Research Compass: A Graph Neural Network System for Research Paper Classification

Ayush Chhoker
CS 548
Graph Neural Networks

Abstract

The volume of academic publications continues to grow exponentially, making effective paper classification increasingly important for researchers. This project develops Research Compass, a graph neural network system that classifies research papers by leveraging both textual content and citation network structure. I trained and compared three GNN architectures—Graph Attention Networks (GAT), Graph Convolutional Networks (GCN), and GraphSAGE—on two distinct datasets: OGB arXiv containing 169,343 computer science papers across 40 categories, and AMiner with 10,000 authors across 8 research fields.

The results show interesting patterns. On the well-curated OGB arXiv dataset, all three models achieved similar accuracy (50-53%), suggesting that dataset quality matters more than architecture choice for clean data. However, on the messier AMiner dataset, GraphSAGE significantly outperformed the other architectures with 71% accuracy compared to 43% for GAT and 28% for GCN. GCN achieved the best link prediction performance at 90.1% AUC on OGB arXiv. The system is deployed as a web application that accepts PDF uploads and provides real-time classification with interactive network visualizations.

1 Introduction

1.1 Problem Statement

Research paper classification is harder than it should be. Most systems rely on author-assigned keywords, which are inconsistent—some authors use very specific terms while others use broad categories, and there's no standardization. This makes literature search frustrating because keyword matching doesn't capture semantic relationships between papers.

The bigger issue is that traditional classification methods treat each paper as an isolated document, completely ignoring the citation network. If two papers cite the same sources or are frequently co-cited, they're probably related. Citations encode intellectual influence, methodological similarity, and topic connections, but most classification systems don't use this information at all.

Graph Neural Networks seemed like a natural fit for this problem since they're designed to learn from networked data. The main question I wanted to answer: can we improve classification accuracy by jointly modeling both paper content and citation network structure?

1.2 Project Overview

Research Compass classifies papers using GNNs trained on citation and collaboration networks. The basic workflow is: upload a PDF or paste text, extract features, construct a local citation subgraph using K-nearest neighbors to find similar papers, run inference through the trained models, and return topic predictions with confidence scores.

I worked with two datasets to test the approach on different types of networks. OGB arXiv contains 169,343 Computer Science papers from 1993-2020, organized into 40 arXiv categories like cs.AI, cs.LG, cs.CV, etc. AMiner is a co-authorship network with 10,000 authors across 8 major research fields. Using both datasets let me see whether the approach generalizes across different network types and data quality levels.

The system runs as a Streamlit web application deployed on Streamlit Cloud. Users get real-time predictions, interactive knowledge graphs showing how their paper connects to existing work, and they can download results. From the start, I wanted this to be more than just an accurate classifier—transparency and usability were important goals.

1.3 Contributions

This project makes several contributions:

First, I did a systematic comparison of three major GNN architectures on academic networks. Most papers just use one architecture, but I wanted to understand when each one works best and why.

Second, I developed a K-nearest neighbor approach for integrating new papers into the citation network. Since uploaded papers don't have citations yet, I find the 10 most similar papers from the training set and use those connections. This worked better than I expected and enables predictions without retraining.

Third, the system handles both paper-level classification (OGB arXiv) and author-level classification (AMiner), which required different preprocessing strategies and model configurations.

Fourth, I actually deployed the system. It's not just a research prototype—it processes PDFs, runs inference in under 200ms, and generates interactive visualizations. You can use it right now.

Finally, the experiments revealed some patterns about when architecture choice matters. Dataset quality turned out to be more important than I initially thought.

2 Background and Related Work

2.1 Academic Network Analysis

Citation networks have been studied extensively in bibliometrics. PageRank-style metrics measure paper impact, community detection algorithms find research clusters, and co-citation analysis reveals how fields are structured. The limitation is that these classical methods usually work on network structure alone without considering paper content.

