

Heterogeneous Graph Neural Networks for Assumption-Based Argumentation

Anna Rapberger*, Fabrizio Russo*, Preesha Gehlot and Francesca Toni

Imperial College London, Department of Computing

Abstract

Assumption-Based Argumentation (ABA) is a powerful structured argumentation formalism, but exact computation of extensions under stable semantics is intractable for large frameworks. We present the first Graph Neural Network (GNN) approach to approximate credulous acceptance in ABA. To use GNNs, we represent ABA frameworks via a dependency graph representation that encodes atoms and rules as nodes and distinguishes support, derive and attack relations by heterogeneous edge labels. We propose two GNN variants—ABAGCN and ABAGAT—that stack residual heterogeneous convolution or attention blocks, respectively, to learn node embeddings. Our models are trained on the ICCMA2023 benchmark, augmented with synthetic ABAFs, with hyperparameters optimised via Bayesian search. Empirically, both ABAGCN and ABAGAT outperform a state-of-the-art GNN baseline that we adapt from the abstract argumentation literature, achieving node-level an F1 score up to 0.71 on the ICCMA instances. Finally, we develop a poly-time extension-reconstruction algorithm driven by our predictor: it reconstructs stable extensions with F1 above 0.85 on small ABAFs and maintains an F1 of about 0.58 on frameworks with 1,000 atoms. Our work opens new avenues for scalable approximate reasoning in structured argumentation.

Keywords

Assumption-Based Argumentation, Approximate Reasoning, Graph Neural Networks

1. Introduction

Computational argumentation provides formal tools for modelling defeasible reasoning over conflicting information. In the *abstract* setting, arguments are treated as opaque nodes and attacks as binary relations, giving rise to abstract argumentation frameworks (AFs) [1]. By contrast, *structured* formalisms expose the internal composition of arguments. Assumption-Based Argumentation (ABA) is a prominent structured framework, enabling transparent reasoning in domains such as decision support [2], planning [3], and causal discovery [4].

The building blocks of ABA are *assumptions*, which are the defeasible elements of an ABA framework (ABAF), and *inference rules*, which are used to construct *arguments* based on a given set of assumptions. *Conflicts* arise between assumption sets S and T if the conclusion of an argument constructed from S is the contrary of some assumption in T ; we say that S attacks T in this case. *Argumentation semantics* provide criteria which renders sets of assumptions as jointly acceptable: the so-called *extensions*. One of the most popular semantics is the stable semantics which accepts a set of assumptions S only if it has no internal conflicts and attacks every set not contained in S . While stable semantics neatly characterise acceptable assumption sets, they are hard to compute: membership testing for stable extensions is NP-complete [5]. Since real-world ABA frameworks can be very large [4], relying on exact approaches is often not viable.

In this work, we address this issue by using Graph Neural Networks (GNNs) [6] to predict the acceptability of assumptions. GNNs leverage the relationships between nodes and edges in a graph to learn representations that capture the its structure and allow to predict labels for nodes, edges or the whole graph. This specialised architecture allows GNNs to *amortise* the graph analysis and use its stored weights to predict feature of interest (e.g. node labels as in our case) in constant time after the initial network training, thus introducing significant efficiency at inference time.

Arg&App 2025: International Workshop on Argumentation and Applications, November 2025, Melbourne, Australia

*Corresponding authors, equal contributions.

✉ a.rapberger@imperial.ac.uk (A. Rapberger); fabrizio@imperial.ac.uk (F. Russo)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

While GNNs have been successfully used to approximate the acceptability of AFs, see, e.g., [7, 8, 9], their potential remains unexplored for ABA and any other structured argumentation formalism to date. In this work, we present the first GNN architecture to predict credulous acceptance for ABA under stable semantics. Our contributions are as follows:

- We introduce a faithful *dependency graph* encoding of ABAFs that distinguishes assumption, claim and rule nodes as well as support, derive, and attack edges.
- We develop two heterogeneous GNN architectures—ABAGCN and ABAGAT—composed of residual stacks of relation-specific convolutional or attention layers, respectively, enriched with learnable embeddings and degree-based features.
- We implement a training and evaluation pipeline combining the ICCMA2023 ABA benchmarks with 19,500 synthetically generated ABAFs. Our experiments demonstrate that both ABAGCN and ABAGAT outperform an AF-based GNN baseline based on state-of-the-art AFGCNv2, achieving node-level F1 up to 0.71 on small ABAFs and 0.65 overall.
- Driven by our credulous acceptance predictor, we develop a polynomial-time extension-reconstruction algorithm which reconstructs stable extensions with high fidelity. Our algorithm attains $F1 > 0.85$ on small frameworks and remains above 0.58 on ABAFs with 1,000 atoms while running $2.3\times$ faster than a state-of-the-art exact solver for the biggest (4,000 and 5,000 atoms) and most challenging instances.

2. Related work

Kuhlmann and Thimm [7] were the first to frame credulous acceptance in an AF as a graph-classification problem. They generated random AFs, labeled each argument’s acceptability using the exact CoQuiAAS solver [10], and trained a Graph Convolutional Network (GCN) on two feature sets: the raw adjacency matrix and adjacency augmented with each node’s in- and out-degree. Including degree information improved accuracy to $\sim 80\%$, with F1 scores below 0.4, and class-balancing during training enhanced detection of under-represented arguments. Despite only marginal effects from graph topology or dataset size, their GCN reduced runtime from over an hour (CoQuiAAS) to under 0.5s for a full test set.

Malmqvist et al. [9] then proposed AFGCN, which augments node features with 64-dimensional DeepWalk embeddings [11] before feeding them—alongside the AF’s adjacency matrix—through 4–6 residual GCN-dropout blocks. To address the skewed accept/reject ratio in real AFs, they exclude frameworks with very few accepted arguments and introduce a randomised per-epoch masking of labels, forcing the model to infer hidden acceptabilities. This scheme, rather than network depth or explicit class-balancing, accounts for most of AFGCN’s increase to $\sim 82\%$ accuracy on rebalanced data (62% reported for [7] on the same data). Malmqvist et al. [12] enrich argument representations with graph-level metrics (centralities, PageRank, colouring) and use four GCN-ReLU-dropout blocks plus an epoch-wise reshuffle-and-rebalance routine and a pre-check on the grounded extension for faster and more accurate inference.

