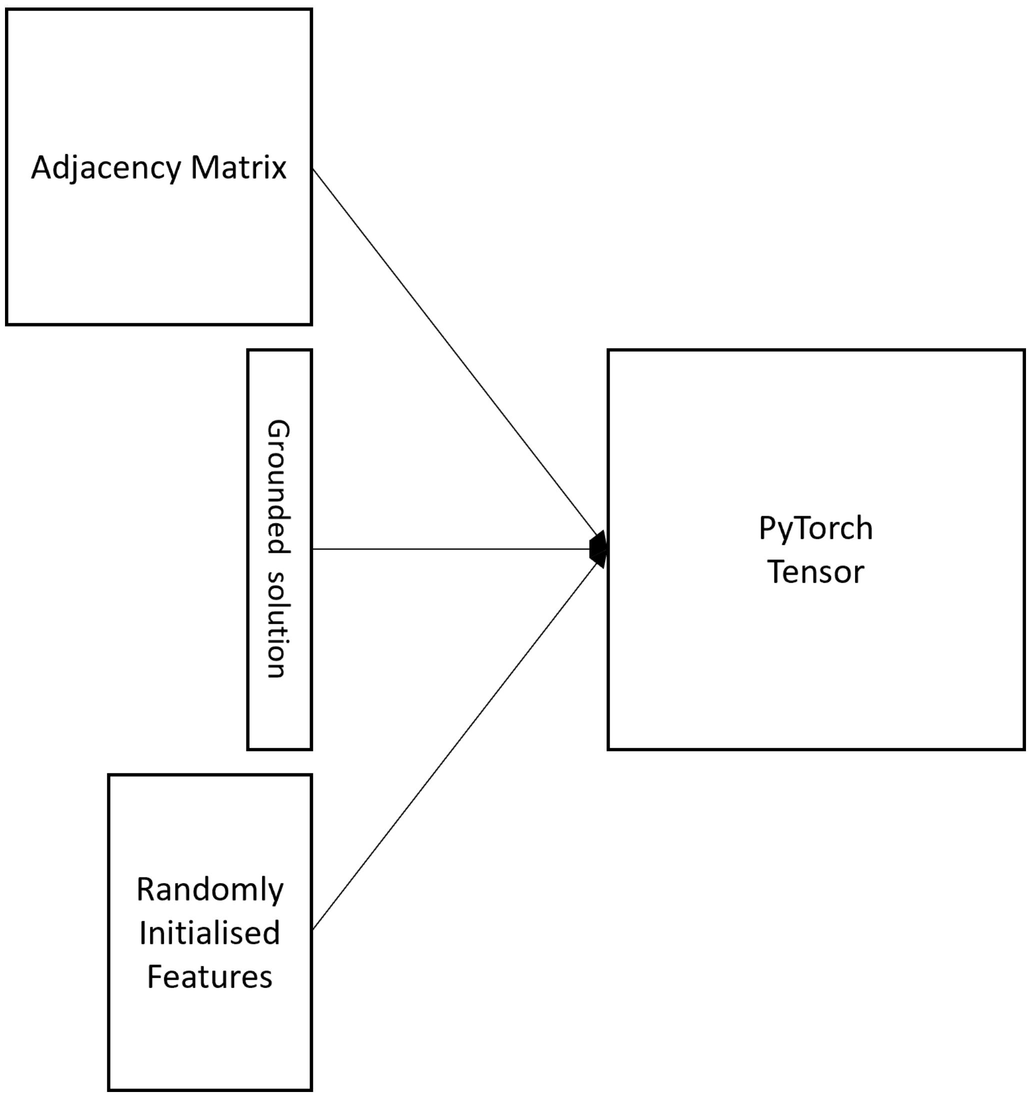


Figure 3.2: Overview of how input features are combined during training of the AFGCN solver

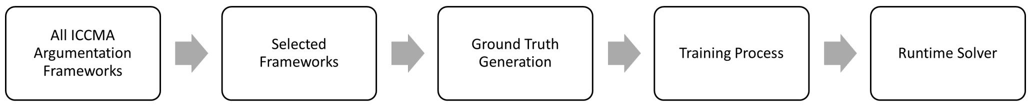


Figure 3.3: Overall process for data processing and training of the AFGCN solver

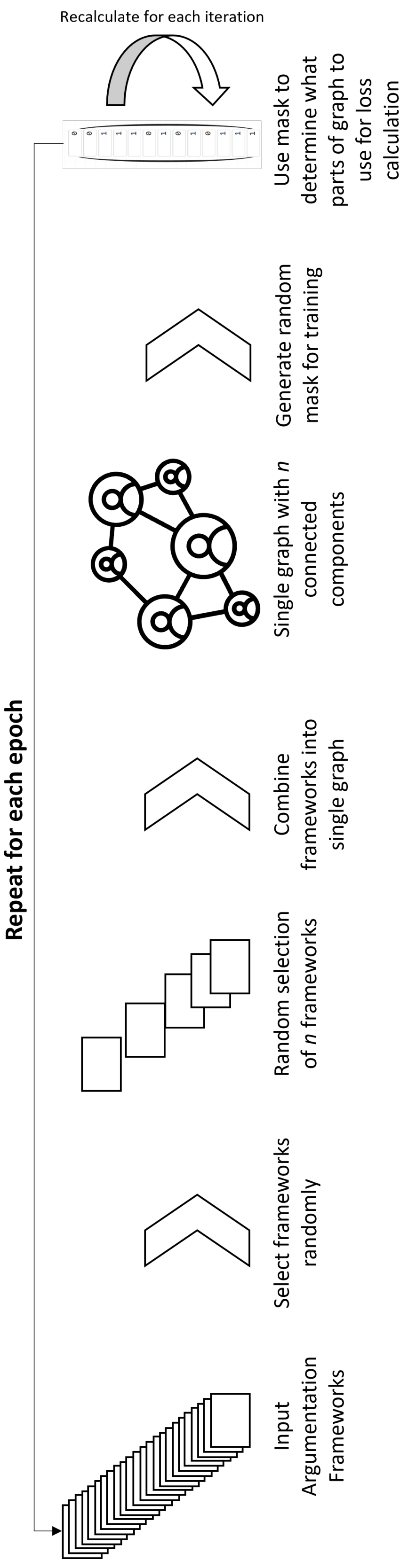

Repeat for each epoch
Input Argumentation Frameworks
Select frameworks randomly
Random selection of *n* frameworks
Combine frameworks into single graph
Single graph with *n* connected components
Generate random mask for training
Use mask to determine what parts of graph to use for loss calculation
Recalculate for each iteration

Figure 3.4: Overview of the process for generating random training batches.

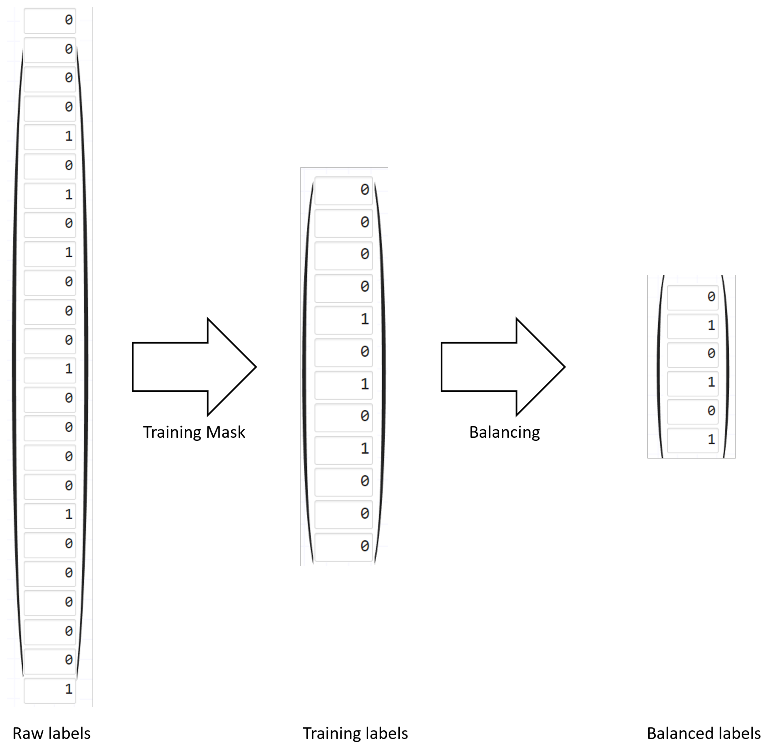
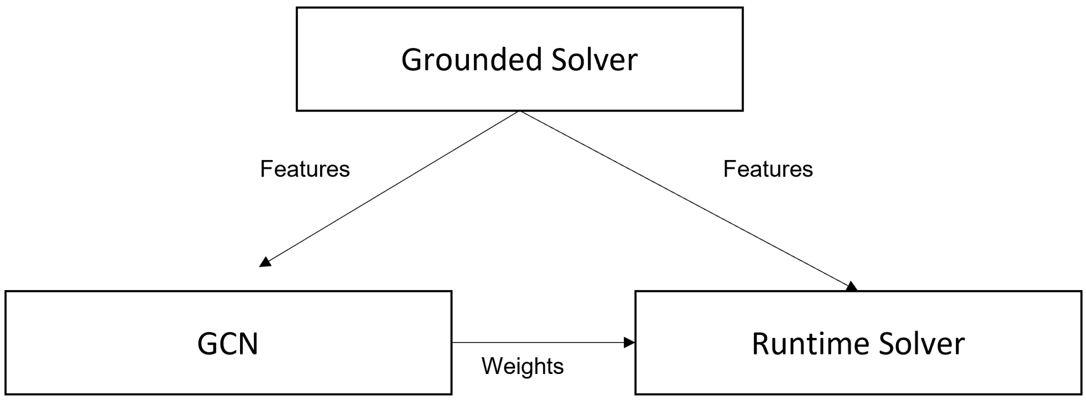
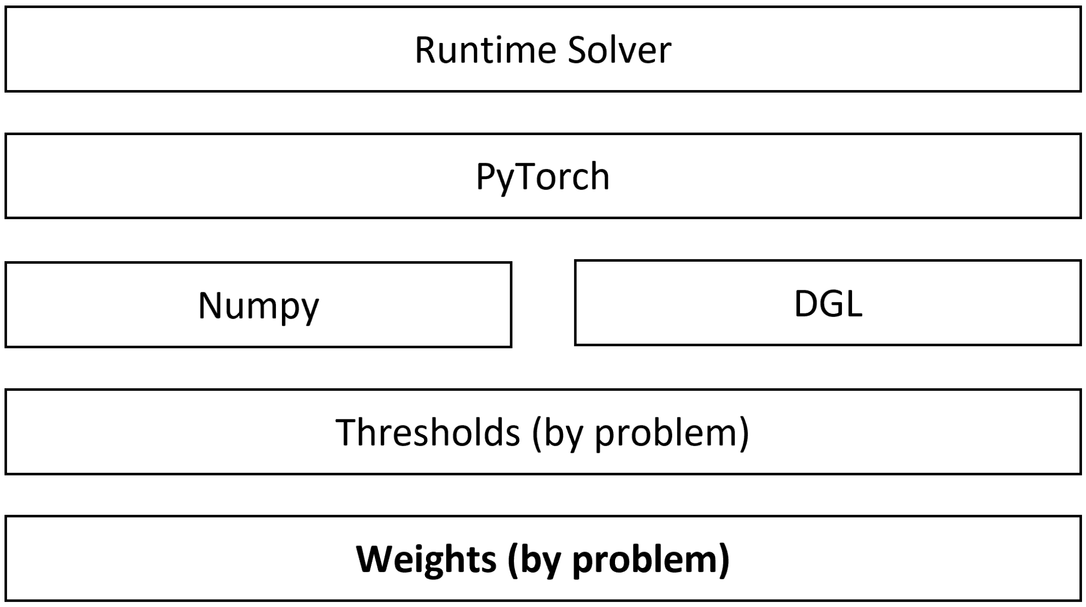


Figure 3.5: Illustration of the dynamic balancing process used by AFGCN

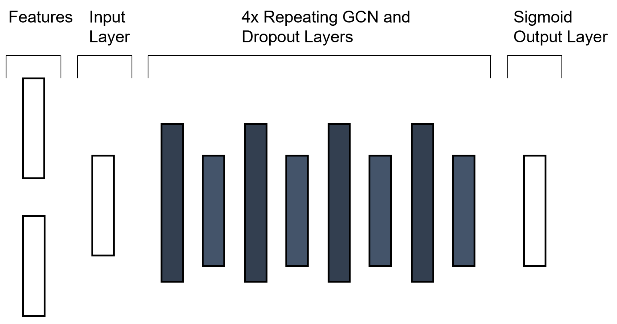


(a) Relationship between Runtime Solver, Grounded Solver and GCN



(b) Runtime solver system architecture

Figure 3.6: Runtime solver architecture for the solver used in the AFGCN experiments



Overall training is done using the ADAM optimizer with Binary Cross-Entropy loss as the loss function.

Training was done using a set of 538 argumentation frameworks from past ICCMA competitions, using cross-validation and a holdout set of 99 frameworks for testing.

One model was trained for each problem in the ICCMA competition.

Figure 3.7: Network architecture for the solver used in the results section for AFGCN