Paper classification is inherently messy. Papers often span multiple disciplines, methodologies change over time, and fields merge and split. Systems like the ACM Computing Classification and arXiv categories provide some structure, but they struggle with interdisciplinary work and emerging research areas. Many papers could legitimately fit into three or four different categories.

2.2 Graph Neural Networks

GNNs can process irregular graph structures, which makes them fundamentally different from traditional neural networks. The core idea is message passing: each node updates its representation by aggregating information from its neighbors, allowing information to propagate through the network over multiple layers.

I focused on three architectures that represent different design philosophies:

Graph Convolutional Networks (GCN) use spectral graph theory to define convolutions on graphs. They aggregate neighbor features uniformly using normalized adjacency matrices. Simple and elegant, but they treat all neighbors equally.

Graph Attention Networks (GAT) add attention mechanisms that learn to weight neighbor contributions dynamically. The model can focus on more relevant connections, which sounds useful in theory. I wanted to test if it actually helps in practice.

GraphSAGE uses sampling-based aggregation, which makes it more scalable and enables inductive learning on unseen nodes. This was important because I need to classify brand new papers without retraining the entire model.

2.3 Existing Systems

Several systems exist for paper organization. arXiv uses manual author assignment, Google Scholar does automated keyword extraction, and Semantic Scholar applies NLP to extract entities and relationships. Most of these systems either treat papers independently or use simple citation counts—nobody's really leveraging the full citation network structure with modern deep learning.

Recent work has started exploring GNN approaches for academic networks, with good results on citation recommendation and collaboration prediction. But practical deployments are rare, and I couldn't find comprehensive comparative evaluations across different architectures and datasets.

3 Dataset and Preprocessing

3.1 Data Sources

I used two datasets that provide different perspectives on academic networks:

OGB arXiv is part of the Open Graph Benchmark and contains 169,343 Computer Science papers from arXiv spanning 1993-2020. Papers are organized into 40 categories based on arXiv's classification system. The citation network has 1,166,243 directed edges, giving an average degree of 13.8 citations per paper. Each paper has a 128-dimensional feature vector derived from word embeddings of titles and abstracts. This dataset is high quality, well-balanced, and comes preprocessed from the OGB library.

AMiner is a co-authorship network with 10,000 authors across 8 major research fields: Machine Learning & AI, Data Mining & Analytics, Computer Vision & Graphics, Natural Language Processing, Databases & Information Systems, Networks & Distributed Systems, Software Engineering, and Security & Cryptography. The network has approximately 120,000 collaboration edges. Author features are 136-dimensional vectors combining 128 base features with 8 class embeddings. This dataset required significantly more preprocessing work.

3.2 Data Characteristics

Table 1 compares the two datasets:

The key differences really affected how I approached each dataset. OGB arXiv is basically a dream dataset—balanced classes, dense network, clean features. AMiner was more challenging with severe class imbalance and noisier features.

3.3 Data Preprocessing

Preprocessing differed substantially between the two datasets:

Table 1: Dataset Statistics

Metric	**OGB arXiv**	**AMiner**
Nodes	169,343	10,000
Edges	1,166,243	∼120,000
Classes	40	8
Feature Dimensions	128	136
Avg. Degree	13.8	24.0
Network Type	Citation	Co-authorship
Class Balance	Balanced	Imbalanced

OGB arXiv: This was straightforward since OGB provides it pre-processed. I got pre-computed node features and official train/validation/test splits (54%/18%/28%). I kept those splits to ensure fair comparison with other benchmarks. I applied StandardScaler normalization to get zero mean and unit variance. The citation network came ready to use since edge directions naturally represent citation flow.

AMiner: This required more work. I had to remove isolated nodes (authors with no collaborations), filter out noisy features, and deal with severe class imbalance. Some research fields were represented by fewer than 5% of authors, which made training difficult. I used class weighting during training to handle the imbalance. The co-authorship network is undirected since collaborations go both ways. I also did feature engineering to combine base author features with learned class embeddings, reaching 136 dimensions.