Cibier and Mailly [8] optimise AFGCNv2 by reimplementing AF parsing and graph-metric computation in Rust, slashing preprocessing time and memory usage. They evaluate the original AFGCNv2 against two variants—adding five semantic features to random features, or retaining only 11 “meaningful” features—and assess each with and without dropout. Their results show that pruning degrades accuracy, and that the no-dropout version of “feature-enhanced” version achieves the best overall performance ($\sim 83\%$ accuracy vs $\sim 75\%$ reported for AFGCNv2 on the same data), highlighting the importance of rich semantic features and careful dropout placement. Finally, they introduce AFGAT, a three-layer GATv2 (an updated attention mechanism [13] with multi-head attention), which surpasses all GCN variants obtaining $\sim 87\%$ on the same benchmark data from ICCMA.

Craandijk and Bex [14] propose AGNN, which learns argument embeddings through iterative message passing that internalises conflict-freeness and defense, then uses the resulting acceptance probabilities to drive a constructive backtracking algorithm for extension enumeration. We build on this enumeration strategy in our experiments in Section 5.

We adopt AFGCNv2 [12] as our baseline¹ but, inspired by the results in [8], we replace its handcrafted features with learnable embeddings and experiment with both GCNs and GAT layers in our final models. Additionally, we handle class imbalance via loss weighting rather than mini-batch rebalancing and disable the grounded-extension heuristic to test the models' predictive accuracy on the entire frameworks rather than on the non-grounded portion of it only. In contrast to the AF-focused solvers discussed, we present the first native approximate solver for ABAFs, combining an enriched and faithful graphical representation with heterogeneous GCN and GAT-based prediction of assumption acceptability incorporated in an AGNN-inspired extension reconstruction.

3. Preliminaries

We recall the relevant elements of abstract and assumption-based argumentation, as well as GNNs.

3.1. Abstract Argumentation

An argumentation framework (AF) [1] is a directed graph $F = (A, R)$ where A are arguments and $R \subseteq A \times A$ are *attacks*. For $x, y \in A$, if $(x, y) \in R$ we say x *attacks* y ; $E \subseteq A$ *defends* x if it attacks each attacker of x ; E is *conflict-free* ($E \in cf(F)$) iff it does not attack itself. A semantics σ is a function that assigns each AF a set of sets of arguments, so-called *extensions*. Here, we focus on stable semantics.

Definition 3.1. Let $F = (A, R)$ be an AF. A set $E \subseteq A$ is *stable* ($E \in stb(F)$) iff E is *conflict-free* and *attacks each* x for every $x \in A \setminus E$.

3.2. Assumption-based Argumentation

We assume a *deductive system*, i.e. a tuple $(\mathcal{L}, \mathcal{R})$, where $\mathcal{L}$ is a set of atoms and $\mathcal{R}$ is a set of inference rules over $\mathcal{L}$. A rule $r \in \mathcal{R}$ has the form $a_0 \leftarrow a_1, \dots, a_n$, s.t. $a_i \in \mathcal{L}$ for all $0 \leq i \leq n$; $head(r) := a_0$ is the *head* and $body(r) := \{a_1, \dots, a_n\}$ is the (possibly empty) *body* of r .

Definition 3.2. An ABA framework (ABAF) [15] is a tuple $(\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$, where $(\mathcal{L}, \mathcal{R})$ is a deductive system, $\mathcal{A} \subseteq \mathcal{L}$ a set of assumptions, and $\neg : \mathcal{A} \rightarrow \mathcal{L}$ a contrary function.

We fix an arbitrary ABAF $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$ below. The ABAF $\mathcal{D}$ is *flat* iff $head(r) \notin \mathcal{A}$ for all $r \in \mathcal{R}$. In this work we focus on *flat* and *finite* ABAFs, i.e. $\mathcal{L}$ and $\mathcal{R}$ are finite.

An atom $p \in \mathcal{L}$ is *tree-derivable* from assumptions $S \subseteq \mathcal{A}$ and rules $R \subseteq \mathcal{R}$, denoted by $S \vdash_R p$, if there is a finite rooted labeled tree t s.t. i) the root of t is labeled with p , ii) the set of labels for the leaves of t is equal to S or $S \cup \{\top\}$, and iii) for each node v that is not a leaf of t there is a rule $r \in R$ such that v is labeled with $head(r)$ and labels of the children correspond to $body(r)$ or $\top$ if $body(r) = \emptyset$. We write $S \vdash p$ iff there exists $R \subseteq \mathcal{R}$ such that $S \vdash_R p$.

Let $S \subseteq \mathcal{A}$. By $\bar{S} := \{\bar{a} \mid a \in S\}$ we denote the set of all contraries of S . The set S *attacks* a set $T \subseteq \mathcal{A}$ if there are $S' \subseteq S$ and $a \in T$ s.t. $S' \vdash \bar{a}$; if S attacks $\{a\}$ we say S *attacks* a . S is *conflict-free* ($S \in cf(\mathcal{D})$) if it does not attack itself. We define stable ABA semantics.

Definition 3.3. Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$ be an ABAF. A set $S \subseteq \mathcal{A}$ is *stable* ($S \in stb(\mathcal{D})$) iff S is *conflict-free* and *attacks each* $\{x\}$ for every $x \in \mathcal{A} \setminus S$.

As for AFs, we call a set $S \in stb(\mathcal{D})$ an *extension*. An assumption $a \in \mathcal{A}$ is *credulously accepted* w.r.t. stb in an ABAF $\mathcal{D}$ iff there is some $S \in stb(\mathcal{D})$ with $a \in S$.

Example 3.4. Consider an ABAF $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$ with atoms $\mathcal{L} = \{a, b, c, d, p, \bar{a}, \bar{b}, \bar{c}, \bar{d}\}$, assumptions $\mathcal{A} = \{a, b, c, d, e\}$, their contraries $\bar{a}, \bar{b}, \bar{c}, \bar{d}$, respectively, and rules $r_1, \dots, r_4$, respectively:

$$\bar{c} \leftarrow a, d \qquad p \leftarrow b \qquad \bar{d} \leftarrow c \qquad \bar{a} \leftarrow p, c$$

¹Available at <https://github.com/lmlearning/AFGraphLib/tree/main/AFGCNV2>. We could not use the improved versions from [8] as no weights were made available.