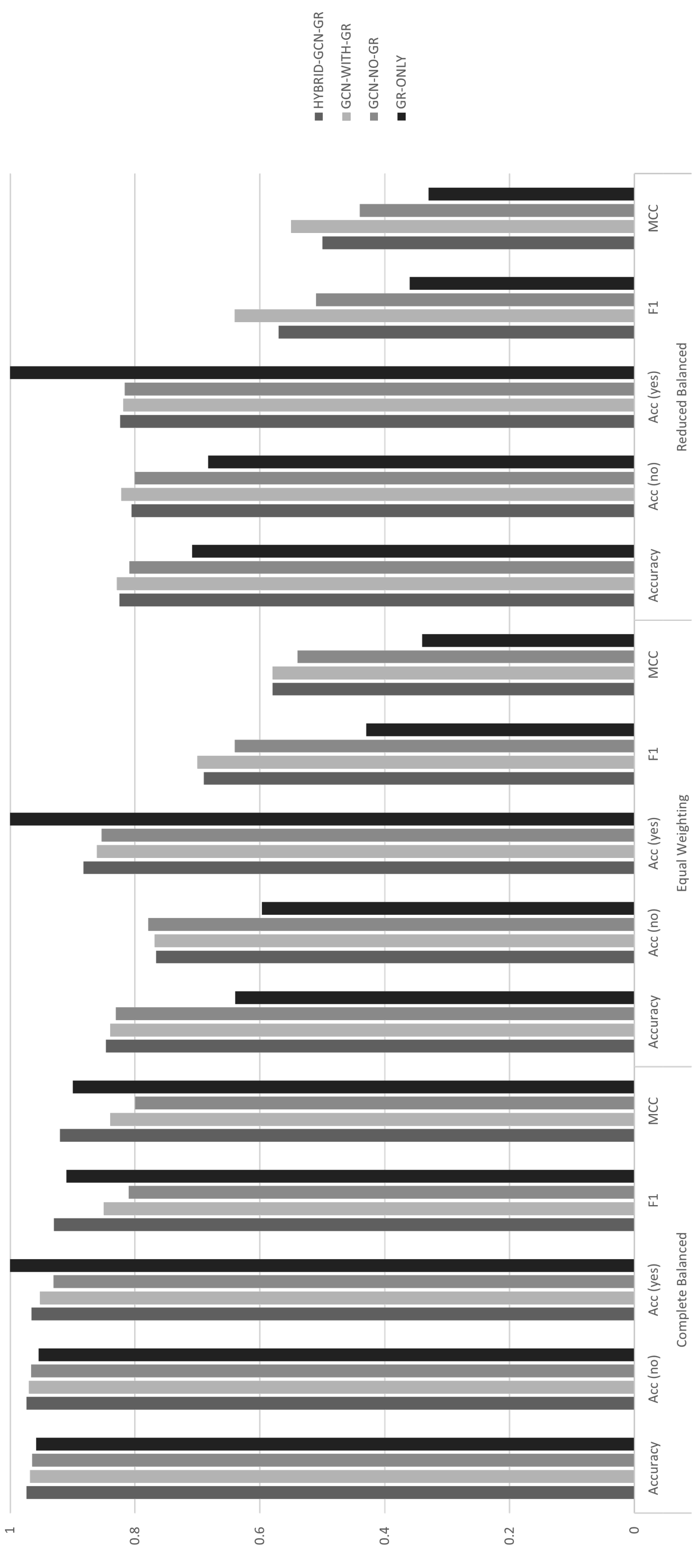


Figure 3.8: Overview of AFGCN approximation results for DC-PR

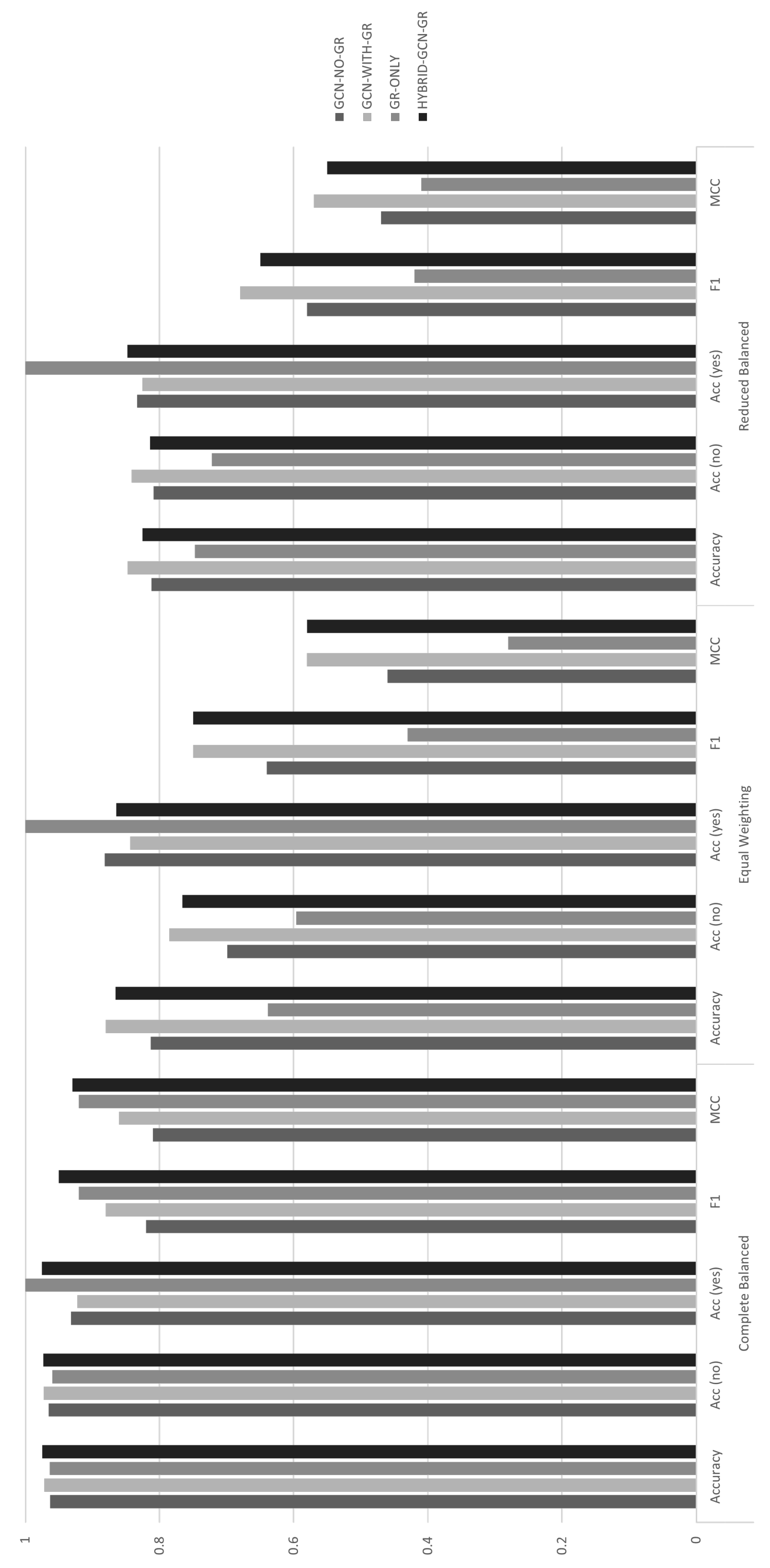


Figure 3.9: Overview of AFGCN approximation results for DC-CO

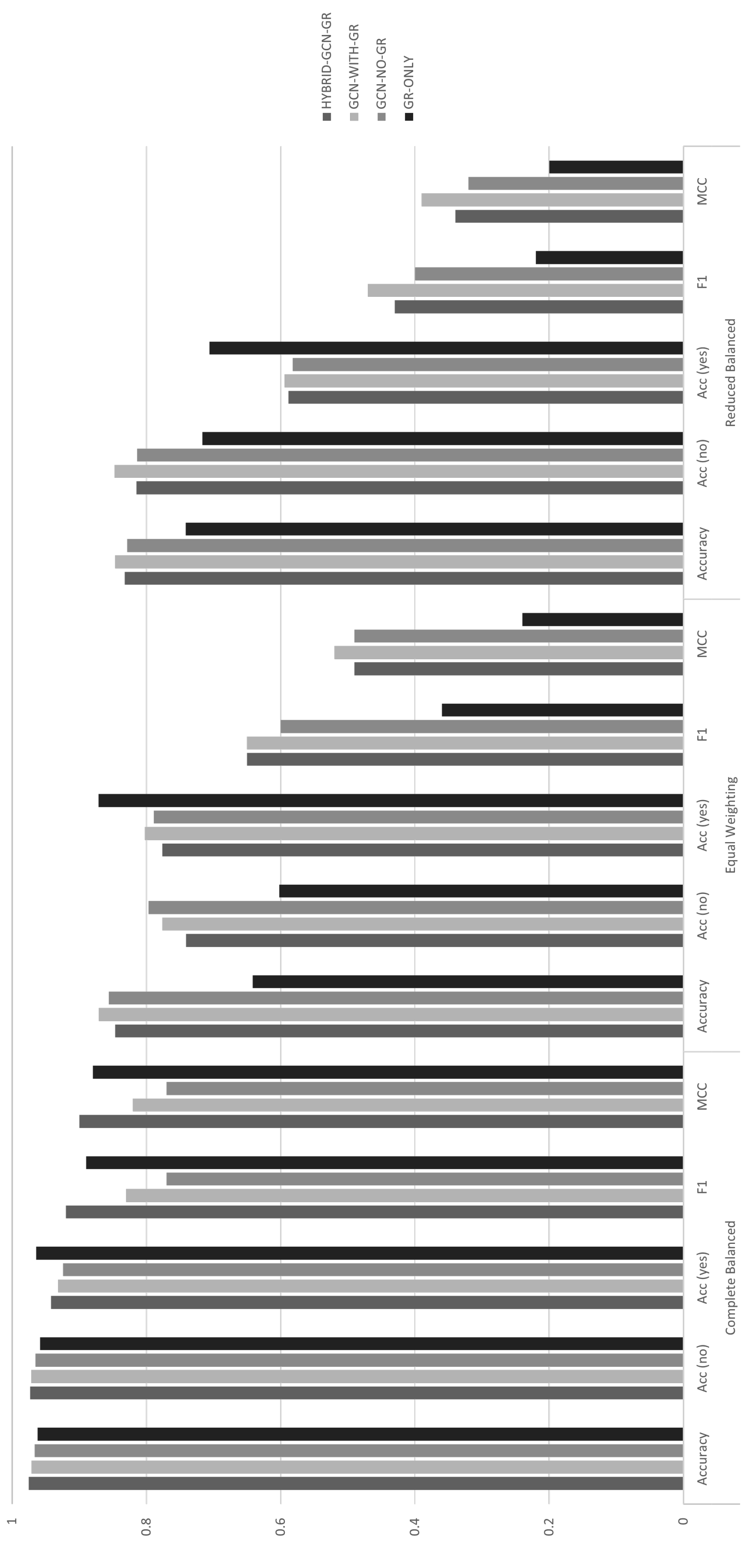


Figure 3.10: Overview of AFGCN approximation results for DC-ST

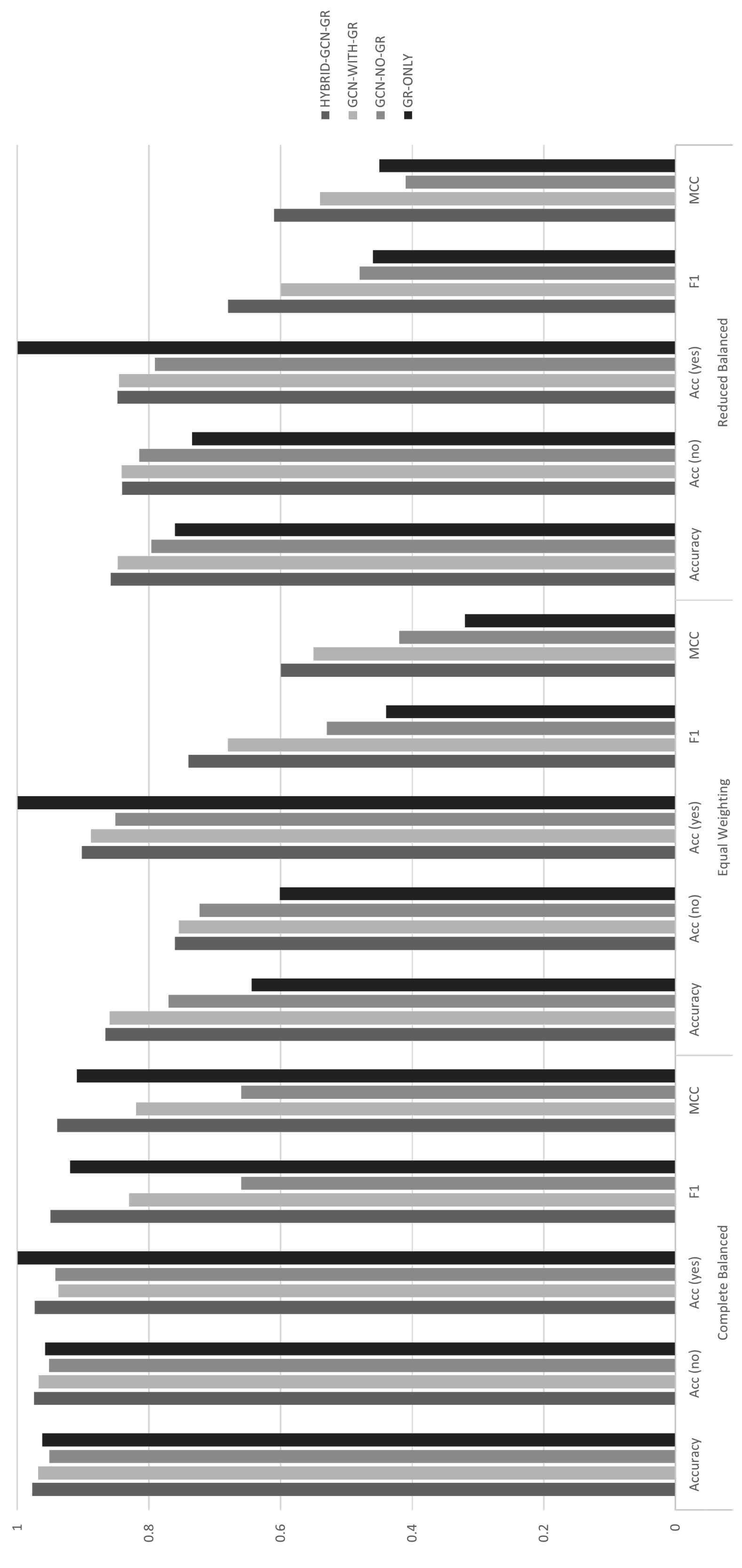


Figure 3.11: Overview of AFGCN approximation results for DC-SST

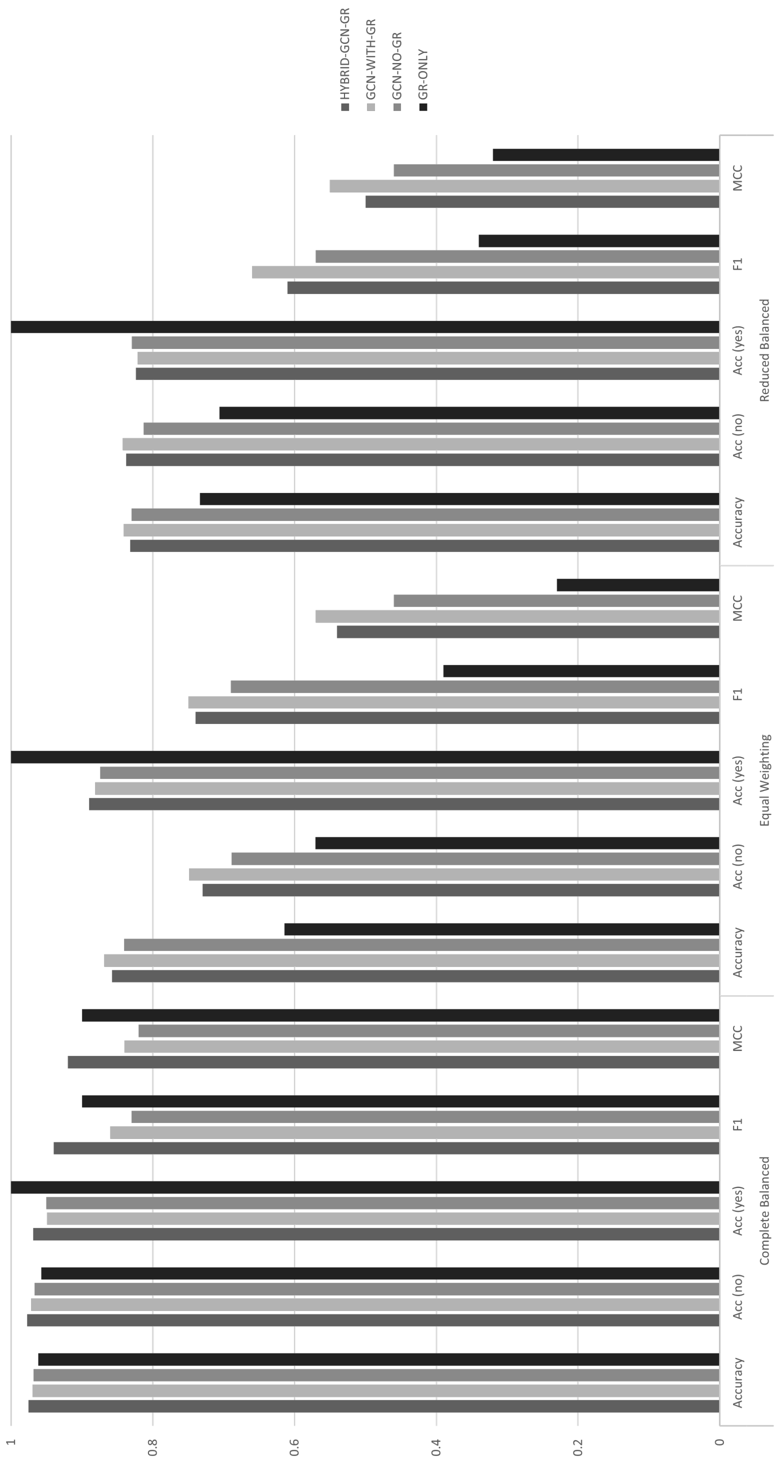


Figure 3.12: Overview of AFGCN approximation results for DC-STG

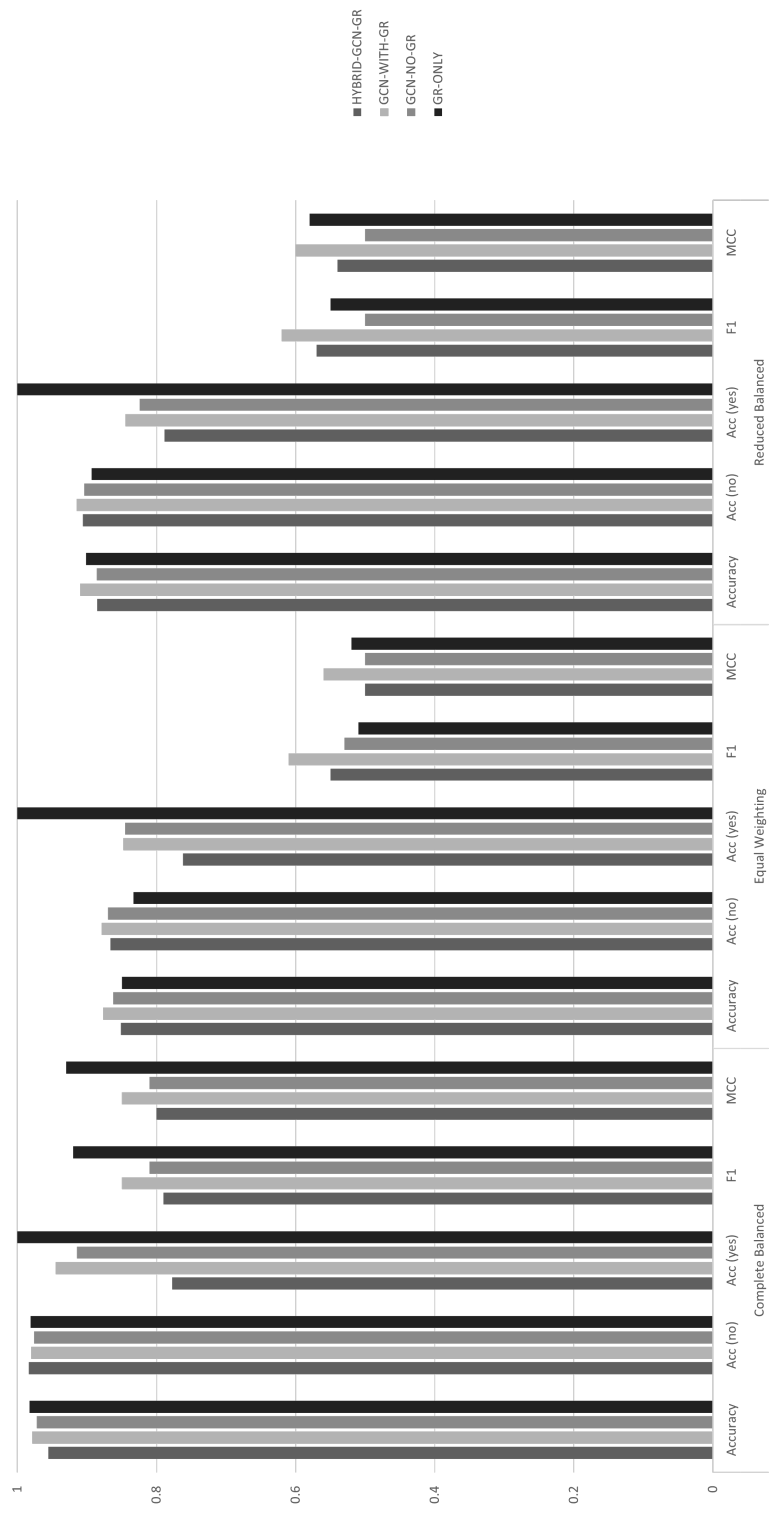


Figure 3.13: Overview of AFGCN approximation results for DS-PR

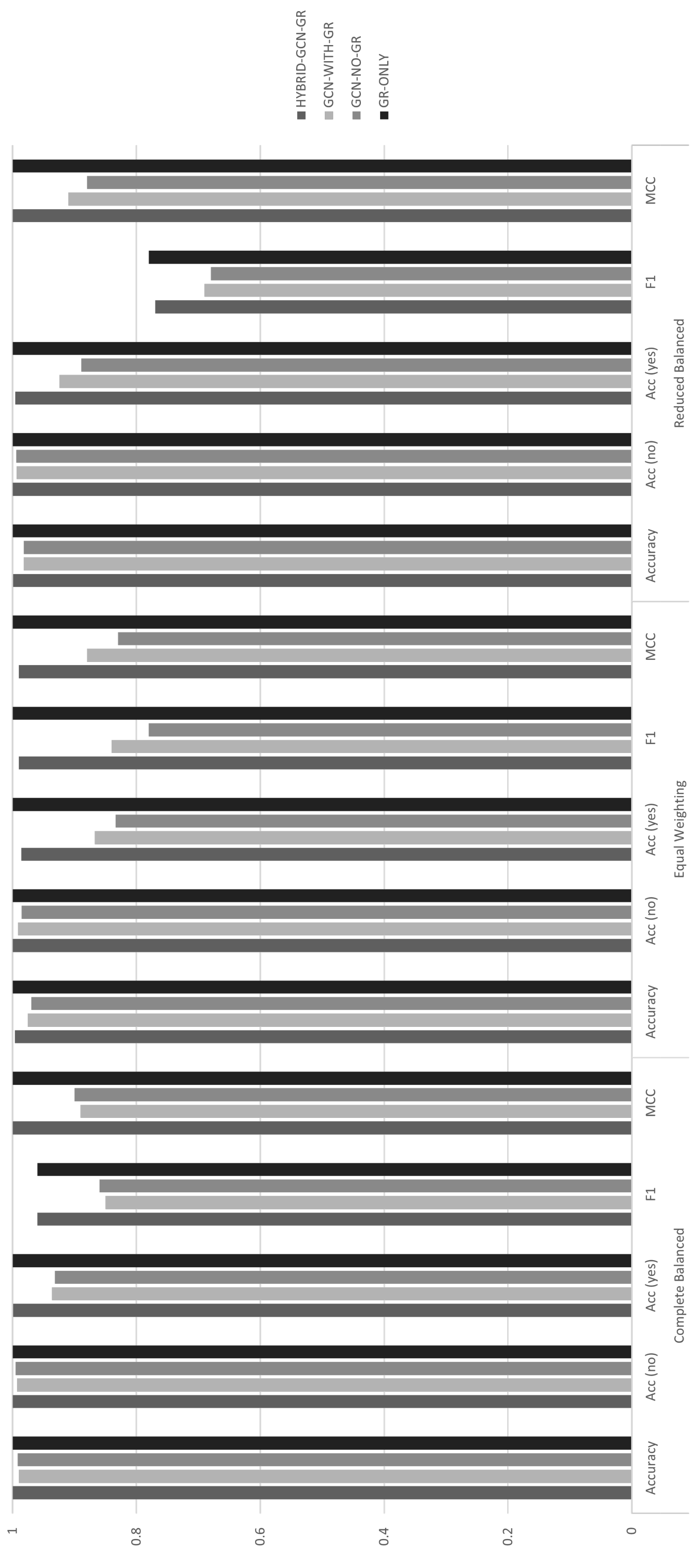


Figure 3.14: Overview of AFGCN approximation results for DS-CO

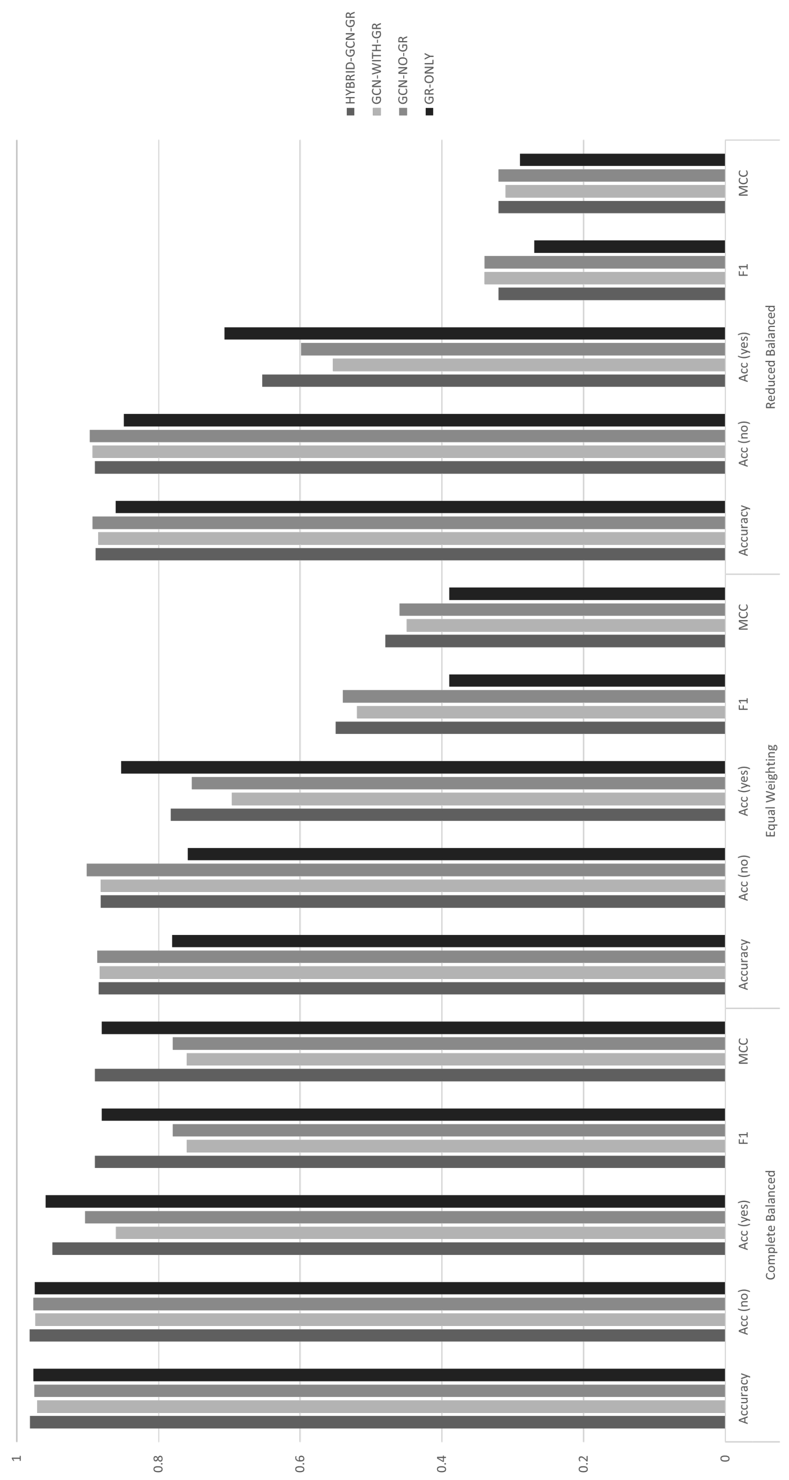


Figure 3.15: Overview of AFGCN approximation results for DS-ST

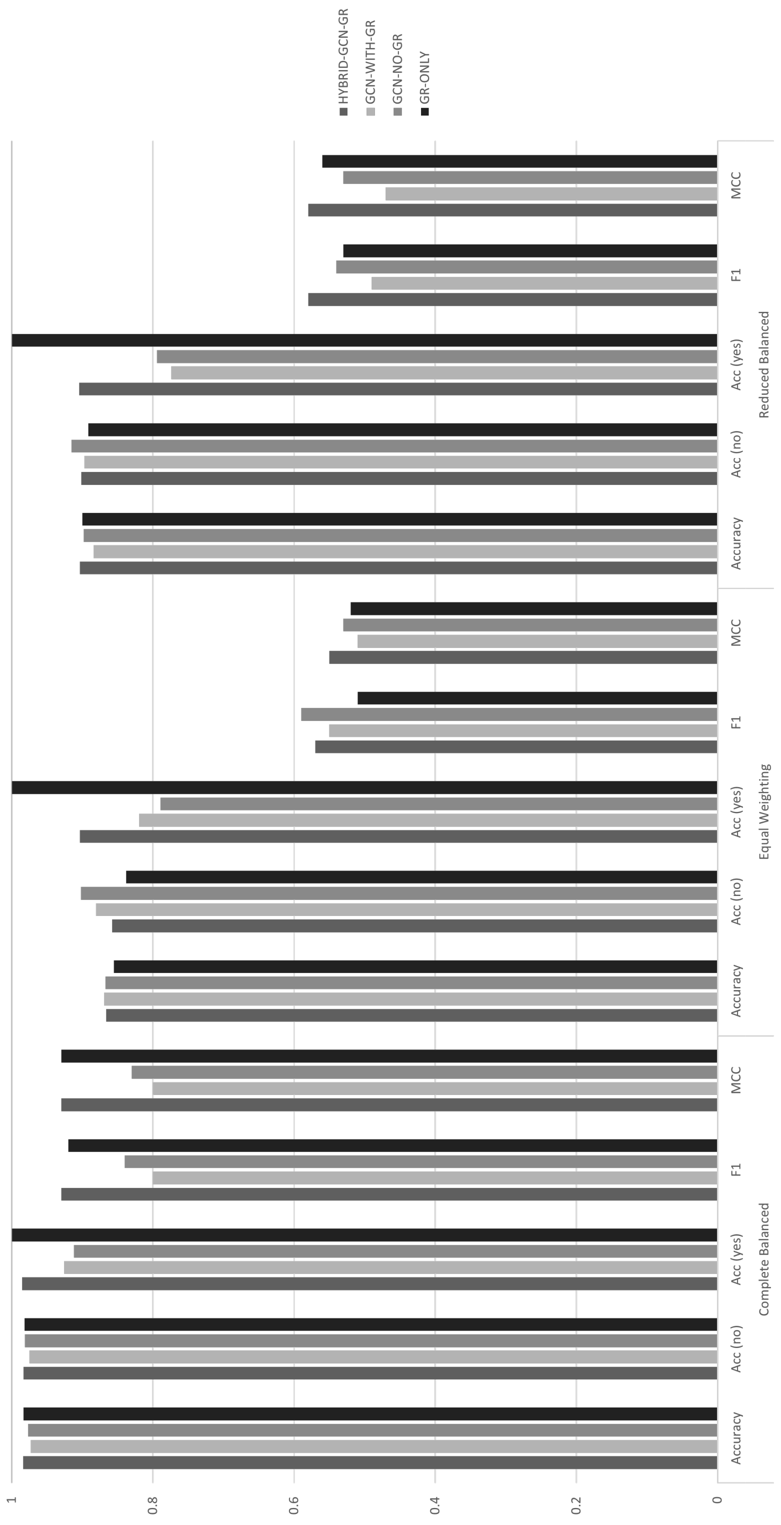


Figure 3.16: Overview of AFGCN approximation results for DS-SST

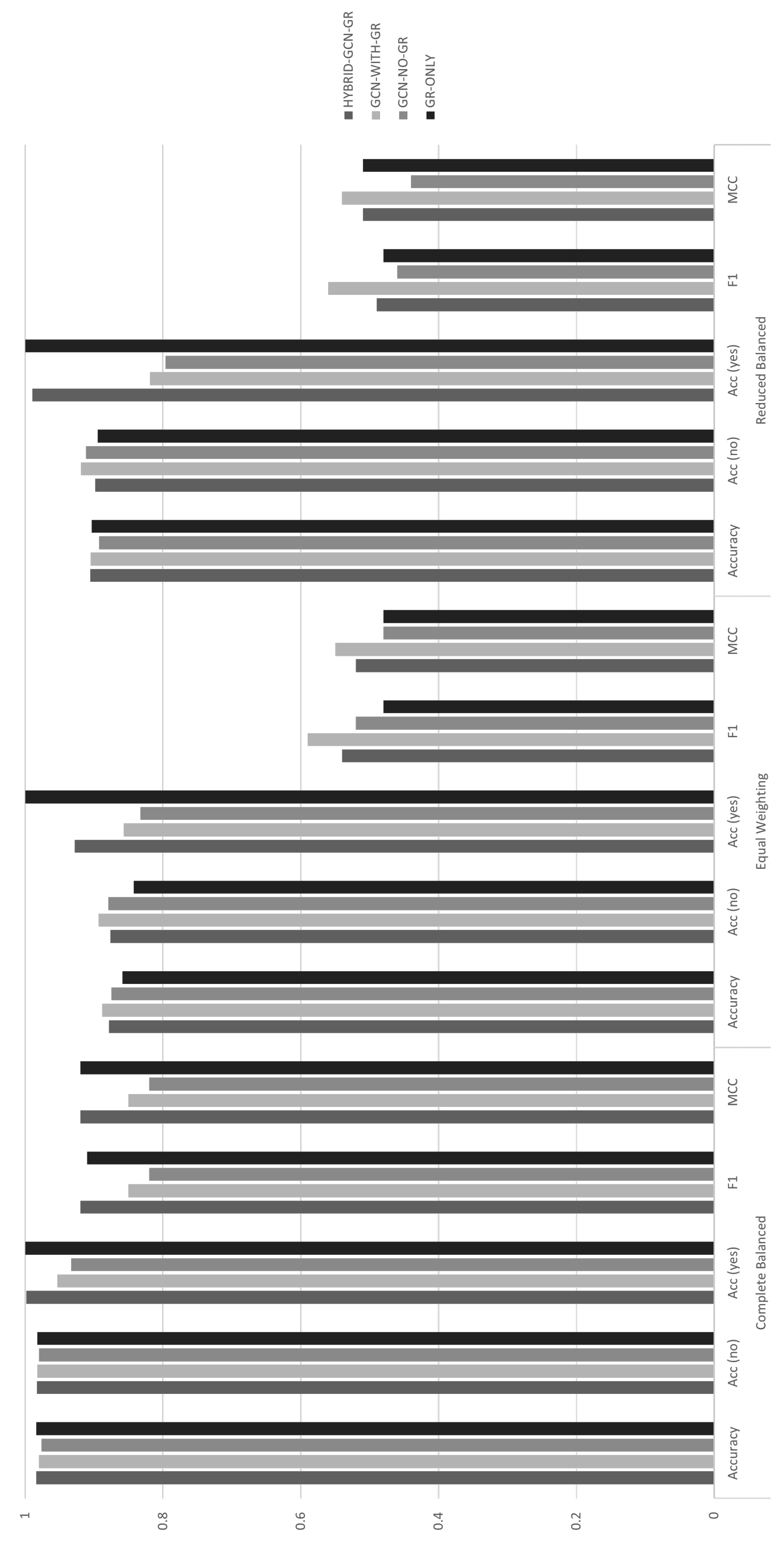


Figure 3.17: Overview of AFGCN approximation results for DS-STG

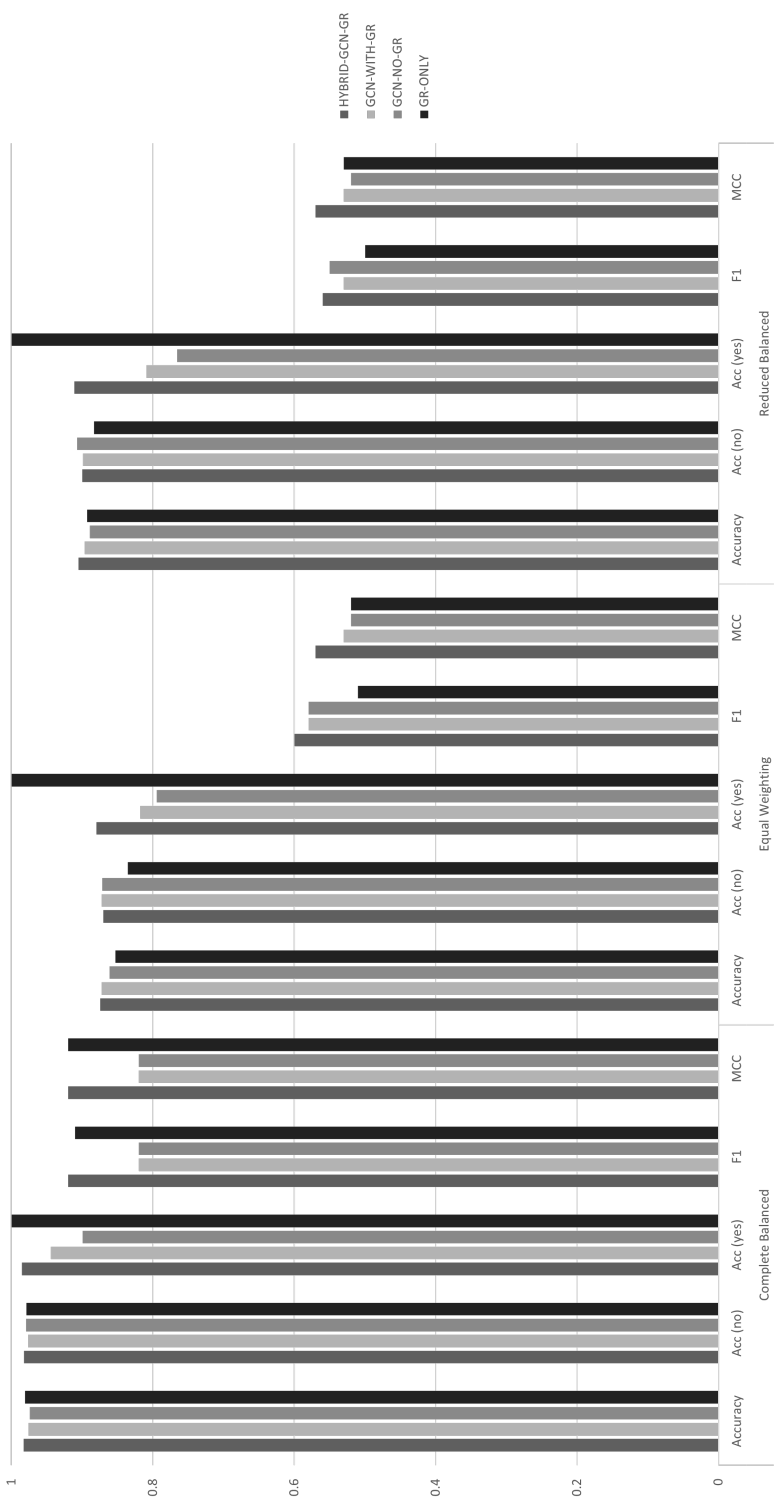