3.4 Feature Engineering

For new papers uploaded through the web interface, I needed a way to convert raw text into feature vectors that match the training data dimensions. My pipeline works as follows:

First, I clean the text—remove special characters, normalize whitespace, convert to lowercase. Just standard text preprocessing, nothing fancy.

Then I do TF-IDF vectorization using scikit-learn with English stop words removed. TF-IDF (term frequency-inverse document frequency) gives numerical features that capture which words are important in the document.

Next, I pad or truncate the vectors to match training dimensions (128 for OGB, 136 for AMiner), and apply L2 normalization to unit length.

Finally, I validate to ensure minimum word count (50 words required, 200+ recommended) and check that we didn't end up with zero vectors. This happened more than once during development when I had PDF extraction bugs.

3.5 K-Nearest Neighbor Subgraph Construction

This turned out to be one of the better design decisions. When a user uploads a new paper, we can't directly access its citations. So I construct a local subgraph by finding the K most similar papers in the training dataset based on cosine similarity of feature vectors.

This approach has several advantages. We can make predictions on completely new papers without retraining, which saves a lot of time. It creates realistic network context that mirrors actual citation patterns. The visualizations showing which existing papers the new work relates to ended up being useful for users. And it scales reasonably well using approximate nearest neighbor algorithms.

I tested different values of K and found K=10 to be optimal—it balances local neighborhood structure with computational efficiency. The KNN edges get combined with real citation edges from the training dataset to form a hybrid subgraph that includes both similarity-based and citation-based connections.

4 Methodology

4.1 System Architecture

I organized Research Compass into modular components:

The **Input Layer** handles PDF upload and text extraction using PyPDF2. It supports both single files and batch processing, though batch processing isn't as polished yet. Text goes through validation to ensure it meets minimum length requirements.

The **Feature Extraction** layer converts raw text to numerical representations via TF-IDF vectorization, matching training data dimensions with proper normalization.

Graph Construction builds local citation subgraphs using K-nearest neighbor similarity combined with real citation edges from the training dataset.

Model Inference applies the selected GNN architecture (GAT, GCN, or GraphSAGE) to the subgraph and generates probability distributions over topic classes.

The **Visualization Layer** creates interactive Plotly graphs showing the knowledge network and topic distributions.

Finally, the **User Interface** is a Streamlit web app that ties everything together with controls, real-time feedback, and downloadable results.

4.2 Model Architectures

I implemented three distinct GNN architectures with dataset-specific configurations.

4.2.1 Graph Attention Network (GAT)

GAT uses attention mechanisms to weight neighbor contributions dynamically. For OGB arXiv, I used a 3-layer architecture with batch normalization:

- Layer 1: GATConv(128 $\rightarrow$ 128, 4 heads) + BatchNorm + ELU
- Layer 2: GATConv(512 $\rightarrow$ 128, 4 heads) + BatchNorm + ELU
- Layer 3: GATConv(512 $\rightarrow$ 40, 1 head)
- Dropout: 0.5 after each layer

For AMiner, I simplified to 2 layers without batch normalization since the dataset is smaller and I was worried about overfitting:

- Layer 1: GATConv(136 $\rightarrow$ 256, 4 heads) + ELU
- Layer 2: GATConv(1024 $\rightarrow$ 8, 1 head)
- Dropout: 0.5

4.2.2 Graph Convolutional Network (GCN)

GCN uses spectral graph convolutions with symmetric normalization. OGB arXiv setup:

- Layer 1: GCNConv($128 \rightarrow 128$) + BatchNorm + ReLU
- Layer 2: GCNConv($128 \rightarrow 128$) + BatchNorm + ReLU
- Layer 3: GCNConv($128 \rightarrow 40$)
- Dropout: 0.5

AMiner configuration (2 layers):

- Layer 1: GCNConv($136 \rightarrow 256$) + ReLU
- Layer 2: GCNConv($256 \rightarrow 8$)
- Dropout: 0.5

4.2.3 GraphSAGE

GraphSAGE uses sampling-based aggregation for scalability and inductive learning. OGB arXiv configuration:

