SoftwareX 32 (2025) 102434

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

A graph convolutional network-based solver for approximating argument acceptability

Lars Malmqvist [b],*, Peter Nightingale [a], Tangming Yuan [a]

[a] University of York, UK
[b] The Tech Collective, Denmark

ARTICLE INFO

Keywords:
Abstract argumentation
Graph convolutional networks
Approximate reasoning
Argument acceptability
ICCMA
Graph neural networks

ABSTRACT

AFGCN is a software tool for approximate solutions to abstract argumentation using a Graph Convolutional Network (GCN). It addresses the computational complexity of determining argument acceptability across several semantics. The model incorporates deep residual connections, randomized training, and grounded-reasoning features to achieve strong approximation accuracy. The solver predicts acceptability status for credulous and skeptical tasks. Leveraging graph-based learning and an optimized runtime, AFGCN provides an efficient and scalable method for large-scale argumentation frameworks.

Code metadata

Current code version	.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00295
Permanent link to Reproducible Capsule	N/A
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python, PyTorch v1.10+, Deep Graph Library v0.9+, Numpy v1.20+, Scikit-learn v1.0+
Compilation requirements, operating environments & dependencies	Python 3.8+, PyTorch, Deep Graph Library, Numpy, scikit-learn. Tested on Linux and Windows.
If available Link to developer documentation/manual	https://github.com/lmlearning/AFGCN/blob/main/README.md
Support email for questions	lama@thetechcollective.eu

1. Motivation and significance

Abstract argumentation, as formalized by Dung [1], provides a foundational framework for modeling defeasible reasoning and conflict resolution in artificial intelligence. Argumentation frameworks, represented as directed graphs of arguments and attacks, allow for the formal analysis of argumentative structures across various domains, from legal reasoning [2] to multi-agent systems [3] and misinformation detection [4]. For general guidance on clarity in technical presentation when describing deep-learning frameworks, see [5].

Determining the acceptability of arguments under different argumentation semantics is a central task in abstract argumentation. Semantics such as complete, preferred, stable, grounded, ideal, semi-stable, and stage semantics define diverse criteria for argument acceptance, each capturing different intuitions about rational argument evaluation [6]. However, most standard reasoning tasks in abstract argumentation, including credulous and skeptical acceptance, are computationally hard (NP-hard or beyond) [7], posing a significant challenge for real-world applications, especially those involving large argumentation frameworks.

The computational intractability of exact argumentation reasoning has motivated research into approximate solution methods. The AFGCN solver [8–11] addresses this need by providing an efficient, scalable, and accurate approach to approximate argument acceptability using Graph Convolutional Networks (GCNs). AFGCN leverages the inherent graph structure of argumentation frameworks to train a deep learning model that can rapidly predict argument acceptability status, offering

* Corresponding author.
E-mail address: lama@thetechcollective.eu (L. Malmqvist).

https://doi.org/10.1016/j.softx.2025.102434
Received 3 May 2025; Received in revised form 9 October 2025; Accepted 29 October 2025
Available online 24 November 2025

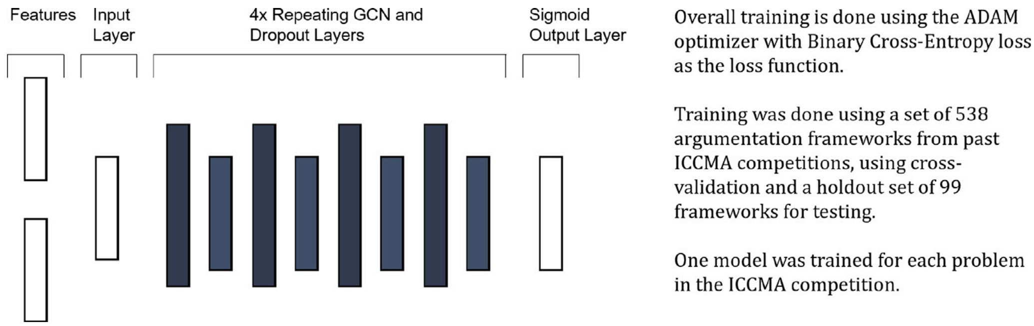

Fig. 1. Schematic representation of the AFGCN model architecture. Each block consists of a Graph Convolutional Network (GCN) layer and a Dropout layer, with residual connections incorporating input features at each block. The final output layer is a Sigmoid layer that produces acceptability probabilities.

a compelling alternative to computationally intensive exact solvers, particularly in scenarios demanding real-time performance or handling large datasets.