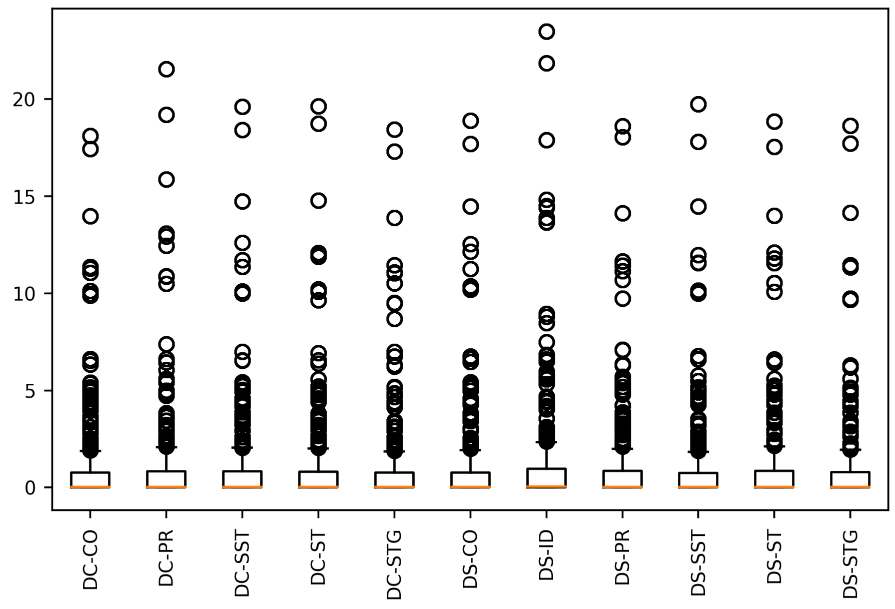
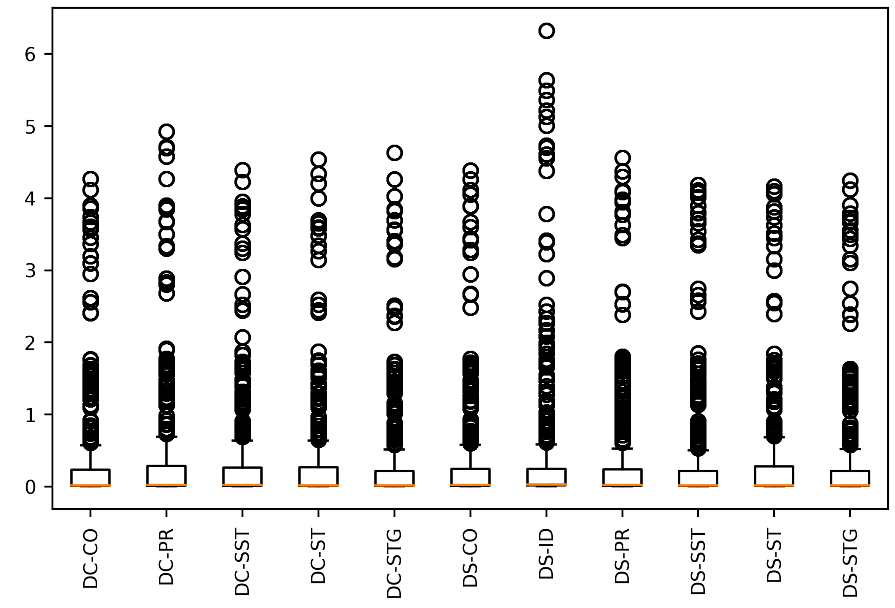


Figure 3.18: Overview of AFGCN approximation results for DS-ID

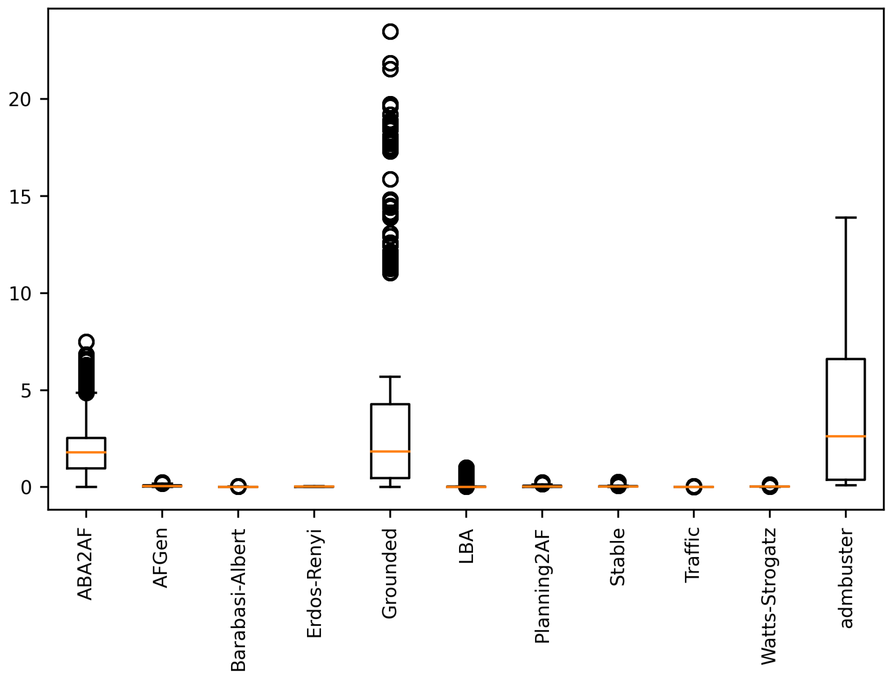


(a) w/GR

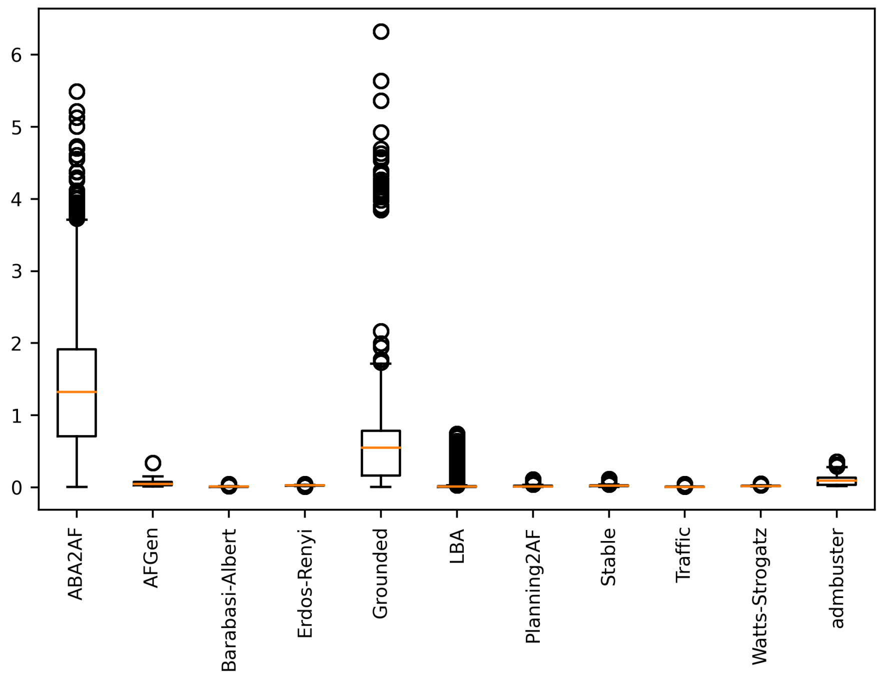


(b) No GR

Figure 3.19: Runtime distribution by semantics for the AFGCN solver experiments

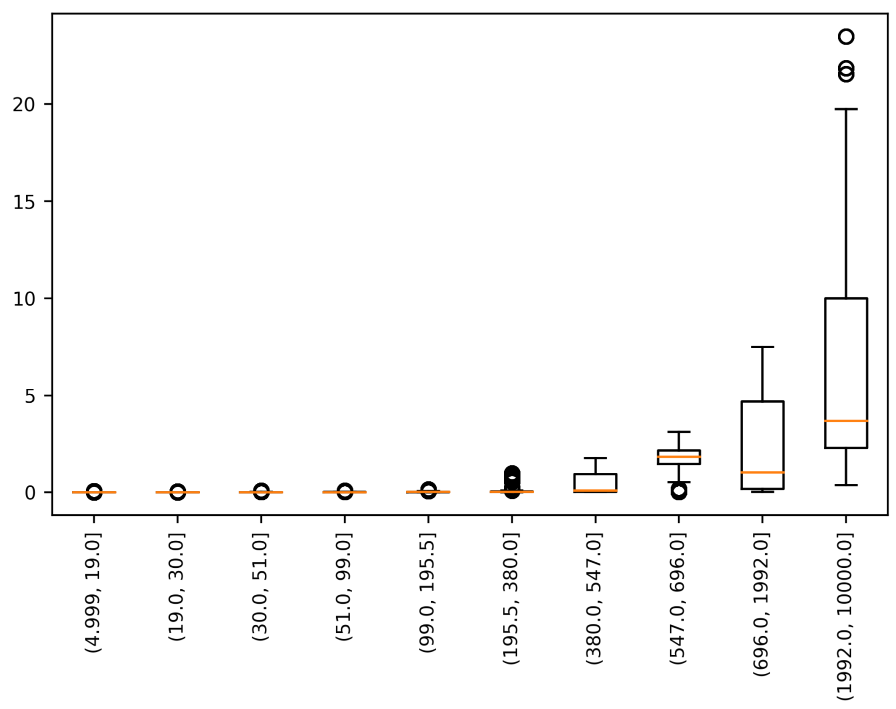


(a) w/GR

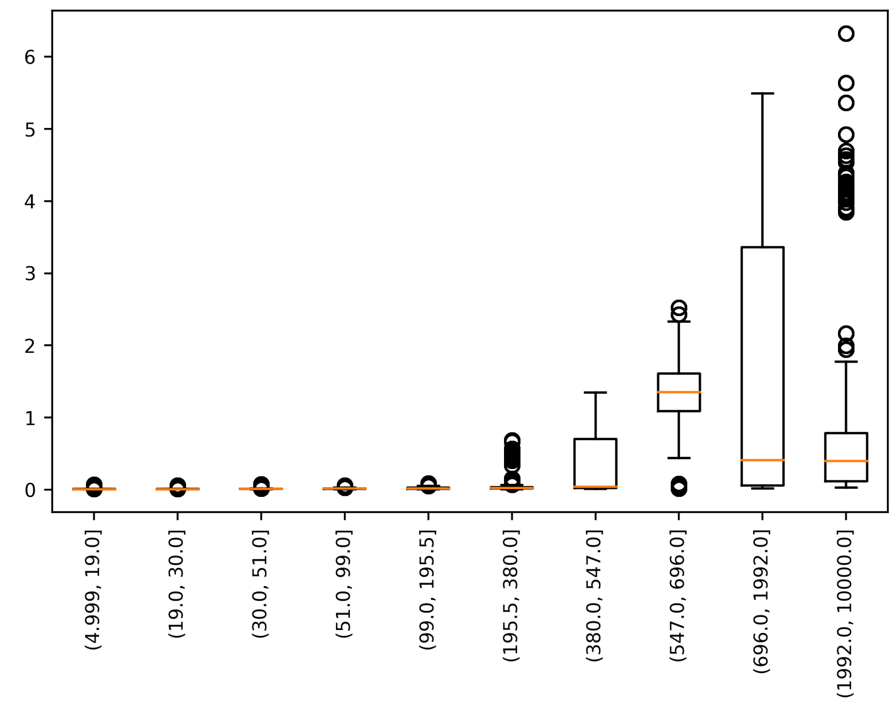


(b) No GR

Figure 3.20: Runtime distribution by benchmark for the AFGCN solver experiments



(a) w/GR



(b) No GR

Figure 3.21: Runtime distribution by size for the AFGCN solver experiments

Chapter 4

Using Co-Admissibility to Predict Admissible Sets and as a SAT Heuristic

We have noted previously in this thesis that most decision problems in the abstract argumentation space are NP-hard and do not allow for exact solutions in a tractable amount of time outside certain limited circumstances [32]. In Chapter 3, we outlined an approximate approach that can achieve good results in predicting the acceptability of arguments under different argumentation semantics.