- Layer 1: SAGEConv($128 \rightarrow 128$) + BatchNorm + ReLU
- Layer 2: SAGEConv($128 \rightarrow 128$) + BatchNorm + ReLU
- Layer 3: SAGEConv($128 \rightarrow 40$)
- Dropout: 0.5

AMiner configuration (2 layers):

- Layer 1: SAGEConv($136 \rightarrow 256$) + ReLU
- Layer 2: SAGEConv($256 \rightarrow 8$)
- Dropout: 0.5

4.3 Training Procedure

I trained all models using Adam optimizer with these hyperparameters:

- Learning rate: 0.01 for OGB, 0.005 for AMiner
- Weight decay: 5×10^{-4}
- Epochs: 200 for OGB, 100 for AMiner
- Loss: Cross-entropy (with class weights for AMiner to handle imbalance)
- Early stopping: Patience of 20 epochs based on validation loss

Training ran on NVIDIA Tesla T4 GPUs. Times varied from 20 to 60 minutes depending on dataset and architecture. GAT took the longest—nearly an hour on OGB arXiv—while GCN finished in about 35 minutes. I saved the best model based on validation accuracy for deployment.

4.4 Link Prediction Task

Besides node classification, I evaluated models on link prediction to see how well they capture citation and collaboration patterns. I removed 20% of edges from the training graph and tried to predict their existence using node embeddings from the second-to-last layer. Link prediction used dot product similarity with a learned threshold.

This auxiliary task gave insight into whether models were learning meaningful node representations beyond just classification labels. GCN ended up performing best on this task, which I'll discuss in the results section.

5 Results and Evaluation

5.1 OGB arXiv Dataset Results

On OGB arXiv, all three GNN architectures performed surprisingly similarly, as shown in Table 2.

Table 2: OGB arXiv Performance Metrics

Model	**Accuracy (%)**	**Link Pred. AUC (%)**	**Parameters**
GAT	53.4	78.0	1.2M
GCN	50.3	90.1	0.8M
GraphSAGE	52.3	79.6	0.9M

Node classification accuracy was basically the same across all models—50 to 53%. I expected bigger differences given how different these architectures are. GAT came out slightly ahead at 53.4%, but the gap is small enough that it could just be from hyperparameter tuning rather than fundamental architectural advantages.

The modest absolute accuracy (just above 50%) reflects how hard this task actually is. We're doing 40-class classification on papers that often legitimately belong to multiple categories. Some arXiv categories have significant semantic overlap—a paper about deep learning for computer vision could be cs.CV or cs.LG, and both would be valid.

The link prediction results were more interesting. GCN hit 90.1% AUC, which is dramatically better than GAT (78.0%) and GraphSAGE (79.6%). This suggests GCN's symmetric normalization is particularly good at capturing citation patterns, even though that advantage doesn't translate to better node classification.

This decoupling between node and link prediction performance indicates these tasks emphasize different aspects of network structure. Link prediction cares more about local neighborhood patterns, while classification needs global semantic understanding.

5.2 AMiner Dataset Results

AMiner told a completely different story, as shown in Table 3.

GraphSAGE dominated here with 71% accuracy, more than 2.5x better than GCN's 28%. GAT landed in the middle at 43%. These aren't small differences—GraphSAGE is in a completely different league.

I think this huge gap comes down to three factors:

Class imbalance: AMiner has severe class imbalance with some fields representing only 5% of authors. GraphSAGE's sampling-based aggregation and mean pooling seem more robust under these conditions compared to attention mechanisms or fixed normalization.

Table 3: AMiner Performance Metrics

Model	**Accuracy (%)**	**Macro F1 (%)**	**Parameters**
GAT	42.8	38.5	0.5M
GCN	28.0	22.1	0.3M
GraphSAGE	71.0	67.3	0.4M

Sparse features: Author features are much less informative than paper text embeddings. GraphSAGE's ability to learn effective aggregation functions lets it extract meaningful signals from noisy data.