We obtain two stable sets here: $\{b, c\}$ and $\{a, b, d\}$. Indeed, $\{b, c\}$ attacks a and d since $\{b, c\} \vdash \bar{a}$ and $\{c\} \vdash \bar{d}$; and $\{a, b, d\}$ attacks c since $\{a, d\} \vdash \bar{c}$.

Viewing tree derivations as arguments, an ABAF induces an AF as follows [16, 17].

Definition 3.5. The associated AF $F_{\mathcal{D}} = (A, R)$ of an ABA $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$ is given by $A = \{S \vdash p \mid \exists R \subseteq \mathcal{R} : S \vdash_R p\}$ and attack relation $(S \vdash p, S' \vdash p') \in R$ iff $p \in \bar{S}'$.

Semantics of AFs and ABAFs are closely related [16].

Proposition 3.6. Given an ABA $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$ and its corresponding AF $F = (A, R)$. If $E \in \text{stb}(F)$ then $\bigcup_{S \vdash p \in E} S \in \text{stb}(\mathcal{D})$; if $S \in \text{stb}(\mathcal{D})$ then $\{S' \vdash p \mid \exists S' \subseteq S, R \subseteq \mathcal{R} : S' \vdash_R p\} \in \sigma(F)$.

Note that the AF instantiation can be exponential in the size of the given ABAF $\mathcal{D}$. To tackle this issue, Lehtonen et al. [18] propose a poly-time preprocessing technique that yields an ABAF so that the resulting AF is polynomial in the size of $\mathcal{D}$. Their procedure handles the derivation circularity and flattens out the nested argument construction; they show that this construction preserves semantics under projection. We refer the interested reader to [18] for more details on the construction.

3.3. Graph Neural Networks

A neural network (NN) is a parametrised function that maps an input vector $\mathbf{x} \in \mathbb{R}^n$ to an output through successive layers of neurons. We set the input neurons as $\mathbf{h}^{(0)} = \mathbf{x}$ and denoting by $\mathbf{h}^{(l)}$ the row-vector of neurons at layer l , each neuron at layer l computes

$$h_i^{(l+1)} = f((\mathbf{w}_i^{(l)})^\top \mathbf{h}^{(l)} + b_i^{(l)}),$$

where $\mathbf{w}_i^{(l)}$ and $b_i^{(l)}$ are the learnable weight-vector and bias for neuron i in layer l , and f is a (possibly non-linear) activation function [19] and $\cdot$. In supervised learning, the NN is trained by comparing predictions to ground-truth labels via a loss function (e.g. cross-entropy), and parameters are optimised by back-propagation and gradient descent.

Graph neural networks (GNNs) generalise NNs to consume graph-structured data $G = (V, E)$, by jointly leveraging a node-feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ and an adjacency matrix $\mathbf{A}$. Through repeated neighbourhood-aggregation steps—each propagating (also known as *message-passing*) and combining information from a node's neighbours—a GNN learns *embeddings* that encode both local topology and node attributes [20]. These embeddings can serve tasks such as node classification, where a final projection and sigmoid, with a threshold (possibly) tuned for class imbalance, yield per-node labels. Popular aggregation schemes include Graph Convolutional Networks (GCNs) [21] and Graph Attention Networks (GATs) [22].

GCNs perform matrix-based updates across all nodes simultaneously and treat every neighbour equally. GCNs update node representations by aggregating feature information from each node and its neighbours. At each layer l , the representation matrix $\mathbf{H}^{(l+1)}$ is computed from the previous layer $\mathbf{H}^{(l)}$ and the graph structure as:

$$\mathbf{H}^{(l+1)} = \sigma \left(\Delta^{-1/2} \tilde{\mathbf{A}} \Delta^{-1/2} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right),$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix of the graph with added self-loops (so each node includes its own features in the aggregation), Δ is the diagonal degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{W}^{(l)}$ is a learnable weight matrix for layer l , and σ is a (non-linear) activation function. The inclusion of self-loops ensures that nodes can retain and refine their own features across layers, rather than relying solely on neighbouring information. The input to the first layer, $\mathbf{H}^{(0)}$, is typically the initial node feature matrix $\mathbf{X}$.

GATs compute learned attention coefficients [23] to weight each neighbour’s contribution differently. Unlike GCNs, which treat all neighbouring nodes equally during aggregation, GATs introduce learnable attention mechanisms to assign different weights to different neighbours. Specifically, for a node i , attention coefficients α_{ij} are computed between i and each of its neighbours j using shared attention:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j]))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k]))},$$

where $\mathbf{W}$ is a learnable weight matrix, $\mathbf{a}$ is a learnable attention vector, $[\cdot \parallel \cdot]$ denotes vector concatenation, and $\mathcal{N}(i)$ is the set of neighbours of node i . LeakyReLU is a non-linear activation function that allows a small negative slope (typically $\epsilon = 0.2$) for negative inputs, helping to avoid dead neurons and improve gradient flow. The updated representation of node i is then computed as a weighted sum of the transformed features of its neighbours: $\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right)$. To stabilise the learning process, GATv2 [13] employs multi-head attention: multiple independent attention mechanisms are applied in parallel, and their outputs are concatenated (in intermediate layers) or averaged (in the output layer).

When modelling heterogeneous graphs—where edges carry distinct semantics—one typically employs relation-specific transformations, e.g. via a Heterogeneous Graph Convolution (HGC) module [24], and adds residual connections and normalisation between layers to stabilise training. To prevent overfitting, dropout [25] can be employed after each layer, randomly setting to zero a proportion of the weights, and early stopping to terminate training when the validation loss does not improve for a number of training steps greater than a patience parameter.

4. Predicting Assumptions Acceptance with GNNs

Here we detail our proposed GNN architecture to predict credulous acceptance of assumptions in an ABA framework. The core and novel component is the *dependency graph*, a faithful graph-based representation of an ABAF, that guides the GNN learning of the ABA relationships (Section 4.1). The neural machinery adopted to learn to classify the assumption nodes of the dependency graph as *credulously accepted* (IN) or *rejected* (OUT) w.r.t. stable semantics is then detailed in Section 4.2.