Relation to prior methods. Exact solvers (often SAT/ASP encodings) provide soundness and completeness but can be slow on large AFs; local-search and heuristic methods trade optimality for speed. AFGCN performs polynomial-time inference for per-argument acceptability with competitive accuracy when many arguments must be evaluated. The evaluation summary in Section 5 and Appendix reference published accuracy and runtime trade-offs. For a general note on contextualizing computational methods within existing paradigms, see [12].

AFGCN is primarily used by researchers in argumentation theory and AI reasoning systems. The typical workflow involves:

1. Converting an argumentation framework into Trivial Graph Format (TGF)
2. Running the AFGCN solver with a command specifying the semantics (e.g., preferred, stable, complete), reasoning mode (skeptical or credulous), and target argument
3. Receiving a binary output (YES/NO) indicating the estimated acceptability status

AFGCN builds upon previous work in Graph Convolutional Networks, particularly the foundational work by Kipf and Welling [13], and extends research on approximate argumentation reasoning such as the local search approaches proposed by Thimm and Cerutti [14]. The software has been benchmarked against exact solvers in the International Competition on Computational Models of Argumentation (ICCMA) and has shown to provide high-quality approximations with significant performance gains.

2. Software description

AFGCN is implemented as a Python library, designed to be flexible and extensible. The core solver logic is contained within the main solver, while the model training and architecture are trained on the basis of a configurable training script and model configuration. The library leverages the Deep Graph Library (DGL) for efficient graph operations and PyTorch for neural network computations.

2.1. Software architecture

AFGCN employs a deep residual Graph Convolutional Network (GCN) architecture, depicted in Fig. 1, that significantly extends the basic GCN model for improved performance in argumentation reasoning tasks.

The architecture consists of:

1. **Input Features Layer**: The input to the AFGCN model captures both structural and inherent properties of argumentation frameworks:
   - **Normalized Adjacency Matrix**: The adjacency matrix representing attack relations is normalized using the symmetrically normalized form
   $$\hat{\mathbf{A}} := \mathbf{D}^{-1/2}(\mathbf{A}+\mathbf{I})\,\mathbf{D}^{-1/2}. \tag{1}$$
   - **Random Initial Features**: Each node is initialized with a 64-dimensional random feature vector using Xavier initialization to enhance model generalization. During training, we use randomized batch construction and randomized label masks; ablation observations and seed handling are summarized in the evaluation section and Appendix [8, 11]. For general points on rigor and transparency, see [15].
   - **Grounded Extension Features**: A binary feature indicating membership in the grounded extension, pre-computed using the efficient grounded solver included in the library. This is used solely as an *input feature*; no training constraint or post-processing is applied.
   - **Graph Property Features**: Node-level graph properties including graph coloring, PageRank, centrality measures (degree, closeness, and eigenvector), and degree features (in-degree and out-degree).
2. **Deep Residual Blocks**: AFGCN utilizes four repeating blocks to construct a deep network. Each block contains:
   - **Graph Convolutional Layer**: A graph convolution layer with 128 hidden features that performs message passing and aggregation.
   - **ReLU Activation**: A non-linear activation function after each GCN layer.
   - **Dropout Regularization**: A dropout layer with a probability of 0.5 to prevent overfitting.
   - **Deep Residual Connection**: The original input features and normalized adjacency matrix are added to the output of each GCN block, helping to mitigate the vanishing gradient problem and allowing for more effective training of deeper architectures.
3. **Output Layer**: A fully connected linear layer that maps the 128 hidden features to a single output dimension per node, followed by a Sigmoid activation function to produce a probability value between 0 and 1 representing the model's confidence in the argument's acceptability.

AFGCN's training system architecture includes a data processing pipeline for TGF files, a randomized batch training module (see Fig. 2), and a model checkpoint system that allows for resuming training and storing the best-performing models for each semantics.