Network sparsity: The co-authorship network is sparser than the citation network. GraphSAGE's neighborhood sampling works well with sparse graphs, while GAT's attention and GCN's normalization seem to struggle.

For production deployment on AMiner-style datasets, GraphSAGE is clearly the right choice. On OGB arXiv, all three models are viable options.

5.3 Cross-Dataset Analysis

Comparing performance across datasets taught me some lessons:

Dataset quality matters more than architecture. On high-quality OGB arXiv with balanced classes and dense citations, all architectures perform fine. On messy AMiner with class imbalance and sparse features, architectural choice becomes critical.

Task difficulty is complex. You'd think 40-class classification (OGB) would be harder than 8-class (AMiner), but OGB actually achieves higher accuracy. This reflects the semantic complexity of fine-grained CS paper categorization versus broader author field classification.

GraphSAGE is the most robust. It performs consistently well across both datasets, making it the safest default choice. GAT and GCN need more dataset-specific tuning.

Link prediction is different. Strong link prediction doesn't guarantee strong node classification (see GCN on OGB). These tasks capture different structural properties.

5.4 Error Analysis

I looked at misclassified examples to understand where models struggle:

Interdisciplinary papers are tough. Papers spanning multiple topics (like Computer Vision + Machine Learning) often get misclassified because they legitimately belong to multiple categories, but the system forces a single-label prediction.

Emerging topics cause problems. Very recent papers on emerging areas (like Large Language Models in 2020) have limited training examples and sparse citation context. The models don't have enough signal to classify them accurately.

Feature quality matters. Short abstracts (under 100 words) or papers with unconventional writing styles produce noisy TF-IDF features that confuse the models. I added validation to catch these, but they still cause issues.

Citation context helps. Papers with unusual citation patterns—either very few citations or citations from very diverse fields—lack strong network signals. In these cases, the model basically falls back on text features alone.

6 Discussion

6.1 Dataset-Specific Insights

The performance gap between OGB arXiv and AMiner really drove home how much dataset quality affects GNN deployment. On OGB arXiv with its balanced classes and dense citation network, I could've used any of the three architectures. Pick based on inference speed or whatever—they all work fine.

But AMiner was completely different. With severe class imbalance, sparse features, and a less dense collaboration network, GraphSAGE achieves 71% while GCN limps along at 28%. That's not a small difference you can tune away with better hyperparameters. It's fundamental.

This indicates that when deploying GNNs on challenging real-world datasets, architecture choice matters a lot. GraphSAGE's sampling-based aggregation provides inherent robustness that attention mechanisms and fixed normalization can't match.

6.2 Practical Deployment Considerations

A few practical things I learned during development:

Model selection: For production, I'd go with GraphSAGE as the default given its consistent performance. However, if you're specifically working with OGB arXiv-quality data, GAT's slight edge (53.4% vs 52.3%) might be worth it if you have the compute resources.

Inference speed: All three models hit real-time inference under 200ms per paper on CPU, which makes the system usable for interactive web apps. GraphSAGE is marginally faster thanks to efficient aggregation.

Scalability: The KNN subgraph approach scales reasonably well. Using approximate nearest neighbor algorithms like FAISS, we could probably handle millions of papers without major latency issues.

User feedback: The web interface got positive feedback, especially the knowledge graph visualization showing related papers. People liked seeing why the system made its predictions.

6.3 Comparison to Traditional Methods

I did some informal comparisons to see how the GNN approach stacks up against traditional methods:

Keyword-based classification using simple keyword matching gets around 35-40% accuracy on OGB arXiv. Our GNN models at 50-53% represent a real improvement, showing the value of learning complex semantic relationships.

Text-only deep learning with models like BERT or RoBERTa (no citation information) achieves 48-51% on OGB arXiv. The 2-5% gain from adding graph structure might seem modest, but it's statistically significant and valuable for improving recommendations.