This is an important application. However, there are other cases where one might be more interested in an admissible set that contains a given argument instead of just the acceptability status of the argument on its own. This includes all cases where a proof of correctness is needed for a solution or in general the solution should be explainable rather than simply probabilistically correct.

In this chapter, we develop an approach to adapt the GCN architecture from chapter 3 to predict co-admissibility rather than just the acceptability status of individual arguments. While that is not in and of itself the same as predicting admissible sets, it is straightforward to go from a set consisting of a chosen argument and all the arguments with which it is co-admissible to an admissible set by finding any admissible subset of the total set of arguments co-admissible with the chosen argument.

We then use predicted co-admissibility as an input to a SAT-based solution approach that bases itself closely on a winning solver from the ICCMA 2017 competition. We pass the predicted co-admissibility of arguments to a SAT solver by setting the phase saving array and demonstrate that this heuristic improves the performance of the solution approach relative to the baseline by solving an additional 6 out of 36 instances that were unsolvable by any solver at the ICCMA 2017 competition. This also allows us to give an answer to research question 5, "Can the output of an approximate solver be used as a heuristic to improve the performance of exact solution methods?"

The key contributions made within this chapter are as follows:

- Formalise the concept of co-admissibility between arguments.
- Extend our GCN architecture to be able to predict co-admissibility using a feature to indicate a particular input argument.
- Demonstrate that good approximations of co-admissibility can be achieved using a GCN architecture.
- Develop an algorithm for converting an approximate set of co-admissible arguments into a formally acceptable admissible set.
- Show how to utilise predicted co-admissibility as a heuristic in a SAT solver.
- Improve on an existing SAT based argumentation solution approach by using this heuristic to solve six additional hard argumentation problems, not solvable under competition rules without the heuristic.

First, we will develop the notion of co-admissibility and how we are going to predict it using a GCN-based approach. Then, we will sketch an algorithm that uses predicted co-admissibility as a search heuristics. After that, we will describe how to use the same information as a SAT-heuristic. Finally, we will present the experimental results from using this approach.

4.1 Predicting co-admissibility

We define co-admissibility as the property of an argument being jointly acceptable with another in one or more admissible sets. Taking all the arguments that are co-admissible with a given argument generates a set of arguments that are all pairwise compatible with the chosen input argument, but not necessarily with each other. It is therefore a different notion than standard admissibility, although it is related.

Formally, we define co-admissibility as follows:

Definition 4.1.1 An argument B is co-admissible with another argument A iff there exists an admissible set S such that $A \in S \wedge B \in S$

Note that while the pairs of arguments are jointly admissible as part of some admissible set, they are not necessarily so on their own. There may be a need to include other arguments in order to form an admissible set.

Co-admissibility is trivially symmetric as two co-admissible arguments are both part of at least one common set describing an admissible extension. It is also trivially reflexive as an argument is clearly co-admissible with itself. It is, however, not transitive as it is clearly feasible for two arguments to be members of the same extension A, but not of a different extension B.

The notion of co-admissibility adds a theoretical contribution to the analysis of abstract arguments by providing a new way to analyse their commonality. Co-admissibility is a fairly weak criterion for commonality between arguments but still powerful for the generation of admissible sets as the co-admissibility of two arguments guarantees that there is at least one admissible set containing them both.

Co-admissibility can be computed exactly only by enumerating all admissible sets of an argumentation framework, which as we have previously seen is an intractable problem. That means an approximate approach is once again warranted, but co-admissibility has the additional benefit of providing a partial bridge from an approximate to an exact approach.

This bridge is constituted by using the predicted co-admissibility probabilities generated by the approximate solution either directly in a local search for a minimal admissible set or by enhancing existing solution approaches with the information gleaned from the approximation as we will do below in our experiment with using these values as a SAT heuristic.

Co-admissibility, then, is useful in starting to bridge the gap between approximate and exact approaches. We will now proceed to look at how to reframe our GCN architecture to be able to predict this property.

4.1.1 Co-admissibility neural network training

Much of the training approach that we have described in chapter 3 remains relevant in the context of predicting co-admissibility. This includes in particular the dynamic generation of training batches, rather than relying on a fixed split between training and validation sets. The basic architectural parameters for the loss function, BCE, and the optimizer, Adam [50], are also identical.

However, co-admissibility as a problem does not suffer from the same level of class imbalance that is found in predicting acceptability because all arguments that are found together with the input argument will be positive cases, which makes up a larger proportion of the total, so the parts of the training approach that involved dynamic balancing and outlier detection do not form part of the training approach for co-admissibility.

The biggest difference is the pre-processing required to generate training data for co-admissibility. The target binary vector used for determining whether arguments are co-admissible or not is not fixed as was the case for acceptability. Instead, the values depend on the input argument (e.g. arguments co-admissible with A do not need to be co-admissible with B).

We pre-process all possible target vectors by enumerating ground truth solutions for our training set. All target vectors are loaded into memory and referenced dynamically at runtime. Each run of the training loop is trained with a single input argument mapping to a definite target vector containing co-admissible arguments.

The actual training process is carried out as in the previous model and there are no significant changes beyond those mentioned above.

4.1.2 Co-admissibility network architecture

We use the same basic GCN style architecture for the co-admissibility network that we previously used for predicting acceptability. While we follow the same basic pattern, we change and add a number of features to allow for inputting a context node in the training process and to accommodate the high memory requirements of the training dataset.

The core GCN architecture retains deep residual connections between layers, but drops input features other than the one indicating what argument is to be predicted. The randomised training regime described in Chapter 3 is also retained.

The core components of the GCN architecture used for co-admissibility includes the following elements:

1. An input feature containing the value 1 for the context node and 0 otherwise.

2. An input layer receiving this input and the normalized adjacency matrix.

3. 4 repeating blocks of a GCN layer [51] and a Dropout layer [89]. We experimentally found that this model performs better with higher dropout values, so this has been increased to 0.7 from 0.5 in the previous model.

4. Residual connections feeding the original features and the normalised adjacency matrix as additional input at each block.

5. A Sigmoid output layer generating an estimate for the co-admissibility of each argument with the input argument.

The model was trained using Adam [50] with Binary Cross-Entropy (BCE) as the loss function. Training data is loaded dynamically from RAM to the GPU at the start of each epoch. See figure 4.1.

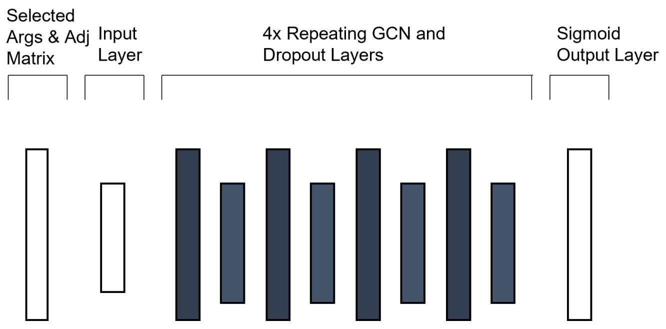

Selected Args & Adj Matrix
Input Layer
4x Repeating GCN and Dropout Layers
Sigmoid Output Layer

Figure 4.1: Network architecture for co-admissibility prediction

4.1.3 Finding an admissible set from a set of predicted co-admissible arguments

The set of predicted co-admissible arguments will not in general constitute an admissible set in its own right. Nor is it particularly likely that any co-admissible argument on its own will form an admissible set. It is, therefore, necessary to perform a search, using some method, to generate an admissible subset from the co-admissibility predictions.

There are many potential search techniques one could utilise to generate an admissible set. Here, we will use an algorithm modelled on Breadth First Search (BFS) that incorporates the probabilities as priorities to guide the search. Admissibility can be tested using a standard method in polynomial time and conflict-freeness follows directly from the definition of admissibility.

While this algorithm would require optimization in terms of memory complexity to work well in practice, it does guarantee that the smallest possible admissible set is returned.

Algorithm 1 Algorithm for finding admissible subsets using co-admissibility predictions as a heuristic

```
1: Input: Sorted list of probabilities $AF$ in descending order, starting node $a$, context
   node $c$
2: Create set $context$
3: $context$.append($c$)
4: Create queue $Q$
5: $Q$.append($a, context$)
6: while $Q$ is not empty do
7:   Let $u, context = Q$.pop()
8:   $context$.append($u$)
9:   if $context$ is admissible then
10:    Return context
11:  end if
     {Ordered so we try most probable arguments first}
12:  for each member $v$ of $AF$ not in $context$ do
13:    if $v \cup context$ is conflict-free then
14:      $Q$.append($v, context$)
15:    end if
16:  end for
17: end while
18: Return False
```

This means no additional computation is carried out after the first positive example is found.

4.2 Using co-admissibility as a SAT heuristic

We have discussed previously, how SAT solvers can be used to solve problems in abstract argumentation. As noted, many of the most successful approaches in the literature have been based on a reduction to SAT.

It is, therefore, highly relevant to look at ways information from approximate approaches can be used to inform and improve SAT based approaches. The most obvious way to do this is via the incorporation of approximate solutions as heuristics to guide an exact search. This once again helps bridge the gap between approximate and exact approaches.

SAT heuristics come in different forms [20, 73]. Decision heuristics, for instance, focus on what variable to select to branch on at a given time, while a polarity selection heuristic selects the value to impart to that variable. In this chapter, we create a polarity selection heuristic based on the predicted co-admissibility values from the neural net.

The way we implement this is by injecting the predicted truth values from the co-admissibility GCN into the phase saving array of the SAT solver [81], the data structure used to determine what branch, true or false, the SAT solver will follow first.

That is to say, all values beyond a certain probability threshold (0.7 in the best experiments) are marked as true in the phase saving array and the rest are marked as false. This means the SAT solver will first try to branch with the polarity that has been predicted by the co-admissibility GCN.

The encoding we use for finding admissible sets is taken directly from the ArgSemSat solver [18], which performed best for preferred semantics at ICCMA 17, and we deploy the same version of the Glucose SAT solver used by this application. In addition, we use python with pytorch and the PySAT library (see www.pytorch.com) to orchestrate the process of getting predictions and feeding them to the phase saving array.

4.3 Experimental results

Here we present the experimental results from predicting co-admissibility and using it as a SAT heuristic.

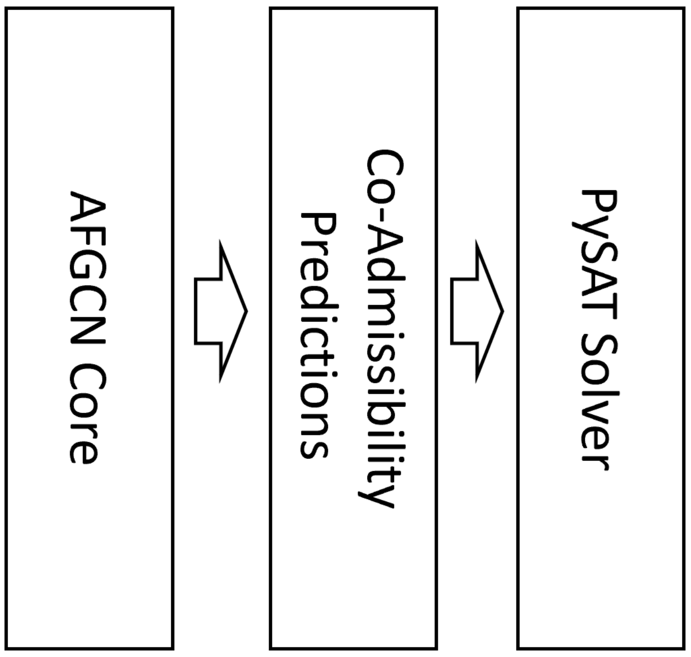


Figure 4.2: Solver architecture for predicting co-admissible sets

4.3.1 Predicting co-admissibility

In this section, we will discuss the experimental setup and results from applying the approach discussed above to predicting co-admissibility with a GCN-based architecture and a randomised training approach.

4.3.1.1 Experimental setup

We trained our models on a dataset of 792 graphs taken from past ICCMA competitions, the same dataset that was used for some of the experiments in chapter 3. The graphs range in size from 2 to 100,000 arguments. The test set consists of 99 graphs constructed to contain a fairly even split of graphs between the benchmarks present at the ICCMA 19 competition.

We generated ground truth using the exact pyglaf [4] solver and wrote a custom python script to pre-process the solutions into target vectors suitable for training. The training was done on a K80 GPU running in the cloud, using pytorch to implement the training loop. The threshold value was optimised using a manual hyperparameter search. This is shown in figure 4.3.

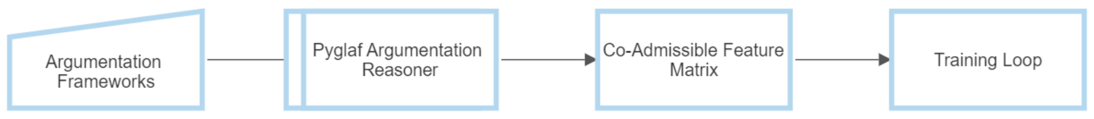


Figure 4.3: Processing pipeline for co-admissibility prediction

4.3.1.2 Results

The results for predicting co-admissibility are shown in Table 4.1. Overall, they are comparable to the results for predicting acceptability, but does not quite reach the level of the AFGCN solver reported in chapter 3. Importantly, both positive and negative accuracy is better than chance, which enables its heuristic use in the SAT example below.

We compare these results to the vanilla performance of the best performing AFGCN model for the preferred semantic. Unsurprisingly, this does not perform particularly well, demonstrating the value of the adaptations, we have described above.

Table 4.1: Overview of results from predicting co-admissibility with adapted AFGCN

Model	**Accuracy**		
	Overall	**Yes**	**No**
Co-Admissibility GCN	93%	59%	94%
Unadapted AFGCN	61%	7%	81%

4.3.2 Co-admissibility as a SAT heuristic

In this section, we will discuss the experimental setup and results from applying the approach discussed above to a SAT based solution approach from the ICCMA 17 competition.

4.3.2.1 Experimental setup

The experiment was based on the results from the ICCMA 17 competition. Based on the detailed results files, describing the problems set for the competition, we recreated the complete set of 310 instances given to the solvers in the DC-PR track.

This set included 36 tasks that no solver managed to solve within the time limit during the competition. Our solver replicated as closely as possible the setup for the best performing solver, ArgSemSAT, using the same underlying SAT solver and SAT encoding and a runtime environment closely matching the one specified by the competition.

We ran our solver against the full set of problems to ensure that we had replicated as closely as possible the setup for the original competition. Although it is not possible to completely eliminate all variability, we have taken the possible steps to make sure that it is kept at a minimum.

4.3.2.2 Results

Our results on the DC-PR task improve on the results from the ICCMA 2017 competition by solving an additional six hard instances that were not solved by any solver at the competition. The results from the competition with our approach added is shown in table 4.2. This table shows the actual performance on the results set from the ICCMA 2017 competition and our results from the recreation.

Table 4.2: Overview of using co-admissibility as a SAT heuristic to guide the solution of abstract argumentation problems

Solver	**Time out**	**Wrong**	**Correct**
AFGCN-SAT (Ours)	40	0	310
ArgSemSAT	46	0	304
argmat-sat	47	0	303
cegartix	51	0	299
pyglaf	53	0	297
goDIAMOND	62	0	288
CoQuiAAS	64	0	286
ArgTools	68	0	282
argmat-dvisat	102	0	248
conarg	117	0	233
argmat-mpg	122	0	228
EqArgSolver	202	0	148
heureka	208	0	142
gg-sts	90	161	99

The graphical representation of the results in figure 4.4 shows our solver following the same pattern that is demonstrated by the progression of the best performing solvers in the competition. That is to say, we make an incremental amount of improvement by solving a few additional instances.

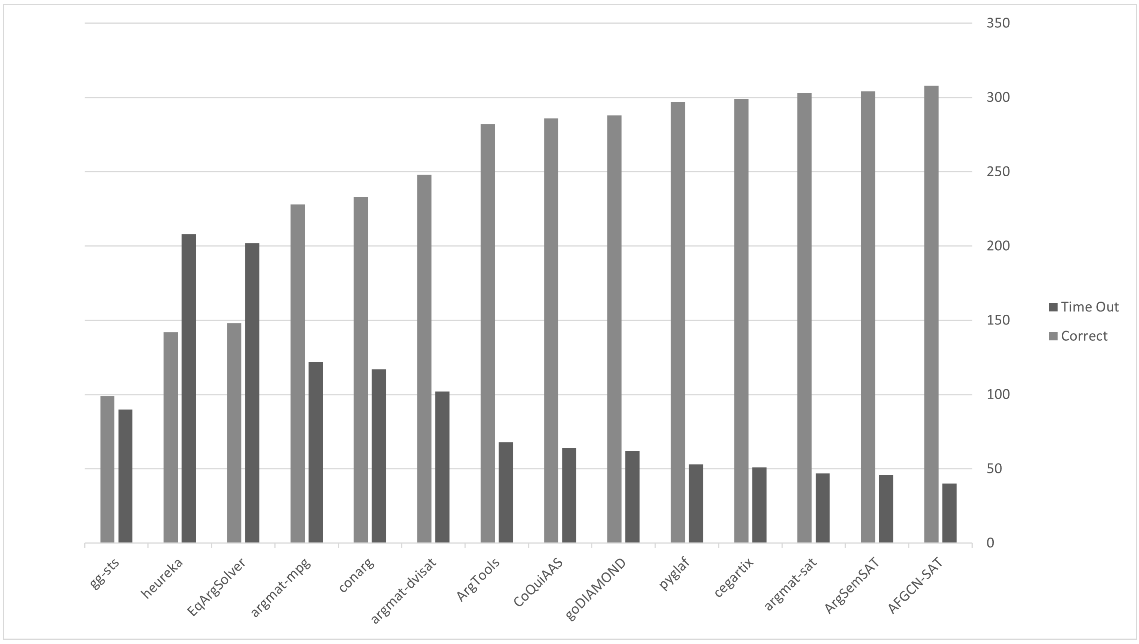


Figure 4.4: Results for the co-admissibility driven solver on ICCMA 2017 test cases

Overall, the results are promising in that we are able to improve on an existing approach by using a polarity heuristic injected via the phase saving array. While there isn't a radical improvement, the improvement comes from difficult instances, which no solver was able to solve under competition rules at the time of the competition. At the very least, we can point to the value of further research into generating SAT heuristics via deep learning approaches in the abstract argumentation space.

4.4 Summary

The experiments in this chapter sketch out the beginning of a way to bridge between approximate and exact methods for solving problems in abstract argumentation. This will potentially enable the best of both worlds, where approximate solutions can be generated quickly, but also allow them to be turned into exact solutions, when the application merits the cost of doing so.

The results of predicting co-admissibility show that it is feasible to predict arguments that are valid as a set and not just the individual status of arguments. This can be taken in different directions either to improve on the accuracy of the existing approach or to adapt it to predict directly admissible sets instead of going through the abstraction of co-admissibility.

The answer to our research question then is a qualified yes. It is in certain cases possible to improve the performance of an exact approach with the information from the approximate solver. We have, however, not systematically addressed the extent to which this is possible and more research is needed to see how far it might be taken.

The algorithm described for going from co-admissibility to admissible sets is another starting point for making progress. While there is much work to be done in finding an optimal algorithm for the use case, we have at least demonstrated that it is feasible to go from approximate to exact solution utilising the information gained from the GCN predictions.

The incorporation of the predictions into a SAT based approach gives another avenue of exploration with promising results. If we can use approximate solutions to guide exact ones, whether using SAT or another solution approach, that may enable us to tackle problem instances of a complexity beyond what is possible with current tools. This could be done both as an initial step, which is what we've experimented with in this chapter, but it may also be possible to strategically refresh the heuristic at particular points during the SAT search to heighten the effect of the heuristic on the search.

In the next c hapter, we will move on to another application of the basic GCN architecture, by examining what happens when we visualise its internals.

Chapter 5

Visualising Argumentation Graphs with Graph Embeddings and t-SNE

This chapter examines whether using graph embeddings [40] with a dimensionality reducing algorithm, t-SNE [61], to visualise argumentation graphs and their structural properties through unsupervised learning can lead to interesting results.