2.2. Software functionalities

AFGCN provides the following key functionalities:

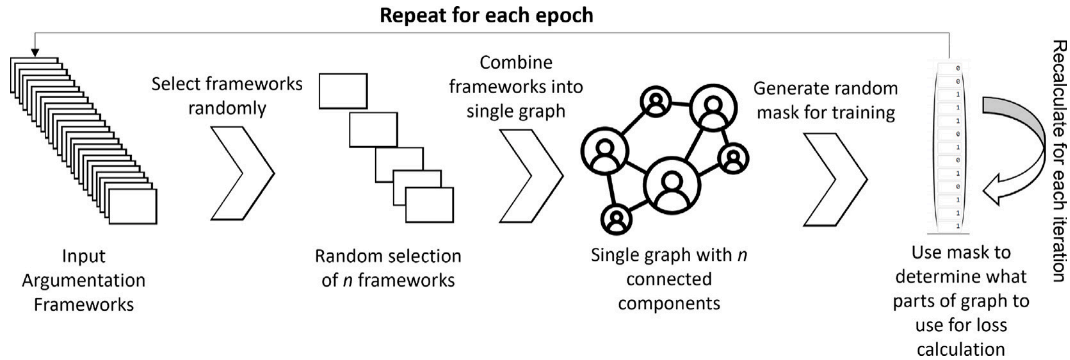

Fig. 2. Schematic representation of the AFGCN batch generation pipeline.

1. **Argumentation Framework Parsing**: Reads and parses TGF files that define arguments and attack relations.
2. **Grounded Extension Computation**: Efficiently computes the grounded extension using an optimized NumPy-based algorithm, serving both as a standalone solver for DC-GR, and DS-GR tasks and as an input feature for the GCN.
3. **Feature Generation**: Generates rich graph-based features for arguments, including:
   - Graph coloring features using NetworkX's greedy coloring algorithm
   - PageRank scores for estimating argument importance
   - Centrality measures (degree, closeness, eigenvector) for capturing graph structure
   - Degree features (in-degree, out-degree) for basic connectivity information

   Rationale. We prioritize features with favorable cost/benefit at scale. Degree, PageRank, eigenvector, and closeness centrality are inexpensive and complementary; betweenness centrality was excluded due to high computational cost (often $O(VE)$ or worse) with limited empirical benefit in pilots. Grounded membership, though non-trivial to compute, is included because it consistently improves approximation for several semantics, while its cost dominates runtime only for very large graphs [11].
4. **Model Training**: Supports training the GCN model with:
   - Randomized batch training to prevent overfitting
   - Dynamic balancing of training examples to address class imbalance
   - Outlier detection to exclude problematic frameworks
   - Checkpointing and best model saving based on validation performance
5. **Argument Acceptability Prediction**: Predicts the acceptability of arguments under various semantics:
   - Complete semantics (DC-CO, DS-CO)
   - Preferred semantics (DC-PR, DS-PR)
   - Stable semantics (DC-ST, DS-ST)
   - Semi-Stable semantics (DC-SST, DS-SST)
   - Stage semantics (DC-STG, DS-STG)
   - Ideal semantics (DS-ID)
   - Grounded semantics (DS-GR) - exact solver, not approximated

Probability interpretation and thresholds. Outputs are per-node probabilities via sigmoid. We default to a 0.5 threshold and allow optional threshold tuning or simple calibration (e.g., temperature scaling) on a validation set; a CLI flag enables both.

2.3. Reproducibility and release

We fix pseudo-random seeds via a single configuration parameter and log Python/PyTorch/DGL versions and hardware. Training uses binary cross-entropy with Adam and a simple schedule (e.g., $10^{-3} \to 10^{-6}$) as in our prior work. We provide fixed train/validation/test splits and a minimal `run.sh` example. A versioned release (`v1.0.0`), `requirements.txt`, and pretrained weights accompany the repository.