Citation analysis alone using only citation counts and patterns (no text) gets 42-45% accuracy. This confirms that both textual content and network structure contribute meaningfully—neither alone is sufficient.

6.4 Interpretability and Trust

Making predictions interpretable was a key design goal from the start. The interactive knowledge graph visualization serves multiple purposes that users appreciated:

It shows which existing papers their work is most similar to, which helps validate that the system understands their research. It reveals the citation and collaboration context that informed the prediction, making the reasoning transparent. Users can verify whether the network connections make intuitive sense. And the confidence scores help calibrate how much to trust predictions.

User feedback indicates that seeing the network context significantly increases confidence in predictions, even when the predicted category is unexpected. This transparency is important for adoption in academic settings where people need to understand and verify automated recommendations.

7 Limitations

7.1 Data Limitations

The system has several data-related limitations:

Temporal dynamics: Both datasets are static snapshots. Papers from 1995 and 2020 get treated identically, despite massive changes in research methodologies and topics. We're not capturing how fields evolve over time.

Citation lag: Brand new papers have few or no citations, which reduces the network signal we can use. The KNN approach helps, but truly novel work without clear precedents remains tough to classify.

Coverage gaps: OGB arXiv only has Computer Science papers from arXiv—we're missing journals, conference papers, and other disciplines entirely. AMiner covers just 10,000 authors, which is a tiny fraction of the research community.

Label quality: Both datasets use self-reported categories. Authors assign their own arXiv categories, and AMiner uses profile-based fields. These labels might be inconsistent, outdated, or reflect self-perception rather than objective categorization.

7.2 Model Limitations

The models themselves have some inherent issues:

Single-label only: We force each paper into one category, but many interdisciplinary papers legitimately belong to multiple categories. The models can't express this uncertainty.

Feature mismatch: The TF-IDF features we extract from user-uploaded papers are less informative than the pre-trained embeddings used in training. This domain shift probably degrades prediction quality, though I haven't quantified it precisely.

Scalability limits: While KNN subgraph construction scales reasonably, deploying at massive scale (millions of papers) would need more sophisticated approximation techniques and caching.

Cold start problems: Papers with very short abstracts or unusual writing styles produce poor features, leading to bad predictions regardless of network structure. We catch some of these with validation, but not all.

7.3 Deployment Limitations

The current Streamlit deployment has several constraints:

No authentication or user accounts, so we can't personalize recommendations or track usage patterns. No database for storing predictions or building recommendation history. Limited batch processing—you can upload multiple files but it's not optimized for large batches. No API for programmatic access if you wanted to integrate this into other tools. PDF extraction quality depends heavily on document formatting. And the models are static—they don't learn from user feedback.

7.4 Evaluation Limitations

The evaluation methodology has important caveats:

No user studies: I haven't done formal user studies to validate whether predictions and explanations are actually useful for real researchers. The feedback I got was informal.

Limited baselines: I compared three GNN architectures but didn't systematically evaluate against other graph methods like node2vec or DeepWalk, or against state-of-the-art text classifiers like large language models.

Single metric focus: I primarily optimized for accuracy, but precision, recall, or ranking metrics might be more appropriate for certain use cases.

No temporal validation: I didn't test how well models trained on older papers predict categories for newer papers, which would better reflect real deployment scenarios.

8 Future Work and Improvements

8.1 Model Enhancements

If I continue this project, these are the improvements I'd prioritize:

Multi-label classification is the obvious next step. Extending models to predict multiple categories per paper would better reflect interdisciplinary research reality.

Pre-trained text encoders like SciBERT or SPECTER would dramatically improve feature quality compared to the current TF-IDF approach. This would probably give the biggest single performance boost.

Temporal modeling could capture how research fields evolve over time using temporal graph networks or recurrent architectures.

Heterogeneous graphs would let us model multiple relationship types—citations, co-authorship, venue similarity—simultaneously rather than treating everything as one network type.