This approach has been effective at data visualisation in Deep Neural Network research for graph-based methods [79]. The contribution of this chapter is to show that this technique also holds promise in the visualisation of argumentation graphs by applying it at both the node and the graph level using both a standard and a custom built embedding approach.

In particular, we show that it is possible to clearly visualise the functional partitions of arguments in the Sembuster domain [86] using this method and to separate argumentation domains into visual clusters at the graph level using a custom GCN embedding, raising the possibility that both differences between argument graphs and the function of arguments within argumentation graphs can be clustered and shown in a visually intuitive way using unsupervised methods.

We also explore the visualisation of the internal state of the approximate GCN solver developed in Chapter 3 in order to gain additional information about approximation performance in line with research question 6, “Does visualising the structure of the GCN model used for approximation give us additional insight into the way it functions?”

First, we proceed to discuss how to visualize graph embeddings with t-SNE and why this approach was considered over UMAP, a common alternative. Second, we disucss how to visualize argumentation graphs at the node level. We follow this with a discussion of graph-level visualization. Finally, we use the approach to visualize embeddings taken from the models developed in chapter 3.

5.1 Visualising embeddings using t-SNE

Embeddings are usually of low dimensionality, but even so, they are not easy to visualise. However, once an embedding has been generated for complex data types such as graphs, words, or images, it becomes possible to use dimensionality reduction techniques to visualise them effectively. As an exception, we will cover the relevant background material here as it is only used in this chapter and the presentation of the background is very tightly linked to the rest of the presentation.

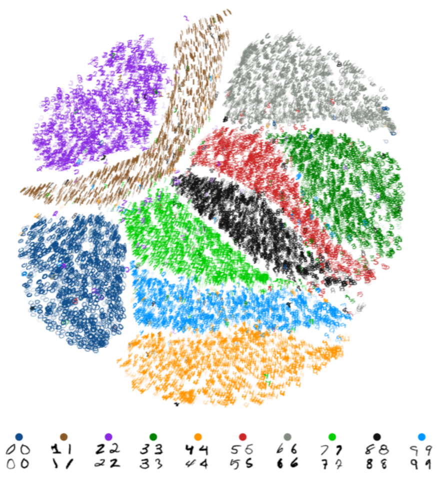
In the Deep Neural Network community, the dimensionality reduction technique of choice is t-distributed stochastic neighbor embedding (t-SNE). In contrast to other common dimensionality reduction techniques such as Principal Component Analysis (PCA), t-SNE does not rely on a linear projection but uses local relationships between points. It uses Student's t-distribution to model the relationship between points in the higher dimensional space and then recreates those relationships in the lower dimensional space by way of a gradient descent based algorithm [61].

UMAP (Uniform Manifold Approximation and Projection) is a dimensionality reduction technique that is similar to t-SNE (t-distributed Stochastic Neighbor Embedding). Both algorithms are designed to reduce the dimensionality of high-dimensional data while retaining as much of its structure as possible [69].

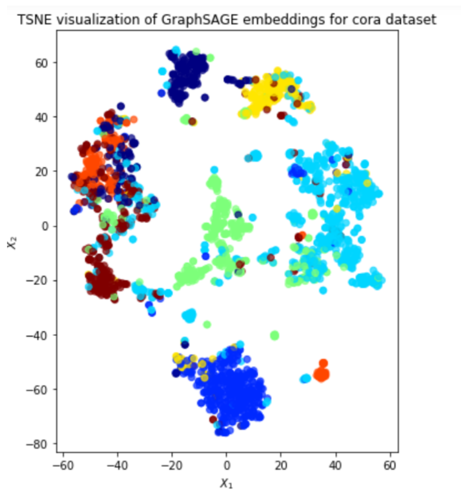
However, UMAP has several advantages over t-SNE. First, UMAP is much faster than t-SNE and can be applied to larger datasets. Second, UMAP is more efficient than t-SNE in terms of memory usage. Third, UMAP is more robust to changes in the data, meaning it performs better when the data changes over time. Finally, UMAP is better able to preserve the global structure of the data, while t-SNE tends to focus more on local structure.

Despite these advantages, t-SNE still has its uses. Since t-SNE is better at preserving local structure than UMAP, it is well-suited for tasks that require good visualization of clusters of nearby points, such as identifying groups in a dataset. That advantage outweighs the other advantages of UMAP in this case, as we'll see that clustering is essential to some of the visualization results we will explore.

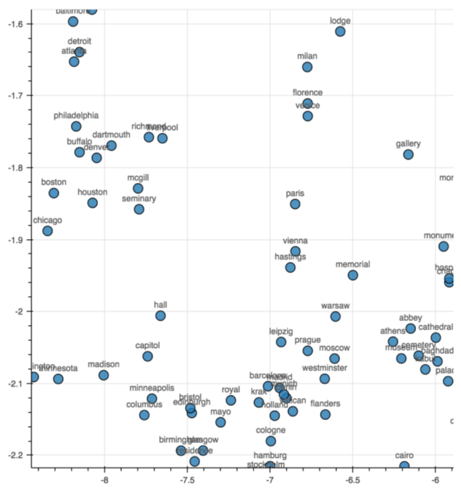
A simple example that demonstrates the way this visualisation approach works can be found by applying it to the MNIST dataset of handwritten digits [56]. We see in figure 5.1a that t-SNE 2-D embeddings are able to cluster the 10 digits from MNIST dataset into 10 distinct clusters [79]. This shows that graph embeddings can extract visual similarity from the raw data without supervision.



(a) MNIST with t-SNE



(b) CORA with t-SNE



(c) Word2Vec with t-SNE

Figure 5.1: Examples of using t-SNE visualisation on three commonly examined datasets.

A more practical example can be found when using t-SNE with word embeddings.

[1]Reproduced from [79]

Word embeddings project words into an embedding space in an analogous way to graph embeddings. A good word embedding would, therefore, place words with similar meanings close to each other in the embedding space. Using Word2Vec [70], a word embedding trained by trying to predict individual words given a context, the chart of cities in figure 5.1c was generated. In general, cities from the same countries are placed closer together, although there are some anomalies, see figure 5.1.

This approach also works well for graph-structured data. Once a graph embedding has been created for a graph it can also be visualised using t-SNE and the results will also tend to preserve a visual representation of the proximity measure that the embedding optimises. An example of this can be found by applying the GraphSAGE mode [41], a Deep Learning approach, to the CORA dataset of academic papers and citations as shown in figure 5.1b. This results in clusters on the class of the papers in the dataset, represented by the point colours, but with several exceptions that invite further investigation. In this case, we see that the information in the graph embeddings enables a clustering close to the underlying distribution of document classes, in this case representing the paper topic, without prior knowledge.

5.2 Visualising argumentation graphs

5.2.1 Introduction

Argumentation graphs are to some extent themselves a way of visualising arguments. However, for large argumentation structures with hundreds or thousands of arguments, such as are standard in ICCMA competitions, looking at the unadorned graph is of little utility. In the following sections, we will give two examples of how one can use graph embeddings to visualise information about the structure of arguments, one using node-level information to analyse a single argument and one to visualise properties of whole argumentation graphs. Both examples are based on the Abstract Argumentation formalism [29], but there is nothing inherent in this approach that limits it to one type of graph-based representation.

[2]Reproduced from https://nlpforhackers.io/word-embeddings/

[3]Reproduced from https://towardsdatascience.com/using-graphsage-to-learn-chapter-embeddings-in-cora-a94bb1e9dc9d

5.2.2 Node-level visualisation

To examine the possibilities of using Graph Embeddings and t-SNE to visualise argumentation graphs, we have started with a formal domain, Sembuster, that has the interesting property of having three types of arguments with distinct structural characteristics. The Sembuster domain, originally proposed by Caminada and Verheij [12], is composed of unique graphs generated for each cardinality, k, that can be partitioned into three different types: A, B, and C.

Arguments in the A partition only attack themselves. An argument B_i attacks all arguments A_j where $i \geq j$, arguments B_j where $i > j$ and the argument C_i. An argument C_i attacks the corresponding argument B_i.

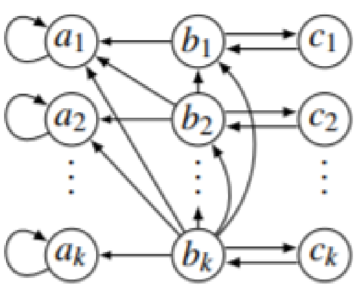
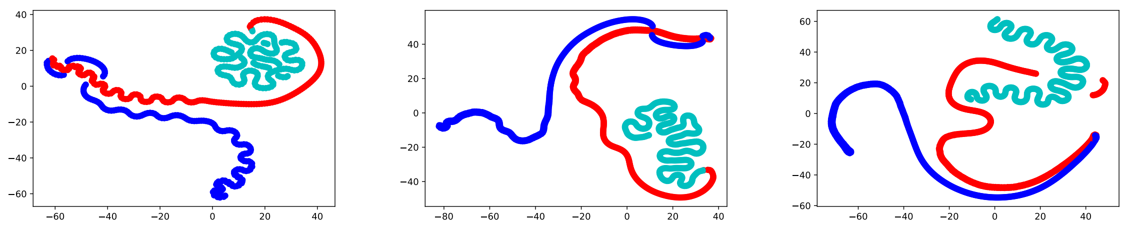


Figure 5.2: Visaulization of the Sembuster Scheme, reproduced from [12]

To visualise this configuration, we trained a graph embedding using the HOPE [76] method on a set of Sembuster graphs with a cardinality from 300 to 4500. HOPE is a directionality preserving embedding that works well on directed graphs such as argumentation graphs. HOPE can be used to generate node embeddings that preserve the structure of a graph while capturing its semantics. This makes HOPE a great choice for many applications, including link prediction, recommendation systems, and graph classification. Additionally, HOPE has been shown to outperform many other graph embedding techniques in terms of accuracy and running time.

The graphs were trained with a dimensionality of 128 features using a single K80 GPU. Training time was from 48 seconds to 5 minutes 12 seconds for a single embedding. These graphs were then visualised using 2-dimensional t-SNE using a colour coding corresponding to the argument partitions and subjected to visual inspection. If we examine the three examples, the three types of arguments stand out clearly. The self-referential A arguments are curled in on themselves in the visualisation and the related B arguments are closer to the A than the unrelated C arguments. The string-like construction of the Sembuster graphs seems at least superficially to also be represented in the string-like nature of the visualisation. While this is a highly formal example, it does indicate that the functional structure of arguments can in some cases be visualised using this type of tech-



(a) k=600 (b) k=1200 (c) k=1800

Figure 5.3: T-SNE Visualisation of Sembuster Graphs. Cyan represents partition A, Red partition B, and Blue partition C. Partitions are indicated in figure 5.1, k references the definition from figure 5.1

nique, although its application to arguments of a less formal nature would need further investigation. Applying similar techniques on other argumentation graphs that are under analysis would potentially be able to show functional clusterings of arguments based on the type of graph embedding that has been applied. It may even be possible to design specific embedding approaches that allow training based on the particular properties we are interested in analysing for.

5.2.3 Graph-level visualisation

We also trained a GCN model [51] to classify variants of graphs from different abstract argumentation graph domains. This allowed us to use a different approach to generate embeddings which is more suitable to the whole graph level, HOPE being more suited to local representations. It also gave us a method to apply to the AFGCN embeddings, which will be examined later.

We trained the embeddings on 10 separate domains using a custom built model. The domains are defined in Rodrigues et al. [86] and table 5.1 gives a description of their basis. The dataset was based on a subset of the tasks from the 2nd International Competition on Computational Models of Argumentation using an even sample drawn from these 10 domains, which all form part of the competition corpus, see http://argumentationcompetition.org/.

The GCN model used 4 convolutional layers followed by 2 fully connected layers to try to predict the class, in this case what benchmark the graph belongs to, of a given input graph. The implication would be that the visualisation should show the graphs in a given domain following in a recognisable pattern (e.g. clustering or equal spacing). This could for instance be useful in situations where you have an argumentation corpus that is not clustered or classified and you want to group them by similarity and be able to show that similarity in a visual mode. After the model had been trained for 4 hours, the training was stopped and the output of the last convolutional layer was extracted as a vector and used for visualisation.

Table 5.1: Argumentation problem domains for graph-level visualisation experiment

Identifier	**Domain**	**Description**
afinput	ABA2AF	Assumption-Based Argumentation translated to abstract argumentation frameworks
admbuster	AdmBuster	AdmBuster graphs, based on Caminada and Podlaszewski [13]
BA	Barabasi-Albert	Barabasi-Albert graphs, randomly generated
ER	Erdös-Rényi	Erdös-Rényi graphs, randomly generated
grd	Grounded	Randomly generated argumentation frameworks containing only a grounded extension
.cnf	Planning2AF	Planning problems transformed to abstract argumentation problems
sembuster	SemBuster	SemBuster graphs, see section 4.2
scc	SccGenerator	Randomly generated argumentation frameworks containing multiple strongly connected components
.gml.	Traffic	Traffic networks converted to abstract argumentation frameworks
WS	Watts-Strogatz	Watts-Strogatz graphs, randomly generated
WS	Logic Based Argumentation	Logic based argumentation problems

This had more ambiguous results. For most domains it was possible to identify a clear separation between classes either by clustering or by equal spacing. Examples of these are shown in Figure 5.4. First, for the *ER*, *.cnf*, and *grd* domains we can see a clear separation between the three classes through three spacing patterns that are distinct by class. For *sembuster* and *scc* this pattern is even more distinct. However, for the afinput, admbuster, and BA domains it is less clear with less separation between the classes. For .gml and

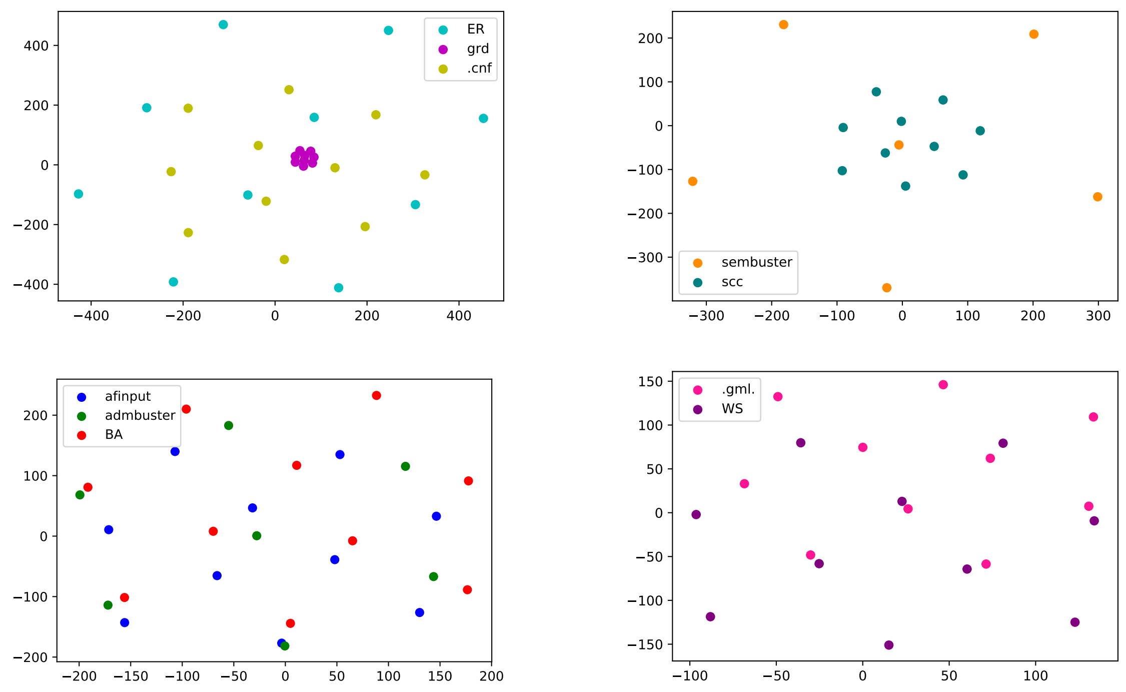


Figure 5.4: Graph-level Visualisation using t-SNE across benchmarks in table 5.1, grouped by similarity of embedding

WS there is some overlap showing that there is a separation between the classes, but that some graphs are difficult to distinguish.

While this approach does demonstrate the feasibility of training and visualising graph-level embeddings of argumentation graphs, it is some way from being practically useful in its present form. What it does demonstrate is that a properly trained unsupervised embedding may be able to accurately separate argumentation graphs based on a training task that will then be available for visualisation. That means potentially being able to cluster corpora of argumentation graphs based on a variety of training tasks in order to discover new ways of classifying and categorising the arguments. In this case a simple supervised training task is used to generate a separation into already known classes, but both other well-known graph embeddings or a custom designed embedding for the task at hand might give improved results.

5.3 Application to embeddings from AFGCN

While we acknowledged above that the current approach to visualising Graph embeddings for the purposes of abstract argumentation is quite experimental, we have still found it worthwhile to attempt to see what if any insights could be gleaned from applying it to the models used by the AFGCN solver described in chapter 3. We have done this in a way identical to the node level classification task described above, cutting off the final classification layers of the model.

The analysis we have focused on is based on a breakdown by benchmark, evaluated under the DC-PR task. There are two reasons for this. First, we wish to find out whether there are any additional explanatory power in the visualisations for explaining the predictability of various benchmarks. Second, from a visual perspective it is relatively intuitive to engage with.

In general, what the visualisations seem to produce are patterns where high performing benchmarks cluster graphs in a tight formation around the centre of the figure and lower performing graphs have a much higher degree of distribution and randomness to where their graphs are placed on the visualisation.

The best performing benchmark under this evaluation setting was Logic Based Argumentation, with a perfect score. This benchmark does in fact show the pattern described above with all of its graph instances clustered tightly around the centre as shown in figure 5.5. That indicates that clustering with t-SNE captures something about the quality of the classifier and vice versa.

The same is broadly speaking true of other benchmarks where the classifier has high performance such as ABA2AF. this is shown in figure 5.6 below. In this case the cluster is slightly less tight, perhaps reflecting the slightly worse performance of the classifier, again adding some substance to our hypothesis.

At the other end of the scale, the AFGEN benchmark, in general one of the worst performing benchmarks for the classifier, shows a pattern that although not entirely random is spread out and not particularly systematic. This is shown in figure 5.7. This reflects the inability of the GCN to accurately separate the AFGEN instances in the high-dimensional hyperspace used by the GCN and therefore of t-SNE to cluster them.

The generality of this pattern can be seen by juxtaposing an image of the best performing patterns set off against the worst performing and vice versa as shown below in figure 5.8. While things do blur a little bit in this visualisation, there is no doubt that the same general pattern is visible.