2.4. Sample code

showstringspaces

```
def solve(adj_matrix):
# Constants for labeling
IN = 1
OUT = 2
UNDEC = 0

# Initialize all arguments as UNDECIDED
labelling = np.zeros((adj_matrix.shape[0]), np.int8)

# Find initially unattacked arguments
a = np.sum(adj_matrix, axis=0) == 0
unattacked_args = np.nonzero(a)[0]

# Mark unattacked arguments as IN
labelling[unattacked_args] = IN

cascade = True
while cascade:
# Find arguments attacked by IN arguments
new_attacks = np.unique(np.nonzero(adj_matrix[unattacked_args,:])[1])
new_attacks_l = np.array([i for i in new_attacks if labelling[i] != OUT])

if len(new_attacks_l) > 0:
# Mark these arguments as OUT
labelling[new_attacks_l] = OUT

# Find arguments that might become IN
affected_idx = np.unique(np.nonzero(adj_matrix[new_attacks_l,:])[1])
else:
```

```
        affected_idx = np.zeros((0), dtype='int64')

        # Find arguments where all attackers are OUT
        all_outs = []
        for idx in affected_idx:
        incoming_attacks = np.nonzero(adj_matrix[:,idx])[0]
        if(np.sum(labelling[incoming_attacks] == OUT) == len(incoming_attacks)):
        all_outs.append(idx)

        if len(all_outs) > 0:
        # Mark these arguments as IN
        labelling[np.array(all_outs)] = IN
        unattacked_args = np.array(all_outs)
        else:
        # No more changes, end the cascade
        cascade = False

        # Return indices of all IN arguments (the grounded extension)
        in_nodes = np.nonzero(labelling == IN)[0]
        return in_nodes
```

Listing 1 Grounded Solver Implementation

3. Primer on argumentation semantics

The following Table 1 describes the semantics handled by AFGCN:

4. Illustrative examples

Suppose we want to assess the credulous acceptability of argument "A" under preferred semantics (DC-PR) for an example framework. The TGF representation would be:

showstringspaces

```
A
B
C
D
E
#
A B
B C
C A
D E
E D
```

Listing 2 TGF representation of an example framework

To execute AFGCN for this task, the following command would be used:

showstringspaces

```
./solver.sh -p DC-PR -f cycles.tgf -a A
```

Listing 3 Command to run AFGCN

Upon execution, AFGCN will:

1. Parse the input file and construct the DGL graph representation.
2. Compute the grounded extension, which in this case is empty, {}.
3. Generate input features for each argument (A, B, C, D, E).
4. Load the pre-trained AFGCN model for DC-PR.
5. Perform a forward pass through the GCN, predicting acceptability probabilities.
6. Extract the predicted probability for argument "A".
7. Compare this probability to the DC-PR threshold.
8. Output "YES" or "NO" based on whether the probability exceeds the threshold.

In this specific example, argument "A" is indeed credulously acceptable under preferred semantics because there exists a preferred extension containing A (specifically, A, C). AFGCN, trained on diverse benchmark instances, is highly likely to correctly predict "YES" for this query, demonstrating its capability to approximate complex argumentation reasoning tasks efficiently.

For more complex frameworks with hundreds or thousands of arguments, where exact computation becomes intractable, AFGCN's advantage becomes more pronounced. In such cases, while exact solvers might timeout or exhaust available memory, AFGCN can provide a high-quality approximation in seconds regardless of framework size, due to the polynomial-time inference properties of the trained GCN model.

5. Evaluation summary

Per-semantics accuracy and MCC. Our AI (2024) article reports Accuracy and Matthews Correlation Coefficient (MCC) across all decision tasks (DC/DS for CO, PR, SST, ST, STG, and DS-ID). Aggregating by semantics shows relatively small spread across tasks, with semi-stable slightly easier and stable slightly harder, though differences are modest (Table 11 in [11]).[1] (Summary source: AI 2024, Table 11.)

Class-aware results (preferred semantics). For DC-PR and DS-PR, the SAFA 2020 paper reports overall/Yes/No accuracies and ablations: e.g., DC-PR (5-layer): overall 92.26%, Yes 73.56%, No 92.95%; DS-PR (balanced data): overall 97.15%, Yes 46.35%, No 94.39%. Randomized training substantially improves positive-class performance relative to fixed batches. (Summary source: SAFA 2020, Tables 2–5.)

Ablations. Increasing depth beyond 4 layers does not consistently help and can hurt (over-smoothing/vanishing gradients); randomized training yields clear gains; grounded input features help most for preferred/complete and less for stable semantics. (Summary source: AI 2024 §5.2; SAFA 2020 §4.3.)

[1] We prefer MCC in imbalanced settings; see [11] Appendix B for details.