4.1. Dependency Graph

Our proposed graph representation contains a node for each atom (assumptions and non-assumptions) and a node for each rule. Also, it contains three distinct edge types. *Support* and *derive edges* connect atom nodes to rule nodes; support edges are labelled $+$, they connect body elements with the respective rule node; derive edges are labelled $\triangleright$; they connect rule node to their respective head elements. *Attack edges* are labelled $-$, they connect contraries and their assumptions.

Definition 4.1. Let $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$ be an ABAF. The dependency graph $G_D = (V, E, l)$ of D is an edge-labelled graph with nodes $V = \mathcal{L} \cup \mathcal{R}$, edges

$$E = \{(p, r) \mid r \in \mathcal{R}, p \in \text{body}(r)\} \cup \{(r, p) \mid r \in \mathcal{R}, p \in \text{head}(r)\} \cup \{(p, a) \mid a \in \mathcal{A}, \bar{a} = p\},$$

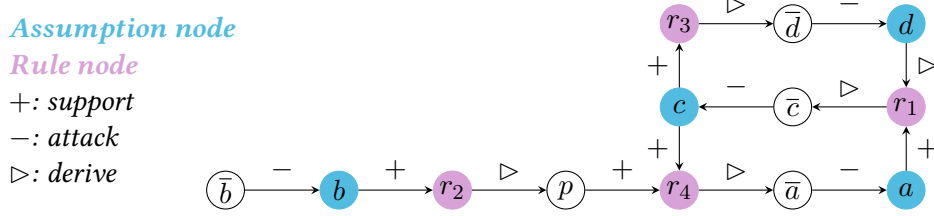
and edge labellings, for $e \in E$,

$$l(e) = \begin{cases} +, & \text{if } e = (p, r), r \in \mathcal{R}, p \in \text{body}(r) \\ \triangleright, & \text{if } e = (r, p), r \in \mathcal{R}, p \in \text{head}(r) \\ -, & \text{if } e = (p, a), a \in \mathcal{A}, \bar{a} = p \end{cases}$$

Observation 4.2. The dependency graph G_D is unique for each $\mathcal{D}$, i.e., for every two ABAFs $\mathcal{D} \neq \mathcal{D}'$, it holds that $G_D \neq G_{D'}$.

Since $G_{\mathcal{D}}$ contains information about all assumptions, contraries, and rules of a given ABAF $\mathcal{D}$, the ABAF can be fully recovered from $G_{\mathcal{D}}$, thus the semantics are preserved, as for the AF instantiation. In contrast to the AF instantiation, the construction of the dependency graph requires a single pass over the atoms and rules of $\mathcal{D}$.

Example 4.3. The dependency graph of the ABAF $\mathcal{D}$ from Example 3.4 is shown below:



We can use $G_{\mathcal{D}}$ to check that b and c jointly derive $\bar{a}$, using rules r_2 and r_4 .

Remark 4.4. Rapberger et al. [26] define a dependency graph for ABA in the spirit of dependency graphs for logic programming [27, 28]. However, their dependency graph does not fully preserve the structure of the ABAF, thus, it is not possible to extract the extensions from a given dependency graph using their approach. In brief, the dependency graph from [26] does not include rule nodes; positive edges (p, q) are introduced whenever p is contained in some rule body of a rule with head q . For example, the rule $(\bar{a} \leftarrow p, c)$ from Example 4.3 would yield the same edges as having two rules $(\bar{a} \leftarrow p)$, $(\bar{a} \leftarrow c)$. In contrast, our dependency graph is unique for every ABAF and carries enough information to preserve the semantics.

4.2. Neural Architecture

Our GNN operates on the dependency graph's adjacency matrix $\mathbf{W} \in \{0, 1\}^{|V| \times |V|}$ and an initial node-feature matrix $\mathbf{F} \in \mathbb{R}^{|V| \times 2}$, where each row F_i contains the in- and out-degrees of node i for each node type (Assumptions $\mathcal{A}$, non-assumptions (or claims) $\mathcal{C} = |\mathcal{L} \setminus \mathcal{A}|$, and rule nodes $\mathcal{R}$). In addition, we maintain a learnable embedding matrix $\mathbf{L} \in \mathbb{R}^{|V| \times d_e}$, whose entries are optimised jointly with the rest of the GNN. We show a diagram of our model architecture in Figure 1.

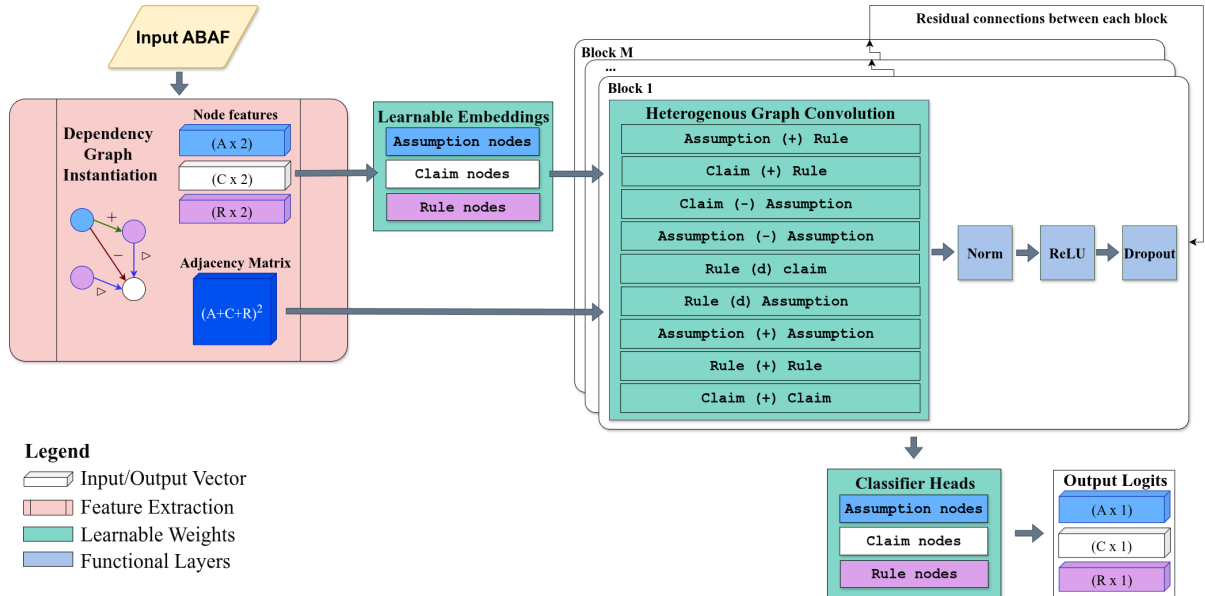


Figure 1: Diagram of the model architecture, including both the feature extraction (using the dependency graph described in Section 4.1) and the GNN described in Section 4.2. The Heterogeneous Graph Convolution (HGC) module would apply Convolutional Layers for ABAGCN, or Attention Layers for ABAGAT.