So, what can we make of this in terms of how our classifier operates? One thing that seems evident is that the structure of the graph model used in a benchmark is pivotal both

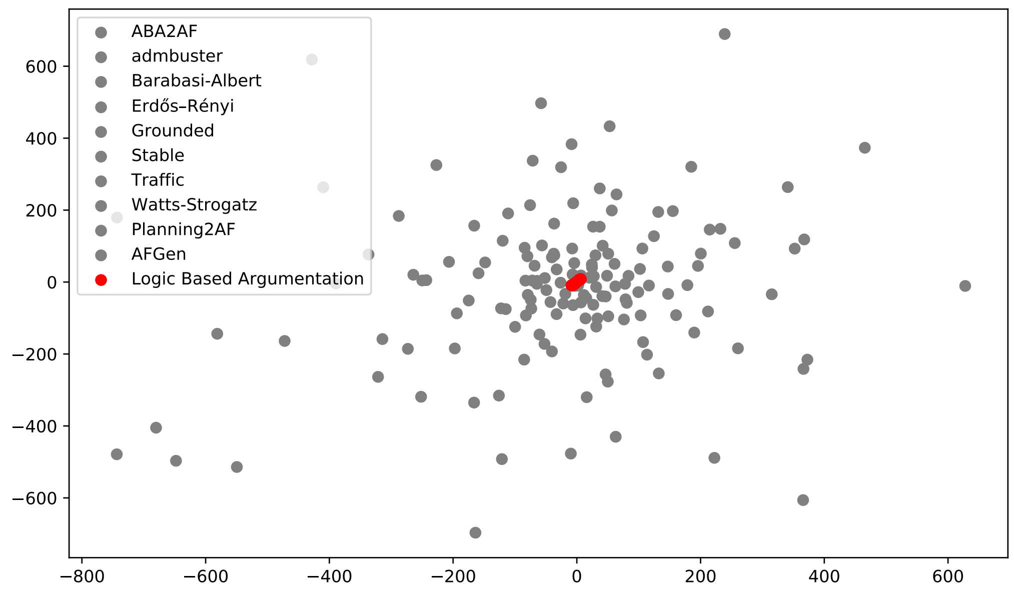


Figure 5.5: Overview of visualizations of AFGCN embeddings. Logic Based Argumentation highlighted in red

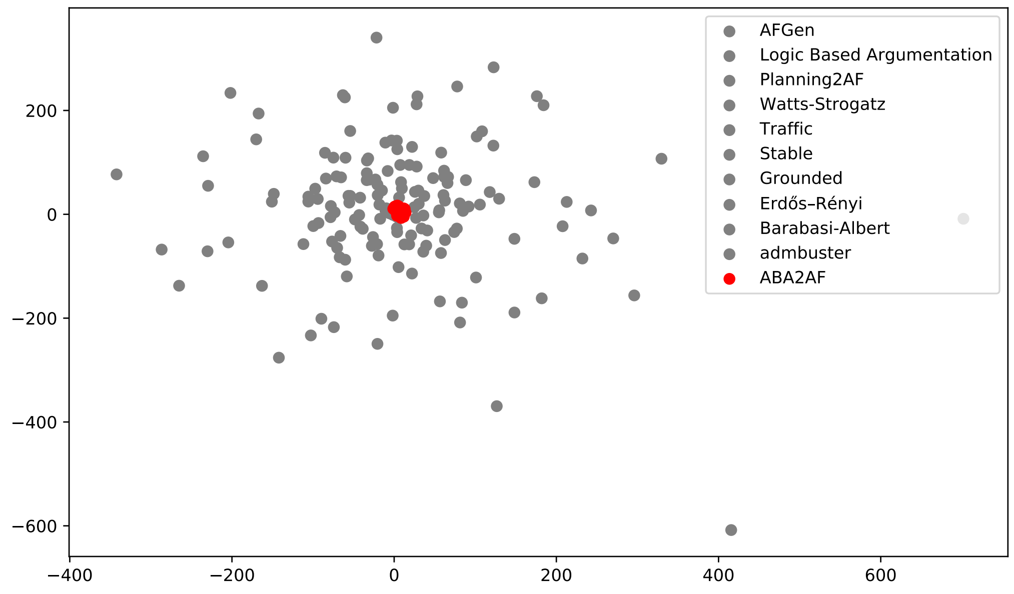


Figure 5.6: Overview of visualizations of AFGCN embeddings. ABA2AF highlighted in red

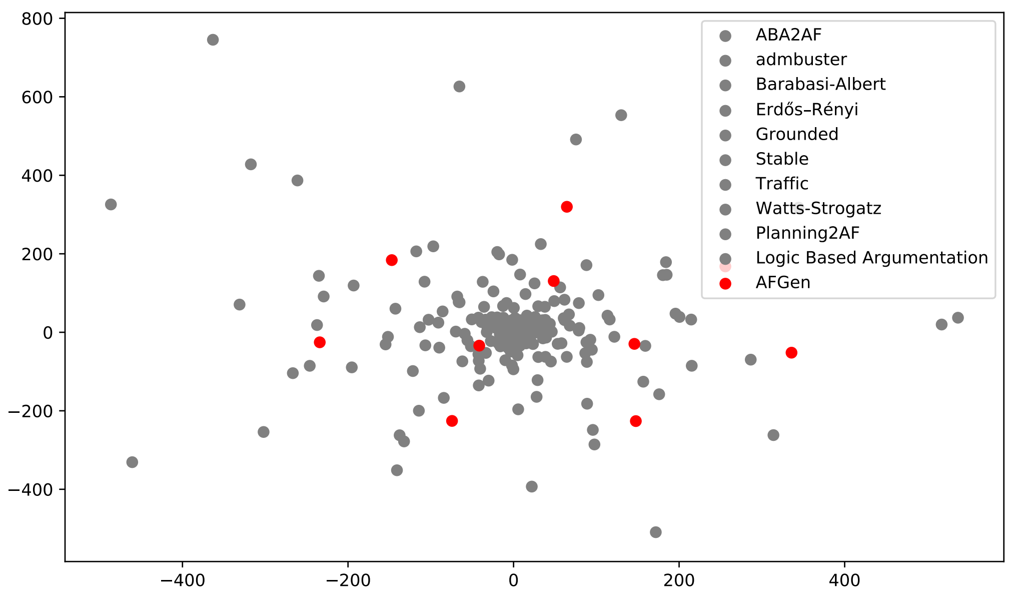


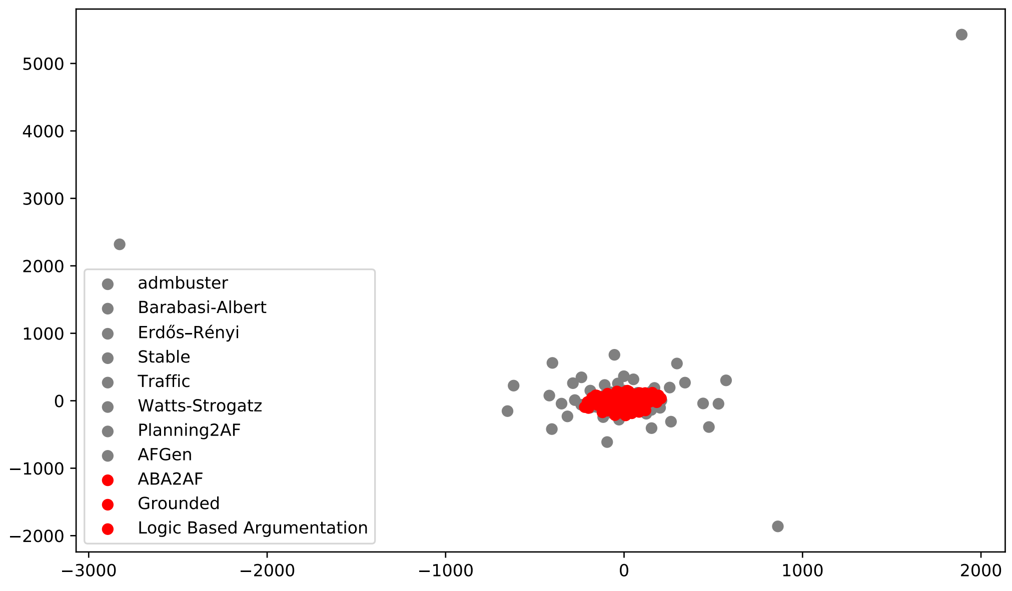
Figure 5.7: Overview of visualizations of AFGCN embeddings. AFGEN highlighted in red

in terms of classifier performance and in terms of the ability of the visualisation to form meaningful clusters. The best performing benchmarks all have a clear logical structure, where is the worst performing benchmarks are effectively based on random graph models. This leads one to speculate that the machine learning model is in fact engaging with the logical reasoning tasks of the abstract argumentation domain and that that is why we see the overlap between good performing classifiers in the areas where systematic consistency is most pronounced. However, it is not possible to declare this with any certainty on the basis of this experiment.

5.4 Summary

In this chapter, we have shown that it is possible to use graph embeddings combined with t-SNE to visualise properties of argumentation graphs at both the node and the graph level. The node-level experiments were more successful, showing three distinct clusters according to the function of the arguments in the graph. However, the graph level example did show that visual clustering of argument graphs by using a graph embedding is a least possible, although it needs some refinement to be applicable in practice. Further development of both the training task and the network architecture should yield improved results.

Application to the embeddings from the AFGCN solver described in chapter 3 showed



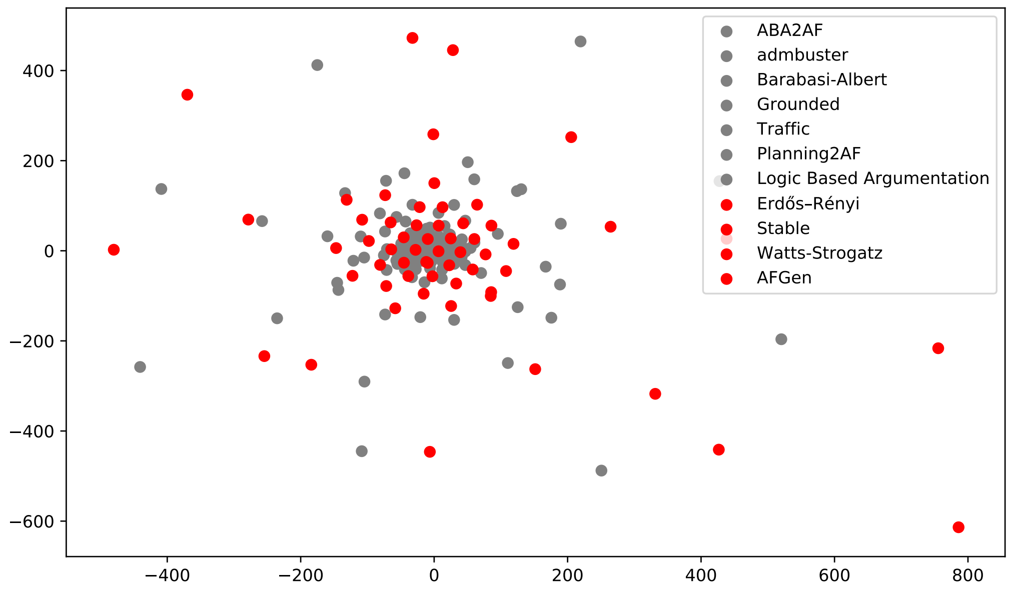


Figure 5.8: Overview of visualizations of AFGCN embeddings. High performing benchmarks on top vs low performing benchmarks on bottom. Relevant group highlighted in red

that some patterns were discernable visually that was able to illustrate points made in the analysis conducted during that discussion. However, the insights were limited in scope and we were only able to suggest relationships between visualisations and the way the machine learning model works in practice rather than get at any underlying explanation for that practice, which would be a stronger result.

Wider exploration of this space would require the analysis of argumentation graphs of diverging provenance across many different domains. This would also involve testing a wider range of graph embeddings and eventually exploring the creation of embeddings from scratch with the kind of properties that would make them particularly useful for visualising argumentation graphs. The aim would be to enable the clustering of argumentation graphs in an unsupervised manner. This would enable visual analysis of similarity at the graph level and commensurately give us the ability to see the visual clustering of individual arguments in graphs. Also we might be able to visually group arguments within a graph by some measure of similarity generated by the graph embedding.

Considering our guiding research question for the chapter, we can say that there is some indication that visualising might yield additional insights, but that they are still of a limited and tentative nature. More research into different ways of representing and embedding the neural network features might yield stronger results. We will now move on to our final application of the model developed in Chapter 3 as we consider applying the GCN architecture to the problem of misinformation detection.

Chapter 6

Improving Misinformation Detection in Tweets with Abstract Argumentation

The problem of detecting misinformation in social network data has received increasing attention recently, not least due to the COVID-19 public health emergency [9]. It is fair to say that deep learning models based on a variety of architectures have been increasingly successful in learning to detect various types of misinformation such as rumours, fake news, and the intentional spreading of false information [45].

For Twitter data two of the most successful recent approaches have been on the one hand detection based on large-scale language models [53], using the characteristics of the language used to determine veracity, and on the other methods that use the features of the propagation graph by which a source tweet is retweeted by other actors on the social network [60], using the structure of the graph as an indicator of veracity.

Our approach seeks to combine and enrich these two approaches by combining graph structure, in the form of an abstract argumentation framework derived from stance information with linguistic node level features from a language model and argumentative features based on the acceptability status of arguments under different semantics.

This chapter presents work towards this goal, as a concrete example of the kinds of applications mentioned in research question 7, “Can we extend the basic architecture of the approximate solver to other related problems with similar structure?”. This chapter builds on Chapter 3, where we presented a way to approximate acceptability in abstract argumentation frameworks using graph neural networks. While the approximation task was different in that example (e.g. determining acceptability), the same GCN architecture can apply in this case.

We define misinformation in the way typical of the literature, which is to say that a tweet is considered misinformation if it is marked so using manual human annotation. Stance is defined similarly, indicating the attitude of a tweet towards a source tweet obtained by manual human annotation.

As part of this research, we make the following contributions:

- Show that adding the structure of an argumentation graph can improve detection of misinformation relative to baseline language models.
- Describe a way to construct abstract argumentation graphs from stance information included in the dataset.
- Determine the possible architectures for combining linguistic and argumentative features for input into Graph Convolutional Networks (GCN).
- Apply the method to a commonly used dataset within the field, obtaining promising results.

We start reviewing previous work on misinformation detection that is relevant to the experiments in this chapter. Then we elaborate on the methodology, we employ to adapt the GCN architecture to the problem of misinformation. Finally, we present the results of our experiments and discuss their implications.

6.1 Misinformation detection on social networks

There is a large extant literature on misinformation detection. This includes both intentionally misleading statements and more subtle attempts to manipulate via information dissemination. The two approaches most directly relevant to our research are those that rely on large-scale language models such as GPT-3 [35] and BERT [27] to analyze tweet content [53] and those that use the patterns of tweet propagation to detect misinformation [60].

However, the most directly parallel work in the literature is found in Orascu and Toni's [23] use of argumentative features to improve a Bi-LSTM model for detection of fake reviews and to determine whether news headlines support tweets. This research showed the power of argumentative features for this problem. However, compared to our research, it relies on a more complex formalism, bipolar argumentation frameworks, and uses the argumentative features as an adjunct to a principally NLP-based model instead of having a primary focus on graph structures.

6.2 Method

Our approach to the problem follows a four step process shown in figure 6.1:

1. Construct an abstract argumentation graph based on stance information.
2. Generate linguistic features by creating embeddings for the input tweets, using a language model.
3. Generate argumentative features by resolving the acceptability of the arguments in the argumentation framework.
4. Train a Graph Convolutional Network using the argumentation graph and the generated features inserted at the node level.

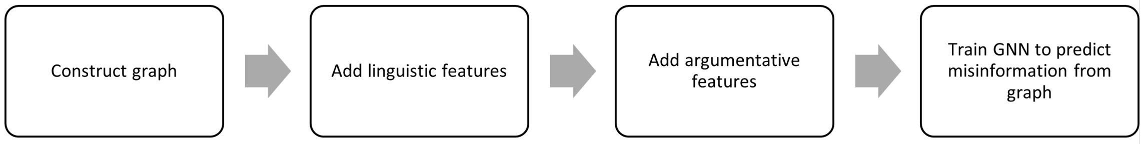

Construct graph
Add linguistic features
Add argumentative features
Train GNN to predict misinformation from graph

Figure 6.1: Methodology for the misinformation detection task

In the following section, we will explore each of these steps in more detail.

6.2.1 Constructing the argumentation framework

Typically, the structure used to represent Twitter data is the propagation graph. A propagation graph is a type of network graph that represents the spread of tweets. It consists of nodes, which represent individual tweets, and edges, which represent connections between tweets. These connections may represent follows, mentions, replies, or retweets.

The characteristics of a Twitter propagation graph can vary depending on the specific data and algorithms used to construct it. However, some common characteristics of these graphs may include a high level of connectivity and clustering, with densely connected groups of users forming around certain topics or ideas. The graph may also exhibit preferential attachment, where popular users or ideas tend to attract more connections over time.

In addition to these structural properties, a Twitter propagation graph may also contain information about the content of the tweets being shared, such as hashtags, keywords, or sentiments. However, it does not take into account the semantic relationship between the tweets, which can be partially achieved by looking at relative stance.

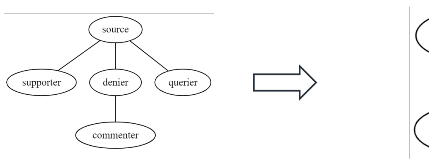
We will, therefore, construct the argumentation framework based on existing stance information present in our source datasets taking into account these characteristics. The problem of stance detection has a substantial literature of its own [54] and we do not seek to add to it with this research.

While stance categorization can vary between datasets it is generally possible to classify the relationship between a source tweet, that is the target for classification, and additional tweets in its propagation graph into a polarity of positive, negative, or neutral. We use this polarity to construct abstract argumentation frameworks, in the broad sense of argumentation framework as a method for representing situations of conflict, containing arguments representing each tweet and attacks based on the following schemes for mapping stance into attack relationships:

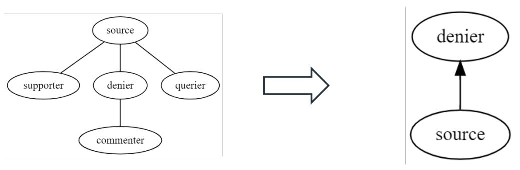
- Adding attack relationships from any node that has a negative stance towards the source node to the source node itself (Scheme 1).
- Adding attack relationships from the source node to any node that has negative stance towards it (Scheme 2).
- Adding attack relationships between nodes that have differing stance to the source node. So if tweet A is positive toward the source node and tweet B is negative, we would add either unilateral or bilateral attack relationships, depending on our chosen scheme (Scheme 3).
- Limiting these attack relationships to only those in a tweet's sub-propagation graph, that is to say only for nodes in the same sub-graph measured from the source node (Scheme 4).
- Adding neutral nodes to the graph both as isolated components (Scheme 5), only linked to themselves and with attack relationships towards the source node (Scheme

6) or to nodes with a negative stance towards the source node (Scheme 7), thereby reclassifying neutral nodes as positive or negative for the purposes of the argumentation graph.

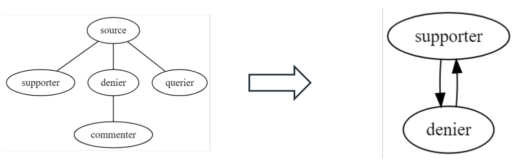
These schemes are illustrated in figure 6.2.



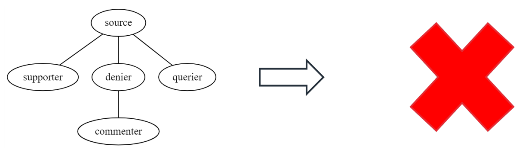
(a) Scheme 1



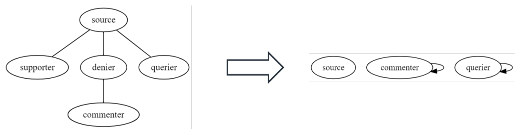
(b) Scheme 2



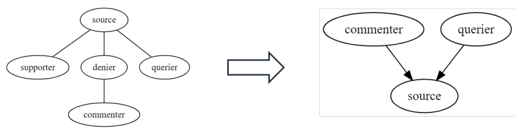
(c) Scheme 3



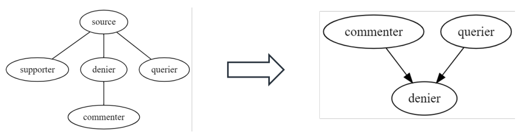
(d) Scheme 4



(e) Scheme 5



(f) Scheme 6



(g) Scheme 7

Figure 6.2: Scheme diagrams, supporter = positive stance, commenter/querier = neutral stance, denier = negative stance

While some of these schemes may seem prima facie strange from a common sense point-of-view, they are designed to allow different ways for neighbourhood information to aggregate in the graph convolutional network, which may help in the classification task. That is to say that we in some cases add more relationships in the argumentation graph than is indicated by the propagation graph in order to aggregate information in larger neighbourhoods and thereby more faithfully represent that actual underlying argument taking place across the tweets.

6.2.2 Linguistic features

We generate linguistic features by creating embeddings for all tweets in our dataset using a variety of language models. First and foremost, we use the large-scale transformer based language models such as BERT. But we will also include simpler models such as Word2Vec for comparison. We generate these using different embedding sizes for comparison. These embeddings are added as node features in the GCN model along with the argumentative features. We will, however, also train a baseline classifier using only linguistic information for the sake of comparison.

6.2.3 Argumentative features

Argumentation specific features are generated by incorporating the acceptability status both for sceptical and credulous acceptance under the Complete, Preferred, Stable, Semi-Stable, and Stage semantics. The features will be pre-calculated using an exact abstract argumentation solver and added as node features by encoding acceptable as 1 and unacceptable as 0.

6.2.4 GCN architecture

We follow the GCN Architecture described in chapter 3, using the GCN-NO-GR model, and adapt it to incorporate the additional feature information from the language model and the argumentation solver. We chose this base model as there is no reason to expect any benefit from grounded reasoning in the schemes that we are constructing. Given the small size of the graphs, we can easily calculate acceptability in an exact manner and instead use the GCN model for its general approximation properties, in this case to approximate misinformation.

The architecture includes the following elements:

1. Pre-computed linguistic and argumentative features along with the normalised adjacency matrix for the argumentation framework.
2. An input layer receiving these inputs.
3. 4 repeating blocks of a GCN layer [51] and a Dropout layer [89].
4. Residual connections feeding the original features and the normalised adjacency matrix as additional input at each block.

5. A layer that aggregates the embeddings generated by the GCN layers for graph level classification. We will experiment with different aggregation functions during the final phase of the research.

6. A sigmoid layer with a single neuron that represents an estimate that the source tweet is true.

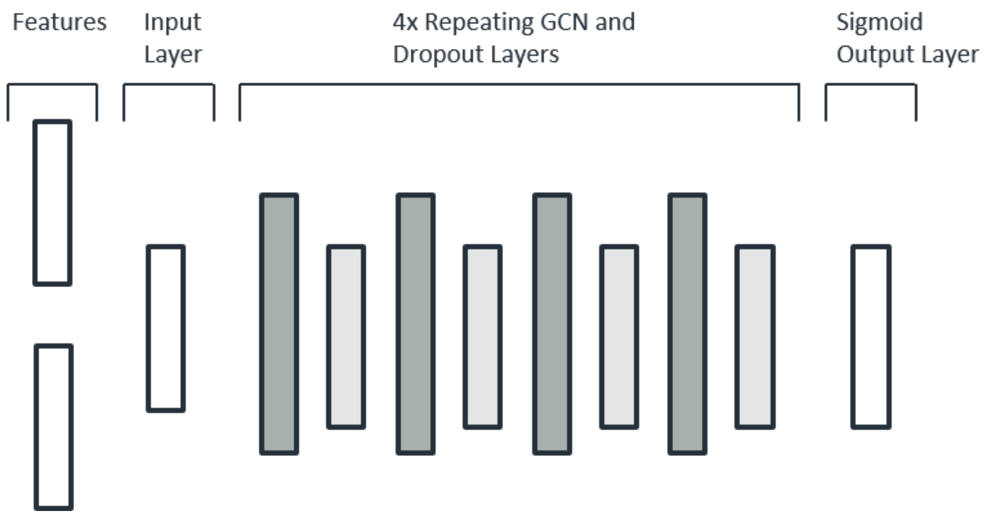


Figure 6.3: GCN architecture used in misinformation detection experiments.