Table 1
Primer on semantics; decision problems appear as DC-* and DS-*.

Semantics	Informal definition
Complete	Conflict-free; defends all its members; contains only defended arguments.
Preferred	Maximal (by inclusion) admissible set.
Stable	Conflict-free; attacks every argument not in the set.
Semi-stable	Conflict-free and admissible with maximal range $S \cup S^{+}$.
Stage	Conflict-free with maximal range $S \cup S^{+}$.
Grounded	Least fixed point of the characteristic function (unique, minimal).
Ideal	Admissible and contained in every preferred extension.

Table 2
Per-semantics Accuracy and MCC aggregated across models (equally weighted). Values reproduced from [11, Table 11].

Semantics	Accuracy (%)				MCC			
	NO-GR	W/GR	GR ONLY	HYBRID	NO-GR	W/GR	GR ONLY	HYBRID
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	0.60
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	0.50	0.56	0.52	0.50
DS-CO	97.00	97.54	100.00	99.64	0.83	0.88	1.00	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

Table 3
DC-PR class-aware accuracy from SAFA 2020 [8, Table 2].

Model	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 (Balanced Data)	81.20%	91.20%	71.00%
Modified GCN (Fixed Batches)	96.40%	7.00%	99.70%
Kuhlmann & Thimm 2019 (Unbalanced)	62.00%	10.00%	97.00%
Kuhlmann & Thimm 2019 (Balanced)	63.00%	17.00%	93.00%

Table 4
DS-PR class-aware accuracy from SAFA 2020 [8, Table 3].

Model	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 (Balanced Data)	97.15%	46.35%	94.39%
Modified GCN (Fixed Batches)	98.44%	0.33%	99.66%

Cross-benchmark behavior and distribution shift. Performance varies with graph family: drops are observed on Barabási–Albert and Traffic, gains on Logic-Based Argumentation (LBA); grounded membership fraction differs sharply by family. (Summary source: AI 2024 Figs. 12–13 and Appendix B tables; dataset description in Tables 3–4.)

Runtime and scaling. Median wall-clock times by semantics/benchmark/size are reported in AI 2024 (Tables C.62–C.64), with additional statistics in Tables 14–17. AFGCN classifies all arguments in a framework typically in 10 to 30 ms for small/medium graphs without grounded features; the grounded-feature computation dominates at large scales, as expected. (Summary source: AI 2024, Tables 14–17, C.62–C.64, Figs. C.19–C.21.)

Comparisons to exact solvers and other GNNs. Compared to PYGLAF, AFGCN achieves mean speedups up to 122.8× ("all-arguments" comparison), with far larger speedups when naïvely amortizing per-argument calls. Against AGNN (Craandijk & Bex), AFGCN attains higher MCC on competition benchmarks in our published comparison. (Summary source: AI 2024 Tables 16–17 and §5.3.)

6. Impact

AFGCN has been significantly impacting the field of computational argumentation by demonstrating the effective application of Graph Neural Networks to formal reasoning tasks. The software addresses a critical gap between theoretical argumentation models and their practical application in scenarios requiring efficient computation.

The development of AFGCN has enabled new research questions to be explored:

1. **Neural Learning for Formal Reasoning**: AFGCN establishes that neural networks can effectively approximate complex logical reasoning tasks traditionally handled by symbolic methods. This has opened new research directions in neural-symbolic integration, particularly for NP-hard reasoning problems [16]. The ICCMA competition series has recognized the importance of approximate algorithms, introducing a dedicated track in the 2021 and 2023 competitions where AFGCN has demonstrated strong performance [17,18].
2. **Graph Representation Learning for Argumentation**: The success of AFGCN has prompted research into what graph structural features are most relevant for argumentation tasks and how different neural architectures capture these features. Cibier and Mailly [19] built upon AFGCN to explore alternative graph neural network architectures such as Graph Attention Networks (GATs).
3. **Dataset Influence on Learning Performance**: Kuhlmann, Wujek, and Thimm [20] used AFGCN as a baseline to investigate how properties of training datasets affect the learning of argumentation semantics, revealing important insights into training data preparation for argumentation tasks.