The core backbone of the GNN consists of M blocks. In block m :

- A HGC layer performs relation-specific neighbourhood aggregation on the current embeddings $H^{(m-1)}$, using either GCN or GAT convolutional kernels (see Section 3 for details).
- The aggregated output is passed through functional layers for stability and regularisation: we use layer-normalisation, a ReLU activation, and dropout with rate δ .
- A residual connection aggregates the block input $H^{(m-1)}$ to its output, yielding $H^{(m)}$.

This design allows distinct transformations per edge type while preserving gradient flow in deep stacks. After M blocks, we extract the rows of $H^{(M)}$ corresponding to assumption, claims and rule nodes and apply a separate linear classifier heads to produce logits $z \in \mathbb{R}^{A+C+R}$, with $A = |\mathcal{A}|$, $C = |\mathcal{C}|$, $R = |\mathcal{R}|$.

At inference time, a sigmoid activation followed by a threshold τ (tuned on validation data) yields the predicted probability of credulous acceptance for each assumption.

We train by minimising a weighted binary cross-entropy loss on the assumption logits, using the Adam optimiser with learning rate λ . Class weights compensate for the imbalance between accepted and rejected assumptions (see Table 2). All hyperparameters—including the number of layer blocks M , the embedding dimension d_e , hidden dimension of each convolution layer, the dropout rate δ , the learning rate λ , the batch size, class-weight multiplier, and classification threshold τ —are chosen via Bayesian optimisation [29] with 3-fold cross-validation, as implemented in [30].

Results of the optimisation are provided in Table 1, where we note that the GAT layer achieves a slightly higher validation F1 score (see Section 5 for more details on data and evaluation). Interesting observations from Table 1 are that the optimal dropout rate, batch sizes and early stopping thresholds reflect the role of attention in reducing the number of parameters (higher dropout) and the number of examples needed (lower batch) but higher tendency to over-fitting (higher patience needed). We also note that the number of layers was strongly positively correlated with better performance, indicating that more than 10 layers would probably improve performance. We leave this as future work because of time and compute constraints.

Table 1

Tested hyperparameter ranges and best configurations for GCN and GAT

Parameter	Values tested	GCN (best)	GAT (best)
Validation F1 Score	—	0.6682	0.6709
Embedding dimension	16, 32, 64, 128, 256	32	64
Hidden dimension	32, 64, 128, 256, 512	32	64
Number of layers blocks	2–10	10	10
Dropout rate	0–0.5 (uniform)	0.0294	0.2198
Learning rate	10^{-2} – 10^{-4} (uniform)	0.0086	0.0066
Batch size	16, 32, 64, 128	128	64
Class imbalance weight	1.25, 1.5, 1.75, 2.0, 2.25	2.25	1.75
Classification threshold	0.20–0.85 (step 0.05)	0.50	0.45
Early stopping patience	10, 30, 50, 70, 90	10	30

5. Empirical Evaluation

In this section we assess the effectiveness of our proposed GNN models—both the convolutional (ABAGCN) and attention-based (ABAGAT) variants—against our AFGCNv2-based baseline on two tasks over ABAFs under stable semantics:

1. **Credulous acceptance classification:** predicting, for each assumption, whether it belongs to any stable extension.
2. **Extension reconstruction:** recovering the full set of accepted assumptions (i.e. a predicted extension) and comparing it to the ground-truth stable extensions.

Table 2

Parameter ranges for ICCMA benchmarks and additional synthetic data

Parameter	ICCMA	Generated
Acceptance Rate (with at least one extension)	36.7% (60.3%)	32.5% (53.4%)
Number of atoms	25, 100, 500, 2000, 5000	25, 50, 75, 100, 250, 500, 750, 1000, 2000, 3000, 4000, 5000
Assumption proportion	10%, 30%	25%, 40%, 50%, 60%, 75%
Max. rules per derived atom	5, 10	2, 4, 8, 16
Max. rule-body length	5, 10	2, 4, 8, 16
Cycle cap	—	0.01, 0.03, 0.05, 0.07, 0.09, 1.0
Total instances (post-filtering)	400 (380)	48 000 (19 500)

For the first task we report node-level precision, recall, F1 and accuracy. For the second task we treat each stable extension as a set and measure how well our predicted sets match ground truth using extension-level F1 (equivalently, the F1 of the predicted versus true assumption sets).

5.1. Data

ICCMA 2023 included an ABA track—but not an approximate ABA one—whose benchmark comprised of 400 ABAFs defined by four parameters: number of atoms, assumption proportion, maximum rules per atom, and maximum rule-body length. Ground-truth labels are obtained by running ASPForABA [31]² on every test instance with timeout threshold 10 min, yielding 380 benchmark ABAFs with exact annotations within the ICCMA data.

To obtain a larger training corpus, we used the ABAF generator from [31] to produce 48,000 flat ABAFs (47,794 after timeout filtering when producing labels), extending the parameter ranges. Parameters are shown in Table 2. To match the original assumptions acceptance rate (36.7% overall, 60.3% within ABAFs with at least one stable extension), we curated a subset of 19,500 ABAFs (13,000 with at least one accepted assumption, 6,500 with none), yielding 32.5% overall acceptance and 53.4% within ABAFs with at least a solution.

Finally, we applied a stratified 75:25 train–test stratified over three groups—small ICCMA (25–100 atoms), full ICCMA, and our generated data. The stratification followed from seeking a fair comparison against AFGCNv2, which cannot handle ABAFs with more than 100 atoms. All results are therefore shown on the unseen test sets from these groups: ICCMA small, ICCMA (full) and ICCMA+Generated.