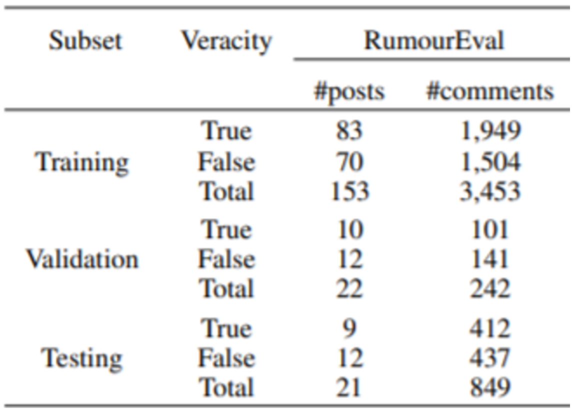
We treat the problem of veracity as a binary classification problem at the level of the graph. That means we aggregate information from all nodes and the embeddings generated by the GCN in order to judge whether the source tweet is true or not.

6.3 Experimental results

We test the model on a dataset that contains both veracity and stance information and is commonly used in the research literature: RumourEval [39]. The dataset contain tweets relating to controversial current events.

We ran our experiments with the following method:

- Generated argumentation graphs using combination of schemes 1-3. We read in the stance information and propagation graphs from JSON files included with the RumourEval dataset and used Python code to construct the graphs with the NetworkX library.

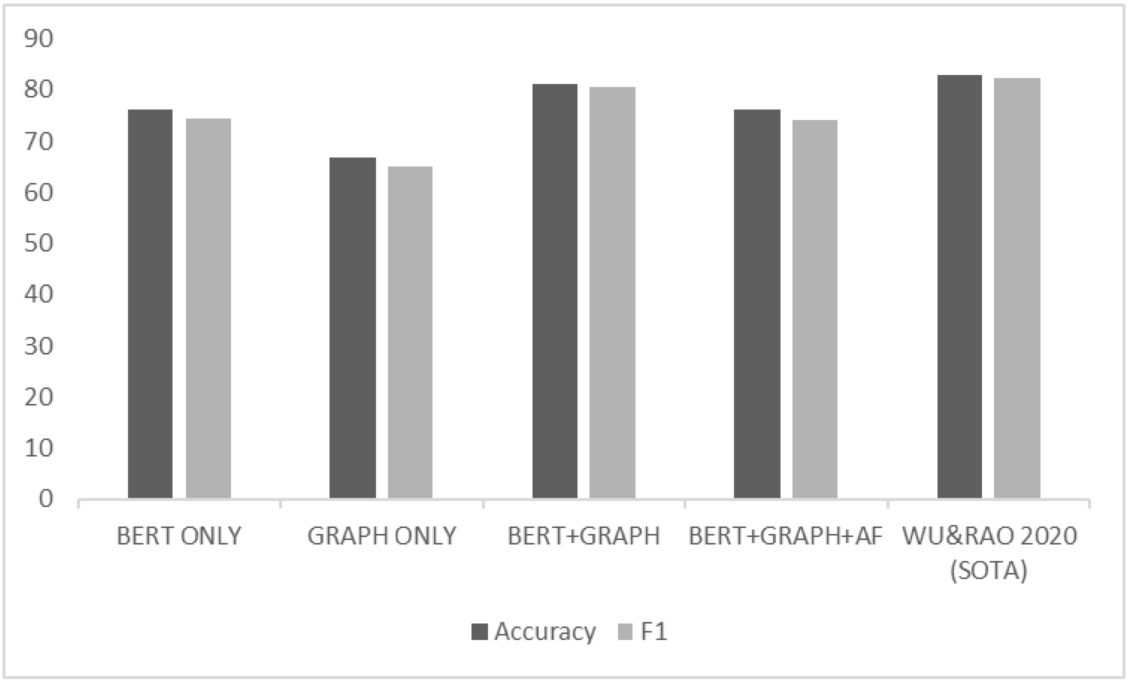


Subset	Veracity	RumourEval	
		#posts	#comments
Training	True	83	1,949
	False	70	1,504
	Total	153	3,453
Validation	True	10	101
	False	12	141
	Total	22	242
Testing	True	9	412
	False	12	437
	Total	21	849

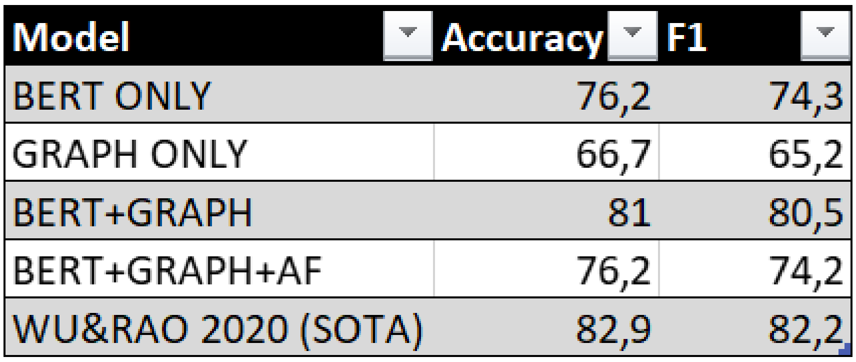
Table 6.1: Key statistics of the RumourEval dataset used for the experiments in Chapter 6, reproduced from [39]

- Fine-tuned BERT model on RumourEval dataset. This was done using the Hugging Face API.
- Generated linguistic features for all tweets in dataset. This was done with the Hugging Face API, using the fine-tuned model from the previous step. Features were added directly to a PyTorch tensor for input to the GCN.
- Added argumentative features from pyglaf. This was done using a batch script that worked on plain text versions (in tgf) of the generated argumentation graphs. The results were added as input features directly to a PyTorch tensor.
- Adapted GCN model to problem of veracity detection. Changed the model described in Chapter 3 by changing the input layer to accomodate the new features and the training to use ground truth read from JSON files provided by the RumourEval dataset, otherwise the model was identical.
- Trained and tested on RumourEval dataset. This was done in an identical manner as described in Chapter 3.
- Compared to two baseline models using BERT only or argumentation graphs only. These models removed the relevant parts of the model for comparison.

The results show that BERT on its own performs relatively well and somewhat better than using the raw argumentation graph, which does, however also have some predictive power. Combining the two approaches gives an uplift in overall performance thereby giving some evidential support for the hypothesis that we wanted to investigate. Adding



(a) Results graph



Model	Accuracy	F1
BERT ONLY	76,2	74,3
GRAPH ONLY	66,7	65,2
BERT+GRAPH	81	80,5
BERT+GRAPH+AF	76,2	74,2
WU&RAO 2020 (SOTA)	82,9	82,2

(b) Results table

Figure 6.4: Evaluation of experiments on RumourEval dataset using GCN based misinformation detection

the argumentative features extracted via the pyglaf argumentation solver actually registers a slight drop in both accuracy and F1. However, this drop is so small that it may not be significant and there is no theoretical reason to believe there should be such a drop. None of these approaches match the performance of the current state-of-the-art for the dataset [53], a model based on adaptive interaction fusion networks. Further, one can note that F1 and Accuracy scores correlate well across all results in these experiments.

6.4 Summary

This chapter has presented the work for our experiments in applying abstract argumentation in conjunction with linguistic features for detecting misinformation. We hope that this will extend the applicability of argumentation based methods for misinformation detection as well as give greater understanding of how argumentation can be used to enrich and improve deep learning architectures.

This can also help decrease the gap between the formal world of abstract argumentation and natural language expressions of argument by incorporating both types of information in the same deep learning model. So far early work has demonstrated the feasibility of this approach and further work might be able to get the approach closer to the state of the art.

In terms of our guiding research question for the chapter, we have shown an application of the approach. This has not produced a state-of-the-art results, but does demonstrate feasibility. Further research could extend both this application and develop others along the same lines.

There is, however, another interesting angle to this work. Because of the way the schemes are constructed, the input graph has an argumentative structure that lends itself to various explainability approaches [106] because the relationships indicate what parts of the overall propagation graph are consistent or in conflict. That is to say the method of recreating the propagation graph as an argumentation graph introduces explainable input features that combined with the output of the GCN can make the classification of an individual tweet potentially explainable despite the fact that GCNs are not typically among the easiest network architectures to explain. There is, therefore, a potentially fruitful avenue of research to be explored in transforming graph structures to argumentative structures prior to being input to a GCN based classifier.

Chapter 7

Conclusion

7.1 Conclusion

In the introduction, we defined an overarching research questions and seven sub questions that we used to explore our initial hypotheses:

RQ: Can you improve on current solution methods in abstract argumentation using Deep Neural Networks (DNNs) in particular GNNs.

SQ 1 Is it possible to develop a high-performing approximate solver for abstract argumentation using GNN methods?

SQ 2 What neural network architecture and training methods work most effectively for the purposes of approximating abstract argumentation problems?

SQ 3 Given the polynomial solvability of the grounded extension, can we use that as a starting point to improve approximation?

SQ 4 Are there significant differences in the approximability of argumentation frameworks based on semantics, benchmark type or size?

SQ 5 Can the output of an approximate solver be used as a heuristic to improve the performance of exact solution methods?

SQ 6 Does visualising the structure of the GCN model used for approximation give us additional insight into the way it functions?

SQ 7 Can we extend the basic architecture of the approximate solver to other related problems with similar structure?

Returning to the progression of our research, we have demonstrated that it is possible to use GNNs to approximate problems in abstract argumentation to a good degree of accuracy. We developed and explained the method and architecture used to fulfil this goal, refining it with a number of ablation studies, and then applied it systematically to a large argumentation data set. That work showed significant variations between semantics, size bands, and in particular benchmarks representing different types of graph models, some of which were readily approximable, others that were less so. This covered our response to the first four research questions.

We moved on, following research question 5, to define a notion of co-admissibility that allowed us to use our GCN architecture to predict sets of arguments compatible with the input argument, which were then used as an input into two different exact methods. First, we designed an experimental search algorithm that leverages the co-admissability predictions as part of the search. Second, we used the co-admissibility predictions in a SAT heuristic and were able to show some improvement on difficult instances by using the heuristic.

Then we defined a new visualisation approach based applying t-SNE to the weights of our machine learning models in order to address research question 6. This allowed us to get slightly more insight into some of the benchmark related variation noted above.

Finally, we adapted the AFGCN solver that we had used for approximating acceptability to a different problem, detecting misinformation in tweets in order to explore research question 7. We did this by constructing an argumentation graph based on stance information and adding linguistic features from a large-scale language model. While this did not obtain state of the art results, it did show the viability of adapting our approach to different problems with good performance.

We will now proceed to consider our key contributions.

7.2 Key contributions

Throughout this thesis, we have highlighted specific contributions that we have been able to make as part of the research presented. However, at a bird's eye view, the key contributions of this thesis are as follows:

- We developed a new approach to approximating solutions to problems in abstract argumentation, using GNNs. This approach set a new state-of-the-art and won 4 out of 6 categories in the ICCMA 21 competition for approximate argumentation solvers

- We systematically investigated the properties of this approach across argumentation semantics and benchmarks, and also conducted ablation studies to determine the key parameters for the neural network architecture
- We linked the approximate approach described with exact approaches by using the approximate solutions as an input to a SAT based solution approach and an experimental, but exact, algorithm
- We presented a new visualisation approach for argumentation frameworks, leveraging the GNN architecture that we developed for approximation, and used it to derive additional insights about our neural network model
- We presented an application of the GNN model that we developed for approximation to the classification of misinformation in Twitter data. Although we didn't set a new state-of-the-art on the key datasets, we demonstrated that the approach we use for approximating acceptability in argumentation frameworks has wider applicability

7.3 Future work

We will end this thesis by pointing out some areas that are ripe for further exploration. We have, of course, not been able to cover all relevant avenues of research here and the sections below, therefore, contain ideas for how it may be extending in different directions.

7.3.1 Further improvements to accuracy

While the accuracy obtained is good in aggregate, there is still much room for improvement in particular when it comes to certain poorly performing benchmark types. We believe there are a number of routes to obtain better performance:

- Developing argumentation specific graph embeddings
- Using more data, targeted data, and data augmentation techniques
- Using additional architecture elements such as Deep Reinforcement learning. We performed a number of experiments using this approach and although we were not

able to make it generalise well, we believe it could be made to work with further research [62]

7.3.2 Extension to other argumentation formalisms

In this thesis, we have only considered the basic formalism of abstract argumentation. There is, however, nothing inherent in our approach that would prevent it from being adapted to other similar formalism such as bipolar argumentation [15], assumption based argumentation [30], or probabilistic argumentation [93]. Extending AFGCN in this direction, would make the solver more versatile and useful for future research.

7.3.3 Adaptation to additional problems

Our exploration of the misinformation problem in chapter 6 opens up another way to extend the work presented in this thesis. Specifically, if we can model the input to a prediction task partly or fully as an abstract argumentation graph, the approach given in this thesis presents a ready-made architecture that can be adapted to the task.

Social media analysis in various forms presents a particularly interesting avenue to explore, but the approach could be extended to any situation of conflict, where an argumentation framework can be inferred based on real-world information.

7.3.4 Deepen neuro-symbolic linkages

The approach presented in chapter 4 can be thought of as following one of five categories of neuro-symbolic linkage as defined by Henry Kautz, in his AAAI 2020 Robert S. Engelmore Memorial Award Lecture. This model is called Neuro $\rightarrow$ symbolic and involves using the outputs of a neural network in a symbolic approach. There are, however, four different additional ways to link neural and symbolic methods in this typology that are at least worth exploring for viability.

In general, it would be a great benefit to be able to retain the provability, explainability, and simplicity of abstract argumentation, while speeding up computational runtime and degree of parallelism by using a neural network.

7.4 Broader knowledge contributions

Below we will cover some additional trends in the larger research environment that while they are not directly the focus of this thesis, provide interesting perspectives on the relevance of our work.

7.4.1 Neuro-symbolic AI

There is an increasing research focus in many quarters on approaches that link symbolic and neural AI. A recent survey [87] showed that there had been a marked increase in the number of papers accepted at top conferences on this topic (from 2 to 15), although the number is still low relative to the total number of papers accepted. This thesis feeds into this trend by using data from a neural network in exact solution approaches.

7.4.2 Argumentation and machine learning

There has also been growth of research into different areas linking argumentation and machine learning, the main thrust of this thesis' approach. The first workshop on Argumentation and ML is due to take place September, 2022, see https://argml2022.csc.liv.ac.uk/.

Some areas such as argument mining has long used machine learning techniques [55]. However, there is now substantial research both into how to engage with problems in argumentation with machine learning techniques, the general area of this thesis, and how to improve machine learning by using argumentation techniques [22]. It's also worth highlighting the research linking argumentation and explainable AI in this discussion [106].

Appendix A

Additional Results Tables

Table A.1: Results DC-PR - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	83.10%	85.36%	77.89%	86.40%	63.06%	0.64	0.54
GCN-WITH-GR	83.96%	86.12%	76.87%	87.52%	69.19%	0.7	0.58
GR-ONLY	63.98%	100.00%	59.69%	100.00%	37.93%	0.43	0.34
HYBRID-GCN-GR	84.69%	88.27%	76.61%	89.93%	68.89%	0.69	0.58

Table A.2: Results DC-PR - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	96.45%	93.08%	96.64%	93.15%	76.11%	0.81	0.8
GCN-WITH-GR	96.82%	95.22%	96.97%	95.27%	80.70%	0.85	0.84
GR-ONLY	95.84%	100.00%	95.47%	100.00%	89.70%	0.91	0.9
HYBRID-GCN-GR	97.39%	96.59%	97.38%	96.65%	93.48%	0.93	0.92

Table A.3: Results DC-PR - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	80.90%	81.63%	80.05%	82.89%	48.52%	0.51	0.44
GCN-WITH-GR	82.90%	81.88%	82.22%	83.21%	61.18%	0.64	0.55
GR-ONLY	70.89%	100.00%	68.31%	100.00%	27.86%	0.36	0.33
HYBRID-GCN-GR	82.52%	82.39%	80.58%	84.81%	54.37%	0.57	0.5

Table A.4: Results DS-PR - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	86.24%	84.53%	86.99%	87.81%	51.82%	0.53	0.5
GCN-WITH-GR	87.66%	84.78%	87.89%	87.55%	57.91%	0.61	0.56
GR-ONLY	84.99%	100.00%	83.31%	100.00%	46.79%	0.51	0.52
HYBRID-GCN-GR	85.14%	76.19%	86.62%	81.00%	57.82%	0.55	0.5

Table A.5: Results DS-PR - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	97.21%	91.44%	97.61%	91.66%	77.09%	0.81	0.81
GCN-WITH-GR	97.88%	94.48%	98.03%	94.59%	80.41%	0.85	0.85
GR-ONLY	98.27%	100.00%	98.13%	100.00%	91.20%	0.92	0.93
HYBRID-GCN-GR	95.58%	77.77%	98.35%	78.33%	92.31%	0.79	0.8

Table A.6: Results DS-PR - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	88.57%	82.43%	90.39%	85.06%	48.17%	0.5	0.5
GCN-WITH-GR	90.97%	84.49%	91.48%	86.22%	57.06%	0.62	0.6
GR-ONLY	90.16%	100.00%	89.33%	100.00%	49.88%	0.55	0.58
HYBRID-GCN-GR	88.54%	78.85%	90.59%	81.87%	56.22%	0.57	0.54

Table A.7: Results DC-CO - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	81.31%	88.16%	69.93%	89.71%	63.11%	0.64	0.46
GCN-WITH-GR	88.02%	84.34%	78.52%	85.13%	75.34%	0.75	0.58
GR-ONLY	63.86%	100.00%	59.65%	100.00%	38.99%	0.43	0.28
HYBRID-GCN-GR	86.58%	86.46%	76.61%	87.42%	77.64%	0.75	0.58

Table A.8: Results DC-CO - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	96.31%	93.21%	96.50%	93.33%	77.62%	0.82	0.81
GCN-WITH-GR	97.15%	92.22%	97.25%	92.28%	85.44%	0.88	0.86
GR-ONLY	96.37%	100.00%	96.01%	100.00%	90.75%	0.92	0.92
HYBRID-GCN-GR	97.47%	97.53%	97.34%	97.56%	94.79%	0.95	0.93

Table A.9: Results DC-CO - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	81.19%	83.34%	80.86%	85.35%	58.25%	0.58	0.47
GCN-WITH-GR	84.75%	82.57%	84.15%	83.60%	65.61%	0.68	0.57
GR-ONLY	74.75%	100.00%	72.21%	100.00%	35.63%	0.42	0.41
HYBRID-GCN-GR	82.56%	84.78%	81.44%	85.98%	63.73%	0.65	0.55

Table A.10: Results DS-CO - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	97.00%	83.37%	98.52%	88.91%	86.84%	0.78	0.83
GCN-WITH-GR	97.54%	86.74%	99.10%	90.76%	91.58%	0.84	0.88
GR-ONLY	100.00%	100.00%	100.00%	100.00%	100.00%	1.0	1.0
HYBRID-GCN-GR	99.64%	98.59%	100.00%	99.05%	100.00%	0.99	0.99

Table A.11: Results DS-CO - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	99.15%	93.16%	99.52%	93.40%	87.92%	0.86	0.9
GCN-WITH-GR	99.00%	93.66%	99.30%	93.91%	85.88%	0.85	0.89
GR-ONLY	100.00%	100.00%	100.00%	100.00%	100.00%	0.96	1.0
HYBRID-GCN-GR	99.98%	99.92%	100.00%	99.93%	100.00%	0.96	1.0

Table A.12: Results DS-CO - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	98.21%	88.93%	99.41%	91.23%	90.42%	0.68	0.88
GCN-WITH-GR	98.17%	92.44%	99.33%	94.12%	91.35%	0.69	0.91
GR-ONLY	100.00%	100.00%	100.00%	100.00%	100.00%	0.78	1.0
HYBRID-GCN-GR	99.90%	99.57%	100.00%	99.67%	100.00%	0.77	1.0

Table A.13: Results DC-ST - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	85.62%	78.92%	79.72%	81.26%	67.60%	0.6	0.49
GCN-WITH-GR	87.06%	80.23%	77.62%	82.23%	72.18%	0.65	0.52
GR-ONLY	64.19%	87.14%	60.22%	90.91%	41.22%	0.36	0.24
HYBRID-GCN-GR	84.66%	77.66%	74.08%	78.79%	73.35%	0.65	0.49

Table A.14: Results DC-ST - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	96.65%	92.44%	96.54%	92.55%	75.15%	0.77	0.77
GCN-WITH-GR	97.13%	93.15%	97.17%	93.24%	81.93%	0.83	0.82
GR-ONLY	96.22%	96.45%	95.86%	96.55%	91.20%	0.89	0.88
HYBRID-GCN-GR	97.55%	94.19%	97.34%	94.24%	94.86%	0.92	0.9

Table A.15: Results DC-ST - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	82.84%	58.21%	81.40%	62.25%	62.66%	0.4	0.32
GCN-WITH-GR	84.67%	59.43%	84.74%	62.96%	70.69%	0.47	0.39
GR-ONLY	74.14%	70.61%	71.67%	76.38%	39.77%	0.22	0.2
HYBRID-GCN-GR	83.23%	58.85%	81.51%	61.29%	64.83%	0.43	0.34

Table A.16: Results DS-ST - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	88.65%	75.31%	90.12%	80.05%	65.01%	0.54	0.46
GCN-WITH-GR	88.29%	69.62%	88.18%	73.61%	60.95%	0.52	0.45
GR-ONLY	78.10%	85.29%	75.87%	89.90%	45.16%	0.39	0.39
HYBRID-GCN-GR	88.44%	78.28%	88.18%	82.45%	64.18%	0.55	0.48

Table A.17: Results DS-ST - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	97.53%	90.35%	97.65%	90.63%	77.64%	0.78	0.78
GCN-WITH-GR	97.14%	86.00%	97.37%	86.31%	76.10%	0.76	0.76
GR-ONLY	97.67%	95.91%	97.48%	96.04%	90.78%	0.88	0.88
HYBRID-GCN-GR	98.14%	94.98%	98.17%	95.12%	92.07%	0.89	0.89

Table A.18: Results DS-ST - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	89.29%	59.92%	89.69%	66.67%	54.63%	0.34	0.32
GCN-WITH-GR	88.50%	55.40%	89.30%	61.24%	54.86%	0.34	0.31
GR-ONLY	86.05%	70.69%	84.89%	76.29%	44.78%	0.27	0.29
HYBRID-GCN-GR	88.86%	65.38%	89.00%	71.22%	52.48%	0.32	0.32

Table A.19: Results DC-SST - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	77.04%	85.15%	72.33%	87.25%	51.36%	0.53	0.42
GCN-WITH-GR	86.00%	88.80%	75.46%	89.93%	65.67%	0.68	0.55
GR-ONLY	64.41%	100.00%	60.12%	100.00%	40.03%	0.44	0.32
HYBRID-GCN-GR	86.64%	90.26%	76.09%	91.54%	72.72%	0.74	0.6

A.20.