AFGCN has improved existing research in several ways:

1. **Scalability for Large Frameworks**: By providing approximate solutions in polynomial time to problems that are traditionally NP-hard or beyond, AFGCN enables researchers to work with larger argumentation frameworks that were previously intractable. This allows for studying more complex argument structures in domains like legal reasoning and online debate analysis [21].
2. **Benchmark for Approximate Reasoning**: AFGCN has become a benchmark against which other approximate argumentation solvers are compared, as demonstrated by its inclusion in evaluations of subsequent approaches [19]. As noted in the ICCMA 2021 results, AFGCN won most of the subtracks in the Approximate track, confirming its strong performance against other approximate methods [18].

The software has been improving research practice for argumentation researchers by:

1. Providing a practical tool for quick exploration of argument acceptability in large frameworks
2. Enabling rapid prototyping of argumentation-based applications without waiting for exact solvers
3. Serving as a fallback method when exact solvers timeout or fail
4. Presenting an alternative approach to the SAT and ASP-based solvers that dominate the field [17,18]

Table 5
Preferred semantics, equal-weighted (Accuracy; Acc(yes); Acc(no); Precision; Recall; F1; MCC) [11, Tables A.19–A.23].

DC-PR	Acc	Acc(yes)	Acc(no)	Prec	Rec	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%	0.70	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
DS-PR	Acc	Acc(yes)	Acc(no)	Prec	Rec	F1	MCC
GCN-NO-GR	86.24%	84.53%	86.99%	87.81%	51.82%	0.53	0.50
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	0.50

AFGCN has been used by research groups working on argumentation theory and formal reasoning systems. For example, the approach has influenced the development of argumentation systems that combine exact and approximate methods, such as the approach developed by Craandijk and Bex [16,22,23].

While primarily used in academic research, AFGCN's techniques for efficient graph-based reasoning have potential commercial applications in legal reasoning systems, policy analysis tools, and automated negotiation platforms where argumentation frameworks are used to model complex decision processes. The development of AFGCN aligns with the trend in argumentation computing towards more diverse algorithmic approaches beyond the dominant SAT and ASP-based methods, as highlighted in recent ICCMA competitions [17,18].

7. Conclusions

AFGCN provides a robust, efficient, and scalable software solution for approximating argument acceptability in abstract argumentation. By leveraging a deep residual GCN architecture, a carefully designed training regime, and optimized runtime implementation, AFGCN achieves state-of-the-art performance on challenging argumentation reasoning tasks.

The software's modular design, open-source availability, and documentation facilitate its use by researchers and practitioners seeking to apply approximate argumentation reasoning in real-world applications. The efficiency gains are particularly valuable for large-scale argumentation frameworks where exact solvers become computationally infeasible.

Future development directions could include exploring further architectural enhancements, incorporating more sophisticated graph-aware features, and extending the solver's capabilities to address a broader range of argumentation reasoning problems beyond acceptability determination, such as approximation of gradual semantics or enforcement problems.

CRediT authorship contribution statement

Lars Malmqvist: Writing – original draft, Visualization, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Peter Nightingale:** Writing – review & editing, Validation, Supervision, Methodology. **Tangming Yuan:** Writing – review & editing, Validation, Supervision, Project administration, Methodology.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used Claude 3.5 in order to format references into correct format and edit some sections. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Table 6
Effect of depth and training optimization on preferred semantics (Accuracy %/MCC). Reproduced from [11, Table 13, §5.2].

Model	DC-PR	DS-PR
4-Layer AFGCN	95.1/0.610	97.5/0.720
5-Layer AFGCN	94.9/0.601	97.4/0.704
6-Layer AFGCN	93.2/0.398	97.4/0.704
4-Layer AFGCN (no training optimization)	92.2/0.327	94.9/0.291

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the ICCMA competition organizers for providing the benchmark datasets used in training and evaluating AFGCN.

Appendix. Consolidated evaluation tables from prior work

This appendix consolidates the key evaluation results used in the paper. Unless otherwise noted, all figures are reproduced from our prior publications and retain their original formatting: Malmqvist et al. *Artificial Intelligence* (2024) [11] and Malmqvist et al. SAFA@COMMA (2020) [8]. We group results by (i) per-semantics accuracy/MCC, (ii) class-aware metrics for preferred semantics, (iii) ablations on depth and training optimization, and (iv) runtime statistics and comparisons.