5.2. Baseline

Our baseline implements a three-stage pipeline that reduces ABAF credulous acceptance under stable semantics to an AF classification task:

1. **ABAF-AF translation:** Given an input ABAF, we use the method from [18] and implemented in the AcBar toolkit [32]³ to convert the input ABAF into a (poly-sized) AF in polynomial time.
2. **Argument classification via AFGCNv2:** The generated AF is represented by its adjacency matrix and graph-level features, then fed into the AFGCNv2 solver (see Section 2) using pre-trained weights for credulous acceptance under stable semantics. To ensure a fair comparison with other models, we disable its grounded-extension pre-check and apply a classification threshold—tuned on a held-out validation set—to the sigmoid output so that any argument with score above the threshold is marked as accepted.
3. **Element acceptability recovery:** Finally, we derive assumption acceptance in the original ABAF: an assumption a is declared accepted if the argument $\{a\} \vdash a$ is accepted.

²Available at <https://bitbucket.org/coreo-group/aspforaba>.

³Available at <https://bitbucket.org/lehtonen/acbar>

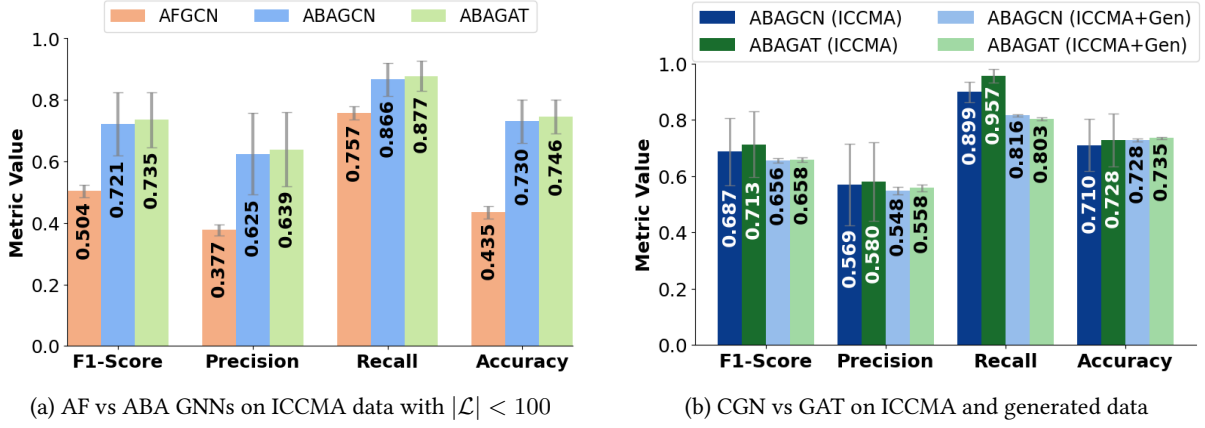


Figure 2: Model Comparison according to F1, Precision, Recall and Accuracy on different cuts of data: small ICCMA ABAFs (with less than 100 elements) in panel (a) to be able to compare to the AFGCN baseline; Comparison between all ICCMA ABAFs and the test set from our augmented training data in panel (b).

This baseline thus leverages existing AF-based GNN techniques for assumption-based frameworks, providing a direct point of comparison for our native ABAF solver.

5.3. Credulous Acceptance Classification

We first evaluate how well each model predicts assumption-level acceptance under stable semantics. Figure 2a shows that both dependency-graph GNNs substantially outperform the AF baseline on the small ICCMA dataset ($|\mathcal{L}| < 100$). F1 increases from 0.5 (AFGCN) to 0.72 (ABAGCN) and 0.74 (ABAGAT); recall climbs by about 10 points; precision improves by about 25 points; and accuracy rises by over 30 points. The attention-based ABAGAT achieves the highest results across all metrics on this test set.

In Figure 2b we examine the effect of augmenting the ICCMA training data with our generated instances. Compared to the full ICCMA test set, both ABAGCN and ABAGAT see modest gains in accuracy but incur a notable drop in precision and recall, resulting in a decrease in F1. This pattern indicates that synthetic augmentation possibly produced more challenging ABAFs than the ones in the competition. Crucially, despite this worsening in performance, both GNN variants maintain strong performance across the held-out ICCMA and generated test split, demonstrating robust generalisation to larger and more diverse ABA frameworks.

Comparing the small ICCMA results in Figure 2a to the (full) ICCMA bars in Figure 2b, F1 dips by under one percentage point, precision by roughly two points and accuracy by about one point on the full dataset, while recall actually climbs by two to six points, for ABAGCN and ABAGAT, respectively. In other words, on larger ABAFs both GNNs trade a bit of overall and positive-class fidelity for stronger coverage of accepted assumptions.

5.4. Extension Reconstruction

For our second experiment, we test a poly-time algorithm that uses our approximate ABA models to calculate a stable extension. The algorithm works as follows, see the sketch in Algorithm 1. First, given an ABAF $\mathcal{D}$, we use the credulous acceptance prediction obtained from, e.g., ABAGAT to pick an assumption a^* (line 3). Intuitively, this pick means that a^* is *accepted* in $\mathcal{D}$. To make our next guess, we modify the ABAF $\mathcal{D}$ accordingly so that the extensions of the modified ABAF $\mathcal{D}'$ (under projection) correspond to the extensions of $\mathcal{D}$ that contain a^* (line 4). Since our prediction is imperfect, this modification may have introduced some inconsistencies in $\mathcal{D}'$, which we check in line 5. If the check passes, we add a^* to our extension $\mathcal{E}$. We outline $\text{MODIFY}(\mathcal{D}, a^*)$ in more detail. When removing from the ABAF an assumption a^* predicted IN, we want to ensure that the modified ABAF $\mathcal{D}'$ satisfies

$$\{S \cap \mathcal{A} \mid S \in \text{stb}(\mathcal{D}')\} = \{S \setminus \{a^*\} \mid S \in \text{stb}(\mathcal{D}), a^* \in S\}, \quad (1)$$

Require: ABAF $\mathcal{D} = (\mathcal{L}, \mathcal{R}, \mathcal{A}, \neg)$, predictor $\mathcal{M}$

```

1: Initialize extension  $\mathcal{E} \leftarrow \emptyset$ 
2: while  $\mathcal{A} \neq \emptyset$  do
3:    $a^* \leftarrow \arg \max_{a \in \mathcal{A}} P_{\mathcal{M}}(a)$ 
4:    $\mathcal{D}' \leftarrow \text{MODIFY}(\mathcal{D}, a^*)$ 
5:   if  $\text{ISCOFLICTING}(\mathcal{D}')$  then
6:     break
7:   else
8:      $\mathcal{E} \leftarrow \mathcal{E} \cup \{a^*\}$ 
9: return  $\mathcal{E}$ 