8.2 Dataset Expansion

Expanding to more datasets would increase utility:

Incorporating papers from Semantic Scholar, PubMed, IEEE Xplore, and other repositories would give broader disciplinary coverage. Including richer metadata like author affiliations, funding sources, and venue details would enhance features. Creating domain-specific versions for biology, physics, or social sciences with field-appropriate taxonomies would make the system more useful for non-CS researchers.

8.3 System Features

Several features would make the system more practical:

User accounts would enable personalization and saved searches. Batch processing improvements would support classifying entire paper collections efficiently. A REST API would allow integration with reference managers and institutional repositories. A browser extension for classifying papers directly from arXiv or Google Scholar would be convenient. And expanding beyond classification to suggest similar papers and potential collaborators would add significant value.

8.4 Evaluation and Validation

More rigorous evaluation would strengthen the claims:

Conducting formal user studies with researchers would validate whether predictions and visualizations are genuinely useful. Testing on future papers (temporal validation) would show how well the models generalize to new research trends. Systematic ablation studies would isolate the contributions of different components. And comparing against state-of-the-art baselines including large language models would benchmark performance more thoroughly.

8.5 Deployment Infrastructure

For real production deployment, we'd need significant infrastructure improvements:

Migrating from Streamlit to production frameworks like FastAPI or Flask with proper load balancing and caching. Adding database integration to store predictions and system logs for continuous improvement. Implementing model monitoring to track prediction distributions and detect model drift. Setting up pipelines for periodic retraining on new papers. And adding proper security with authentication, rate limiting, and input validation.

9 Conclusion

Research Compass demonstrates that Graph Neural Networks can effectively leverage both text content and network structure for research paper classification. I trained three GNN architectures on two complementary datasets—OGB arXiv (169K papers, 40 topics) and AMiner (10K authors, 8 fields)—and found some interesting patterns about when and how different architectures work.

The main finding: on high-quality datasets with balanced classes and dense networks like OGB arXiv, all three models (GAT, GCN, GraphSAGE) perform similarly at 50-53% accuracy for node classification, though GCN dominates link prediction at 90% AUC. But on challenging datasets with class imbalance and sparse features like AMiner, GraphSAGE achieves 71% while alternatives struggle at 28-43%. This suggests GraphSAGE is the safest default choice for academic network applications, while GAT and GCN may offer advantages on specific high-quality datasets.

Beyond the performance numbers, I made some contributions to practical deployment. The K-nearest neighbor subgraph construction enables predictions on completely new papers without retraining. The interactive web interface with knowledge graph visualizations provides transparency and helps users understand predictions. The system successfully processes PDFs, generates real-time classifications, and presents results through intuitive visualizations—all important for real-world adoption.

That said, there are significant limitations. The static datasets don't capture temporal dynamics or emerging research trends. The single-label classification doesn't represent genuinely interdisciplinary work. TF-IDF features for new papers are less informative than pre-trained embeddings. And the Streamlit deployment lacks production-grade infrastructure, authentication, and scalability.

Looking forward, the path to broader impact is fairly clear. Integrating pre-trained scientific language models like SciBERT or SPECTER would dramatically improve text understanding. Expanding to heterogeneous graphs that model citations, co-authorship, and venue relationships would capture richer academic context. Implementing multi-label classification with hierarchical taxonomies would better reflect research reality. Developing browser extensions and APIs would enable seamless integration with existing research workflows. And conducting user studies would validate practical utility.

The exponential growth of academic literature shows no signs of slowing. As knowledge accumulates, effective organization and discovery tools become increasingly important. Research Compass demonstrates one approach to this challenge—not as a complete solution, but as a step toward more intelligent, transparent, and useful systems for navigating the expanding landscape of human knowledge.

References

[1] Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30.

[2] Hu, W., et al. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33.

[3] Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations.*

[4] Tang, J., et al. (2008). ArnetMiner: Extraction and mining of academic social networks. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 990-998.

[5] Veličković, P., et al. (2018). Graph attention networks. *International Conference on Learning Representations.*

[6] Wu, Z., et al. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4-24.