A.1. Evaluation settings used in [11]

We report three standard aggregation settings from [11]: *Equally weighted* (each framework weighted equally), *Complete balanced* (all arguments weighted; large frameworks contribute proportionally), and *Reduced balanced* (complete balanced excluding benchmarks solvable by grounded reasoning alone). See [11] for formal definitions.

A.2. Class-aware metrics for preferred semantics (Yes/No)

Credulous acceptance (DC-PR), SAFA 2020. See Table 3.

Sceptical acceptance (DS-PR), SAFA 2020. See Table 4.

Preferred semantics, equal-weighted class-aware (2024). For completeness we include DC-PR and DS-PR equal-weighted class-aware summaries from [11, App. A].

A.3. Ablation summaries (depth and training optimization)

See Tables 6–12.

Table 7
AFGCN runtime statistics for classifying a full framework (ms). Reproduced from [11, Table 14].

	Min	25%	50%	75%
Runtime w/GR	6.83	12.44	28.96	810.58
Runtime no GR	6.12	10.55	20.72	242.72

Table 8
PYGLAF single-argument runtime by semantics group (ms). Reproduced from [11, Table 15].

Group	Mean	Median	Min	Max
DC-CO	123 490	51 294.7	122.480	594 880
DC-PR	123 646	50 806.5	137.729	595 089
DC-SST	190 652	64 992.1	137.916	600 172
DC-ST	147 624	68 084.8	151.857	588 594
DC-STG	470 557	599 532	135.897	601 137
DS-CO	92 184.7	46 223.1	108.567	570 255
DS-ID	490 946	595 811	170.028	601 034
DS-PR	239 657	104 572	114.881	600 682
DS-SST	191 973	66 349.2	145.543	600 861
DS-ST	192 526	93 378.4	137.009	589 832
DS-STG	460 623	599 432	152.218	601 131

Table 9
AFGCN runtime (all arguments per framework) by semantics group (ms). Reproduced from [11, Table 16].

Group	Mean	Median	Min	Max
DC-CO	29 014.5	18 456.3	976.1	63 070.5
DC-PR	29 294.5	19 194.8	1012.3	62 154.1
DC-SST	27 876.7	17 168.7	1039.0	65 651.4
DC-ST	28 333.2	18 064.9	972.0	67 038.8
DC-STG	8952.2	7669.2	980.9	36 944.3
DS-CO	32 225.2	31 585.8	995.3	62 023.9
DS-ID	3988.8	3260.6	982.8	15 123.2
DS-PR	10 107.5	8449.7	965.2	33 380.9
DS-SST	9120.6	7917.3	1024.8	42 996.7
DS-ST	21 555.2	15 009.3	1015.2	62 530.0
DS-STG	8938.6	7894.1	975.2	39 858.8

Table 10
AFGCN runtime per argument by semantics group (ms). Reproduced from [11, Table 17].

Group	Mean	Median	Min	Max
DC-CO	0.511720	0.325508	0.0172153	1.11236
DC-PR	0.517241	0.338915	0.0178741	1.09743
DC-SST	0.493667	0.304039	0.0183997	1.16261
DC-ST	0.521155	0.332283	0.0178783	1.23310
DC-STG	0.992253	0.850040	0.108718	4.09485
DS-CO	0.450129	0.441197	0.0139025	0.866363
DS-ID	3.29238	2.69133	0.811174	12.4827
DS-PR	0.922482	0.771186	0.0880943	3.04659
DS-SST	1.01557	0.881593	0.114106	4.78765
DS-ST	0.689990	0.480453	0.0324956	2.00161
DS-STG	0.971628	0.858082	0.106003	4.33264

A.4. Runtime statistics and comparisons

Remark on speedups vs exact solvers. When compared against PYGLAF (ICCMA'21 preferred track winner), AFGCN achieves mean speedups up to 122.8× in the "all arguments" mode and theoretical per-argument speedups exceeding 10^5× for some groups; see detailed discussion and caveats in [11, §5.2.5; Tables 15–17]. *Provenance:* All values in Tables 2–12 are reproduced verbatim from [8,11].

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2025.102434.

Table 11
Median runtime by *benchmark* (s), with and without grounded computation. Reproduced from [11, Table C.63].