```

Algorithm 1: Extension reconstruction sketch

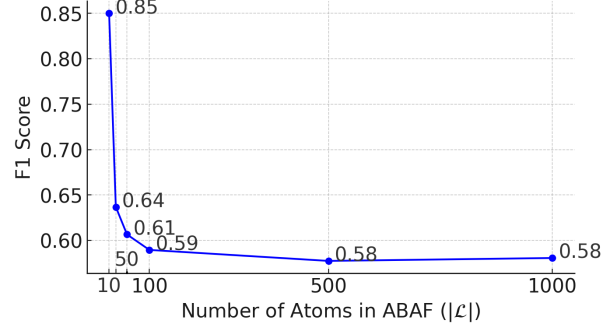


Figure 3: F1 score of the extension by ABAF size.

that is, the stable extensions of $\mathcal{D}'$ projected on $\mathcal{A}$ correspond to the extensions of $\mathcal{D}$ that contain a^* . We project onto $\mathcal{A}$ because the modification requires the introduction of some dummy-assumptions to ensure that rules that derive the contrary of a^* act as constraints. Concretely, given an ABAF $\mathcal{D}$ and an assumption a^* , $\text{MODIFY}(\mathcal{D}, a^*)$ performs the following steps:

1. We remove the assumption a^* and all occurrences of it; that is, we let $\mathcal{A}' = \mathcal{A} \setminus \{a^*\}$ and we replace any rule r with $r' = \text{head}(r) \leftarrow \text{body}(r) \setminus \{a^*\}$.
2. We modify each rule r with $\text{head}(r) = \overline{a^*}$: for each such rule r , we introduce a new dummy assumption d_r and replace r with $r' = \overline{d_r} \leftarrow \text{body}(r) \cup d_r$. This modification ensures that not all body elements of r can be true (accepted or derived) at the same time.⁴
3. If a^* is the contrary of another assumption c , then c is rejected. We remove c and any rule r with $c \in \text{body}(r)$ from the ABAF.
4. Lastly, we remove all facts from $\mathcal{D}$: for each rule r with $\text{body}(r) = \emptyset$, we remove r from the rule set and we remove $p = \text{head}(r)$ from the body of each rule r' , i.e., we replace r' with $r'' = \text{head}(r') \leftarrow \text{body}(r') \setminus \{p\}$. If p is the contrary of an assumption c , the assumption is attacked and we proceed as in Step 3. We repeat this process until there are no remaining facts.

It can be checked that if a^* is credulously accepted with respect to stable semantics, then the algorithm $\text{MODIFY}(\mathcal{D}, a^*)$ satisfies the statement in Equation 1. Otherwise, in case a^* is not credulously accepted in $\mathcal{D}$, Algorithm 1 will encounter a rule of the form $\overline{d_r} \leftarrow d_r$ which indicates an inconsistency in $\mathcal{D}$. We perform this inconsistency check in $\text{ISCOFLICTING}(\mathcal{D}')$ by checking the rules of the ABAF $\mathcal{D}$ (line 5 of Algorithm 1) and return the partially computed set as soon as we encounter an inconsistency.

Note that $\text{MODIFY}(\mathcal{D}, a^*)$ runs in polynomial time; naively, the computation requires a loop over all rules in $\mathcal{R}$, for each $p \in \mathcal{L}$; thus, Algorithm 1 is in P.

Figure 3 illustrates that our constructive extension reconstruction method achieves very high F1 (0.85) on small ABAFs (10 atoms), but performance drops sharply to around 0.60 by 50–100 atoms and then plateaus at approximately 0.58 for larger frameworks. This decline reflects how early misclassifications by the GNN propagate through subsequent modification steps, reducing both precision and recall. Nonetheless, sustaining an F1 above 0.5 even on ABAFs with 1,000 atoms indicates that the approach remains moderately robust at scale, highlighting opportunities for future work on error-correction or lookahead strategies to mitigate this degradation.

We also compare runtimes on the 15 most challenging ABAFs (4,000–5,000 atoms) from our generated dataset, held out in the test split. The exact ASPForABA solver averaged 435s per instance, whereas our approximate extension reconstruction ran in 192s on average— $2.3\times$ faster—while still achieving F1 of 0.68 and accuracy of 0.77 on those large frameworks. These results demonstrate that GNN-driven approximate reasoning can yield substantial speed-ups over exact methods, maintaining a similar level predictive performance as compared to the easier single assumption prediction task.

⁴This modification resembles a constraint, i.e., a rule with empty head in logic programming, cf. [33].

6. Conclusion

We have demonstrated that heterogeneous GNNs over our dependency-graph encoding can accurately and efficiently approximate credulous acceptance in ABA under stable semantics. Both ABAGCN and ABAGAT outperform an AF-based GNN baseline—achieving node-level F1 of 0.65 overall and up to 0.71 on small frameworks—while our poly-time extension-reconstruction procedure reconstructs stable sets with F1 greater than 0.85 on small ABAFs and maintains F1 of about 0.58 at scales of 1,000 atoms. Crucially, on the 4,000–5,000-atom ABAFs where exact ASPForABA averaging 435s, our approximate enumeration runs in 192s (2.3× faster) with F1 of 0.68, underscoring substantial runtime gains without sacrificing predictive quality. These findings show that neural approximation can bridge the gap between accuracy and tractability in structured argumentation.

Future work includes integrating lightweight symbolic checks to boost precision (e.g. grounded semantics as done in [12]), extending the approach to other semantics, including admissible, complete, and preferred, where GNNs may be able to exploit local structural patterns even more efficiently. Additionally we aim at lifting the flatness restriction to handle fully general ABAFs and cover applications of ABA where non-flat models are used [4]. Together, these directions promise more scalable and interpretable reasoning tools for complex argumentative scenarios.