Table A.20: Results DC-SST - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	95.16%	94.27%	95.23%	95.00%	63.51%	0.66	0.66
GCN-WITH-GR	96.87%	93.75%	96.82%	93.84%	77.91%	0.83	0.82
GR-ONLY	96.24%	100.00%	95.83%	100.00%	90.59%	0.92	0.91
HYBRID-GCN-GR	97.75%	97.38%	97.52%	97.42%	94.81%	0.95	0.94

Table A.21: Results DC-SST - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	79.64%	79.10%	81.51%	81.07%	46.17%	0.48	0.41
GCN-WITH-GR	84.77%	84.53%	84.15%	85.89%	56.67%	0.6	0.54
GR-ONLY	76.09%	100.00%	73.48%	100.00%	40.14%	0.46	0.45
HYBRID-GCN-GR	85.84%	84.82%	84.12%	86.32%	66.96%	0.68	0.61

Table A.22: Results DS-SST - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	86.75%	78.94%	90.20%	82.34%	59.76%	0.59	0.53
GCN-WITH-GR	86.94%	81.95%	88.05%	85.23%	52.29%	0.55	0.51
GR-ONLY	85.51%	100.00%	83.80%	100.00%	47.43%	0.51	0.52
HYBRID-GCN-GR	86.63%	90.36%	85.76%	92.41%	53.47%	0.57	0.55

Table A.23: Results DS-SST - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	97.70%	91.17%	98.12%	91.33%	80.95%	0.84	0.83
GCN-WITH-GR	97.32%	92.59%	97.50%	92.73%	74.91%	0.8	0.8
GR-ONLY	98.33%	100.00%	98.18%	100.00%	91.27%	0.92	0.93
HYBRID-GCN-GR	98.39%	98.55%	98.35%	98.58%	92.09%	0.93	0.93

Table A.24: Results DS-SST - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	89.78%	79.43%	91.55%	81.55%	51.25%	0.54	0.53
GCN-WITH-GR	88.43%	77.39%	89.71%	79.93%	43.61%	0.49	0.47
GR-ONLY	90.02%	100.00%	89.16%	100.00%	47.98%	0.53	0.56
HYBRID-GCN-GR	90.37%	90.47%	90.17%	91.54%	52.84%	0.58	0.58

Table A.25: Results DC-STG - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	84.05%	87.42%	68.89%	88.05%	67.34%	0.69	0.46
GCN-WITH-GR	86.84%	88.13%	74.87%	88.85%	73.22%	0.75	0.57
GR-ONLY	61.45%	100.00%	57.08%	100.00%	32.89%	0.39	0.23
HYBRID-GCN-GR	85.76%	89.01%	72.98%	90.01%	74.32%	0.74	0.54

Table A.26: Results DC-STG - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	96.82%	95.02%	96.70%	95.05%	77.67%	0.83	0.82
GCN-WITH-GR	97.00%	94.95%	97.19%	95.00%	81.78%	0.86	0.84
GR-ONLY	96.16%	100.00%	95.76%	100.00%	88.91%	0.9	0.9
HYBRID-GCN-GR	97.54%	96.89%	97.74%	96.94%	94.15%	0.94	0.92

Table A.27: Results DC-STG - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	83.00%	82.97%	81.29%	83.78%	53.11%	0.57	0.46
GCN-WITH-GR	84.08%	82.17%	84.26%	83.29%	64.14%	0.66	0.55
GR-ONLY	73.35%	100.00%	70.59%	100.00%	23.07%	0.34	0.32
HYBRID-GCN-GR	83.20%	82.40%	83.76%	84.36%	59.44%	0.61	0.5

Table A.28: Results DS-STG - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	87.48%	83.28%	87.94%	86.99%	49.03%	0.52	0.48
GCN-WITH-GR	88.81%	85.70%	89.34%	88.01%	55.31%	0.59	0.55
GR-ONLY	85.90%	100.00%	84.24%	100.00%	42.60%	0.48	0.48
HYBRID-GCN-GR	87.86%	92.84%	87.65%	94.58%	50.12%	0.54	0.52

Table A.29: Results DS-STG - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	97.62%	93.31%	98.02%	93.50%	78.48%	0.82	0.82
GCN-WITH-GR	98.00%	95.32%	98.26%	95.40%	81.09%	0.85	0.85
GR-ONLY	98.39%	100.00%	98.24%	100.00%	89.98%	0.91	0.92
HYBRID-GCN-GR	98.42%	99.84%	98.30%	99.84%	90.12%	0.92	0.92

Table A.30: Results DS-STG - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	89.30%	79.63%	91.17%	83.10%	42.10%	0.46	0.44
GCN-WITH-GR	90.50%	81.88%	91.88%	83.86%	51.58%	0.56	0.54
GR-ONLY	90.36%	100.00%	89.49%	100.00%	40.04%	0.48	0.51
HYBRID-GCN-GR	90.54%	98.96%	89.84%	99.14%	40.84%	0.49	0.51

Table A.31: Results DS-ID - equal weighting

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	86.16%	79.43%	87.16%	82.54%	56.30%	0.58	0.52
GCN-WITH-GR	87.28%	81.77%	87.27%	84.35%	54.77%	0.58	0.53
GR-ONLY	85.33%	100.00%	83.57%	100.00%	46.70%	0.51	0.52
HYBRID-GCN-GR	87.44%	87.98%	87.03%	89.93%	57.87%	0.6	0.57

Table A.32: Results DS-ID - complete balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	97.39%	89.95%	97.94%	90.14%	79.71%	0.82	0.82
GCN-WITH-GR	97.59%	94.42%	97.64%	94.52%	76.93%	0.82	0.82
GR-ONLY	98.09%	100.00%	97.92%	100.00%	90.03%	0.91	0.92
HYBRID-GCN-GR	98.31%	98.54%	98.22%	98.57%	91.30%	0.92	0.92

Table A.33: Results DS-ID - reduced balanced

Type	Accuracy	Acc (yes)	Acc (no)	Precision	Recall	F1	MCC
GCN-NO-GR	88.91%	76.57%	90.72%	79.14%	51.67%	0.55	0.52
GCN-WITH-GR	89.64%	80.90%	89.87%	82.69%	49.02%	0.53	0.53
GR-ONLY	89.30%	100.00%	88.34%	100.00%	44.02%	0.5	0.53
HYBRID-GCN-GR	90.51%	91.09%	90.00%	92.14%	51.18%	0.56	0.57

References

[1] Sultan Alahmari, Tommy Yuan, and Daniel Kudenko. Reinforcement learning for abstract argumentation : A q-learning approach. *Adaptive and Learning Agents workshop (at AAMAS 2017)*, 2017.

[2] Sultan Alahmari, Tommy Yuan, and Daniel Kudenko. Policy generalisation in reinforcement learning for abstract argumentation. *CMNA Workshop Proceedings*, 2018.

[3] Sultan Alahmari, Tommy Yuan, and Daniel Kudenko. Reinforcement learning of dialogue coherence and relevance. *CMNA@PERSUASIVE 2019*, 2019.

[4] Mario Alviano. The pyglaf argumentation reasoner. *OpenAccess Series in Informatics*, 58:2–5, 2018.

[5] Pietro Baroni. An introduction to abstract argumentation. *EPCL Basic Training Camp*, 2013.

[6] Stefano Bistarelli, Fabio Rossi, and Francesco Santini. Not only size , but also shape counts : Abstract argumentation solvers are benchmark-sensitive. *Journal of Logic and Computation*, 2017.

[7] Bernhard Bliem and Stefan Woltran. On efficiently enumerating semi-stable extensions via dynamic programming on tree decompositions. *Frontiers in Artificial Intelligence and Applications*, 287:107–118, 2016.

[8] G Boella, D M Gabbay, L Van Der Torre, and S Villata. Support in abstract argumentation. volume 216, pages 111–122, 2010.

[9] Jennifer L Bonnet and Senta Sellers. The covid-19 misinformation challenge: An asynchronous approach to information literacy. *Internet Reference Services Quarterly*, 24:1–8, 2019.

[10] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A comparative study of ranking-based semantics for abstract argumentation. *Thirtieth AAAI Conference on Artificial Intelligence*, 2 2016.

[11] Benedikt Bünz and Matthew Lamm. Graph neural networks and boolean satisfiability. *arXiv:1702.03592[cs.AI]*, pages 1–9, 2017.

[12] Martin Caminada. An algorithm for stage semantics. *Frontiers in Artificial Intelligence and Applications*, 216:147–158, 2010.

[13] Martin Caminada and Mikolaj Podlaszewski. Admbuster : a benchmark example for (strong) admissibility. *ICCMA 17*, ICCMA 2017, 2017.

[14] Martin W A Caminada and Interdisciplinary Centre. Semi-stable semantics. *Journal of Logic and Computation*, 22:1207–1254, 2011.

[15] Claudette Cayrol and Marie Christine Lagasquie-Schiex. Bipolarity in argumentation graphs: Towards a better understanding. *International Journal of Approximate Reasoning*, 54:876–899, 9 2013.

[16] Federico Cerutti, Paul E Dunne, Massimiliano Giacomin, and Mauro Vallati. Computing preferred extensions in abstract argumentation: A sat-based approach. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8306 LNAI:176–193, 2014.

[17] Federico Cerutti, Sarah Gaggl, Matthias Thimm, and Johannes Wallner. Foundations of implementations for formal argumentation. *IfCoLog Journal of Logics and their Applications*, 4:2623–2705, 2017.

[18] Federico Cerutti, Massimiliano Giacomin, and Mauro Vallati. How we designed winning algorithms for abstract argumentation and which insight we attained. *Artificial Intelligence*, 276:1–40, 2019.

[19] Federico Cerutti, Matthias Thimm, and Mauro Vallati. An experimental analysis on the similarity of argumentation semantics. *Argument & Computation*, 11:269–304, 2020.

[20] Wenjing Chang, Yang Xu, and Shuwei Chen. A new rewarding mechanism for branching heuristic in sat solvers. *International Journal of Computational Intelligence Systems*, 12:334, 2019.

[21] Günther Charwat, Wolfgang Dvořák, Sarah A Gaggl, Johannes P Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation - a survey. *Artificial Intelligence*, 220:28–63, 2015.

[22] Oana Cocarascu and Francesca Toni. Argumentation for machine learning: A survey. *Computational Models of Argument*, 2016.

[23] Oana Cocarascu and Francesca Toni. Combining deep learning and argumentative reasoning for the analysis of social media textual content using small data sets. *Comput. Linguist.*, 44:833–858, 12 2018.

[24] Dennis Craandijk and Floris Bex. Deep learning for abstract argumentation semantics. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, {IJCAI-20}*, pages 1667–1673, 2020. Main track.

[25] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*, pages 1–22, 6 2019.

[26] David Devlin and Barry O'Sullivan. Satisfiability as a classification problem. *Proc. of the 19th Irish Conf. on Artificial Intelligence and Cognitive Science*, 2008.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[28] P M Dung, P Mancarella, and F Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171:642–674, 2007.

[29] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 9 1995.

[30] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. Assumption-based argumentation. *Argumentation in Artificial Intelligence*, pages 199–218, 2009.

[31] Paul E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artificial Intelligence*, 171:701–729, 2007.

[32] Paul E. Dunne and Michael Wooldridge. Complexity of abstract argumentation. *Argumentation in Artificial Intelligence*, pages 85–104, 2009.

[33] Wolfgang Dvořák, Matti Järvisalo, and Johannes P Wallner. Cegartix v2017-3-13: A sat-based counter-example guided argumentation reasoning tool. *ICCMA 17*, 2017.

[34] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Aspartix: Implementing argumentation frameworks using answer-set programming. *Lecture Notes in Computer Science*, 5366:734–738, 2008.

[35] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[36] Sarah Gaggl. Computational complexity of abstract argumentation. *Technical Report - TU Dresden*, 11 2013.

[37] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010*, 9, 2010.

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[39] Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. Semeval-2019 task 7: Rumoureval, determining rumour veracity and support for rumours. *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 845–854, 2019.

[40] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[41] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, pages 1025–1035, 2017.

[42] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 2017-Decem:1025–1035, 2017.

[43] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[44] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E Oliphant. Array programming with {NumPy}. *Nature*, 585:357–362, 9 2020.

[45] Md Rafiqul Islam, Shaowu Liu, Xianzhi Wang, and Guandong Xu. Deep learning for misinformation detection on online social networks: a survey and new perspectives. *Social Network Analysis and Mining*, 10:82, 2020.

[46] Richard Arnold Johnson and Dean W Wichern. *Applied multivariate statistical analysis*. Prentice Hall, 5. ed edition, 2002.

[47] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international sat solver competitions. *AI Magazine*, 33:89–92, 2012.

[48] Cezary Kaliszyk, François Chollet, and Christian Szegedy. Holstep: A machine learning dataset for higher-order logic theorem proving. *arXiv:1703.00426 [cs.AI]*, 3 2017.

[49] Cezary Kaliszyk, Henryk Michalewski, Josef Urban, and Mirek Olšák. Reinforcement learning of theorem proving. *Advances in Neural Information Processing Systems*, 2018-Decem:8822–8833, 2018.

[50] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *ICLR 2015*, pages 1–15, 2015.

[51] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 9 2019.

[52] Isabelle Kuhlmann and Matthias Thimm. Using graph convolutional networks for approximate reasoning with abstract argumentation frameworks: A feasibility study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11940 LNAI:24–37, 2019.

[53] Sebastian Kula, Michał Choraś, and Rafał Kozik. Application of the bert-based architecture in fake news detection bt. *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020)*, pages 239–249, 2021.

[54] Dilek Küçük and Fazli Can. Stance detection: A survey. *ACM Computing Surveys (CSUR)*, 53:1–37, 2020.

[55] John Lawrence and Chris Reed. Argument mining: A survey. *Computational Linguistics*, 45:765–818, 2019.

[56] Yann LeCun and Corinna Cortes. {MNIST} handwritten digit database, 2010.

[57] Henrique Lemos, Marcelo Prates, Pedro Avelar, and Luis Lamb. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. *IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019.

[58] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67:97–109, 5 2019.

[59] Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep network guided proof search. *arXiv:1701.06972 [cs.AI]*, 46:85–63, 2018.

[60] Jing Ma, Wei Gao, and Kam-Fai Wong. Rumor detection on {T}witter with tree-structured recursive neural networks. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, 7 2018.

[61] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[62] Lars Malmqvist. Solving abstract argumentation frameworks using monte carlo tree search. *UOY Computer Science Poster Session*, 2019.

[63] Lars Malmqvist. Afgcn: An approximate abstract argumentation solver. *ICCMA 2021*, 2021.

[64] Lars Malmqvist. Approximate solutions to argumentation frameworks with graph neural networks. *Online Handbook of Argumentation for AI*, page 32, 2021.

[65] Lars Malmqvist, Tommy Yuan, and Suresh Manandhar. Visualising argumentation graphs with graph embeddings and t-sne. *COMMA Workshop on Argument Visualization*, 2021.

[66] Lars Malmqvist, Tommy Yuan, and Peter Nightingale. Improving misinformation detection in tweets with abstract argumentation. *CMNA Workshop Proceedings 21*, 2937:40–46, 2021.

[67] Lars Malmqvist, Tommy Yuan, Peter Nightingale, and Suresh Manandhar. Determining the acceptability of abstract arguments with graph convolutional networks. *SAFA 2020 Workshop Proceedings @ COMMA*, 2672:47–56, 2020.

[68] Peter McBurney and Simon Parsons. Dialogue games for agent argumentation. *Argumentation in Artificial Intelligence*, pages 261–280, 2009.

[69] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. 2 2018.

[70] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representationsof words and phrases-and their compositionality. *Advances in neural information processing systems*, pages 3111–3119, 2013.

[71] Sanjay Modgil and Henry Prakken. The aspic+ framework for structured argumentation: a tutorial. *Argument & Computation*, 00:1–30, 2013.

[72] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:5425–5434, 2017.

[73] Alexander Nadel. Polarity and variable selection heuristics for sat-based anytime maxsat system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 12:17–22, 2020.

[74] Alice J Toniolo Timothy Norman, Anthony Etuk Federico Cerutti, Robin Wentao Ouyang Mani Srivastava, Nir Oren, Timothy A Dropps John Allen Honeywell, and Usa Paul Sullivan. Supporting reasoning with different types of evidence in intelligence analysis. *AAMAS '15: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 781–789, 2015.

[75] Peter Novák and Cees Witteveen. Context-aware reconfiguration of large-scale surveillance systems: Argumentative approach. *Argument and Computation*, 6:3–23, 1 2015.

[76] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-Augu:1105–1114, 2016.

[77] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS 2017*, 2017.

[78] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[79] N. Pezzotti. Dimensionality-reduction algorithms for progressive visual analytics. *PhD Thesis - Delft University of Technology*, 2019.

[80] Baroni Pietro, Martin Caminada, and Giacomin Massimilliano. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 00:1–24, 2004.

[81] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4501 LNCS:294–299, 2007.

[82] Henry Prakken and Giovanni Sartor. Law and logic: A review from an argumentation perspective. *Artificial Intelligence*, 227:214–245, 10 2015.

[83] Iyad Rahwan and Guillermo R Simari. *Argumentation in artificial intelligence.* Springer, 2009.

[84] Pau Riba, Andreas Fischer, Josep Lladós, and Alicia Fornés. Learning graph distances with message passing neural networks. *Proceedings - International Conference on Pattern Recognition*, 2018-Augus:2239–2244, 8 2018.

[85] Daniel A. Roberts, Sho Yaida, and Boris Hanin. *The principles of deep learning theory : an effective theory approach to understanding neural networks.* Cambridge University Press, 2022.

[86] O. Rodrigues, E. Black, M. Luck, and J. Murphy. On structural properties of argumentation frameworks: Lessons from iccma. *CEUR Workshop Proceedings*, 2171:22–35, 2018.

[87] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence: Current trends. *arXiv:2105.05330 [cs.AI]*, 5 2021.

[88] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, David L Dill, and Leonardo De Moura. Learning a sat solver from single-bit supervision. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–11, 2019.

[89] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[90] Matthias Thimm. Stochastic local search algorithms for abstract argumentation under stable semantics. *Frontiers in Artificial Intelligence and Applications*, 305:169–180, 2018.

[91] Matthias Thimm. Stochastic local search algorithms for abstract argumentation under stable semantics. *COMMA 18*, 2018.

[92] Matthias Thimm. Harper++: Using grounded semantics for approximate reasoning in abstract argumentation. *ICCMA 21*, 5 2021.

[93] Matthias Thimm, Pietro Baroni, Massimiliano Giacomin, and Paolo Vicig. Probabilities on extensions in abstract argumentation. *Proceedings of the 2017 International Workshop on Theory and Applications of Formal Argument (TAFA'17)*, 2017.

[94] Matthias Thimm and Serena Villata. The first international competition on computational models of argumentation : Results and analysis. *Artificial Intelligence*, 252:267–294, 2017.

[95] Petar Veličković, Arantxa Casanova, Pietro Liò, Guillem Cucurull, Adriana Romero, and Yoshua Bengio. Graph attention networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–12, 2018.

[96] Bart Verheij. Two approaches to dialectical argumentation: Admissible sets and argumentation stages. *Proc. NAIC 96*, pages 357–368, 1996.

[97] Douglas Walton. Argumentation theory: A very short introduction. *Argumentation in Artificial Intelligence*, pages 1–22, 2009.

[98] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. *Advances in Neural Information Processing Systems*, 2017-Decem:2787–2797, 2017.

[99] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv:1909.01315 [cs.LG]*, pages 1–18, 2019.

[100] Po-Wei Wang, Priya L. Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. *arXiv:1905.12149 [cs.LG]*, 5 2019.

[101] Stefan Woltran. Abstract argumentation – all problems solved ? *ECAI 2014*, pages 1–93, 2014.

[102] Michael Wooldridge. An introduction to multiagent systems [paperback]. *Chichester, England*, page 484, 2009.

[103] Z Wu, S Pan, F Chen, G Long, C Zhang, and P S Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2020.

[104] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks. *arXiv:2106.06218 [cs.LG]*, 11 2019.

[105] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Can graph neural networks help logic reasoning? *arXiv:1906.02111 [cs.LG]*, 6 2019.

[106] Kristijonas Čyras, Antonio Rago, Emanuele Albini, Pietro Baroni, and Francesca Toni. Argumentative xai: A survey. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, 5 2021.