Benchmark	w/ GR	No GR
ABA2AF	1.79	1.32
AFGen	0.06	0.05
Barabasi–Albert	0.01	0.01
Erdős–Rényi	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 12
Median runtime by *semantics* (s), with and without grounded computation. Reproduced from [11, Table C.62].

Semantics	w/ GR	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

References

[1] Dung PM. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 1995;77(2):321–57.
[2] Bench-Capon TJ, Dunne PE. Argumentation in artificial intelligence. Artificial Intelligence 2007;171(10–15):619–41.
[3] Modgil S, et al. The added value of argumentation. In: Argumentation in multi-agent systems. Springer; 2013, p. 357–403.
[4] Kotonya N, Toni F. Explainable automated fact-checking for public health claims. In: Proceedings of the 2020 conference on empirical methods in natural language processing. 2020, p. 7740–54.
[5] Lafta NA. A comprehensive analysis of keras: Enhancing deep learning applications in network engineering. Babylon J Netw 2023;2023:94–100. http://dx.doi.org/10.58496/BJN/2023/012.
[6] Baroni P, et al. An introduction to argumentation semantics. Knowl Eng Rev 2011;26(4):365–410.
[7] Dvorák W, Dunne PE. Computational problems in formal argumentation and their complexity. In: Handbook of formal argumentation. College Publications; 2018, p. 631–87.
[8] Malmqvist L, Yuan T, Nightingale P, Manandhar S. Determining the acceptability of abstract arguments with graph convolutional networks. In: Proceedings of the second international workshop on systems and algorithms for formal argumentation. 2020, p. 47–56.
[9] Malmqvist L. AFGCN: An approximate abstract argumentation solver. In: International competition on computational models of argument. 2021.
[10] Malmqvist L. Approximate solutions to abstract argumentation problems using graph neural networks (Ph.D. thesis), University of York; 2022.
[11] Malmqvist L, Yuan T, Nightingale P. Approximating problems in abstract argumentation with graph convolutional networks. Artificial Intelligence 2024;336:104209. http://dx.doi.org/10.1016/j.artint.2024.104209.
[12] Hakim BA. Using of computational fluid dynamics (CFD) and machine learning (ML) for predictive modelling in fluid dynamic. KHWARIZMIA 2024;2024:85–94. http://dx.doi.org/10.70470/KHWARIZMIA/2024/012.
[13] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: International conference on learning representations. 2017.
[14] Thimm M, Cerutti F. Approximate mechanisms for solving abstract argumentation frameworks. In: TAFA@iJCAI. 2019.
[15] Turyasingura B, Willbroad B, Alhasani AT, Deyab WS. Correcting convexity proofs for two signomial functions in geometric programming. Appl Data Sci Anal 2025;2025:125–31. http://dx.doi.org/10.58496/ADSA/2025/009.

[16] Craandijk D, Bex F. Deep learning for abstract argumentation semantics. In: Proceedings of the twenty-ninth international joint conference on artificial intelligence. 2020.
[17] Bistarelli S, Kotthoff L, Lagniez J-M, Lonca E, Mailly J-G, Rossit J, Santini F, Taticchi C. The third and fourth international competitions on computational models of argumentation: design, results and analysis. Argum Comput 2024;15(1):1–73.
[18] Järvisalo M, Lehtonen T, Niskanen A. ICCMA: 5th international competition on computational models of argumentation. Artif Intell 2023.
[19] Cibier S, Mailly JG. A comparison of graph neural networks for approximating argumentation semantics. In: Proceedings of the 10th international conference on computational models of argument. 2024.
[20] Kuhlmann I, Wujek M, Thimm M. On the impact of data selection when applying machine learning in abstract argumentation. In: Computational models of argument: proceedings of COMMA 2022. 2022, p. 271–82.
[21] Malmqvist L, Yuan T, Nightingale P. Improving misinformation detection in tweets with abstract argumentation. In: CMNA. 2021, p. 40–6.
[22] Craandijk D, Bex F. Enforcement heuristics for argumentation with deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, (5):2022, p. 5573–81.
[23] Craandijk D, Bex F. Effects of graph neural network aggregation functions on generalizability for solving abstract argumentation semantics. 2024.