References

- [1] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, *Artif. Intell.* 77 (1995) 321–357.
- [2] L. Amgoud, H. Prade, Using arguments for making and explaining decisions, *Artif. Intell.* 173 (2009) 413–436. doi:10.1016/j.artint.2008.11.006.
- [3] K. Cyras, A. Rago, E. Albini, P. Baroni, F. Toni, Argumentative XAI: A survey, in: *Proc. of IJCAI*, 2021, pp. 4392–4399. doi:10.24963/IJCAI.2021/600.
- [4] F. Russo, A. Rapberger, F. Toni, Argumentative causal discovery, in: *Proc. of KR*, 2024, pp. 938–949. doi:10.24963/KR.2024/88.
- [5] K. Čyras, Q. Heinrich, F. Toni, Computational complexity of flat and generic assumption-based argumentation, with and without probabilities, *Artif. Intell.* 293 (2021) 103449. doi:<https://doi.org/10.1016/j.artint.2020.103449>.
- [6] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, *IEEE Transactions on Neural Networks* 20 (2009) 61–80. doi:10.1109/TNN.2008.2005605.
- [7] I. Kuhlmann, M. Thimm, Using graph convolutional networks for approximate reasoning with abstract argumentation frameworks: A feasibility study, in: *Proc. of SUM*, 2019, pp. 24–37. doi:10.1007/978-3-030-35514-2_3.
- [8] P. Cibir, J. Mailly, Graph convolutional networks and graph attention networks for approximating arguments acceptability, in: *Proc. of COMMA*, 2024, pp. 25–36. doi:10.3233/FAIA240307.
- [9] L. Malmqvist, T. Yuan, P. Nightingale, S. Manandhar, Determining the acceptability of abstract arguments with graph convolutional networks, in: *Proc. of COMMA*, 2020, pp. 47–56. URL: https://ceur-ws.org/Vol-2672/paper_5.pdf.
- [10] J. Lagniez, E. Lonca, J. Mailly, Coquiaas: A constraint-based quick abstract argumentation solver, in: *Proc. of ICTAI*, 2015, pp. 928–935. doi:10.1109/ICTAI.2015.134.
- [11] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: *Proc. of KDD*, 2014, p. 701–710. doi:10.1145/2623330.2623732.
- [12] L. Malmqvist, T. Yuan, P. Nightingale, Approximating problems in abstract argumentation with graph convolutional networks, *Artif. Intell.* 336 (2024) 104209. doi:10.1016/J.ARTINT.2024.104209.
- [13] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, in: *Proc. of ICLR*, 2022. URL: <https://openreview.net/forum?id=F72ximsx7C1>.

- [14] D. Craandijk, F. Bex, AGNN: A deep learning architecture for abstract argumentation semantics, in: Proc. of COMMA, 2020, pp. 457–458. doi:10.3233/FAIA200532.
- [15] A. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni, An abstract, argumentation-theoretic approach to default reasoning, *Artif. Intell.* 93 (1997) 63–101. doi:10.1016/S0004-3702(97)00015-5.
- [16] K. Cyras, X. Fan, C. Schulz, F. Toni, Assumption-Based Argumentation: Disputes, Explanations, Preferences. In *Handbook of Formal Argumentation.*, College Publications, 2018.
- [17] P. M. Dung, P. Mancarella, F. Toni, Computing ideal sceptical argumentation, *Artif. Intell.* 171 (2007) 642–674. doi:10.1016/J.ARTINT.2007.05.003.
- [18] T. Lehtonen, A. Rapberger, M. Ulbricht, J. P. Wallner, Argumentation frameworks induced by assumption-based argumentation: Relating size and complexity, in: Proc. of KR, 2023, pp. 440–450. doi:10.24963/KR.2023/43.
- [19] C. M. Bishop, Pattern recognition and machine learning, 5th Edition, Information science and statistics, Springer, 2007. URL: <https://www.worldcat.org/oclc/71008143>.
- [20] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: Going beyond euclidean data, *IEEE Signal Process. Mag.* 34 (2017) 18–42. doi:10.1109/MSP.2017.2693418.
- [21] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proc. of ICLR, 2017. URL: <https://openreview.net/forum?id=SJU4ayYgl>.
- [22] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: Proc. of ICLR, 2018. URL: <https://openreview.net/forum?id=rJXMpikCZ>.
- [23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. of NeurIPS, 2017, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
- [24] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P. S. Yu, Heterogeneous graph attention network, in: Proc. of WWW, 2019, pp. 2022–2032. doi:10.1145/3308558.3313562.
- [25] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* 15 (2014) 1929–1958. doi:10.5555/2627435.2670313.
- [26] A. Rapberger, M. Ulbricht, J. P. Wallner, Argumentation frameworks induced by assumption-based argumentation: Relating size and complexity, in: Proc. of NMR, 2022, pp. 92–103. URL: <https://ceur-ws.org/Vol-3197/paper9.pdf>.
- [27] K. Konczak, T. Linke, T. Schaub, Graphs and colorings for answer set programming, *Theory Pract. Log. Program.* 6 (2006) 61–106. doi:10.1017/S1471068405002528.
- [28] J. Fandinno, V. Lifschitz, Positive dependency graphs revisited, *Theory Pract. Log. Program.* 23 (2023) 1128–1137. doi:10.1017/S1471068422000333.
- [29] J. Snoek, H. Larochelle, R. P. Adams, Practical bayesian optimization of machine learning algorithms, in: Proc. of NeurIPS, 2012, pp. 2960–2968. URL: <https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html>.
- [30] L. Biewald, Experiment tracking with weights and biases, 2020. URL: <https://www.wandb.com/>, software available from wandb.com.
- [31] T. Lehtonen, A. Rapberger, F. Toni, M. Ulbricht, J. P. Wallner, Instantiations and computational aspects of non-flat assumption-based argumentation, in: Proc. of IJCAI, 2024, pp. 3457–3465. doi:10.24963/ijcai.2024/383.
- [32] T. Lehtonen, A. Rapberger, M. Ulbricht, J. P. Wallner, Acbar–atomic-based argumentation solver, *ICCMA 2023* 42 (2021) 16.
- [33] A. Rapberger, M. Ulbricht, F. Toni, On the correspondence of non-flat assumption-based argumentation and logic programming with negation as failure in the head, in: Proc. of NMR, 2024, pp. 112–121. URL: <https://ceur-ws.org/Vol-3835/paper12.pdf>.