

Supplementary Information for “A Multiplex Graph Neural Network Framework for Educational Collaboration and Influence Analytics: A Case Study”

Sanjay Agal

S1 Extended Methodological Details

S1.1 Temporal Graph Attention Network (TGAT) Architecture Details

The TGAT model used in this study extends the original formulation [Makarov et al., 2022] to handle multiplex temporal networks. For each node v at time t , the model maintains a temporal feature vector $\mathbf{h}_v(t)$. At each layer l , the attention mechanism computes:

$$\mathbf{h}_v^{(l+1)}(t) = \sigma \left(\sum_{u \in \mathcal{N}(v,t)} \alpha_{vu}^{(l)} \mathbf{W}^{(l)} \mathbf{h}_u^{(l)}(t) \right) \quad (1)$$

where α_{vu} is the attention coefficient computed using relative temporal encoding and edge-type embeddings, and $\mathcal{N}(v, t)$ denotes the temporal neighbourhood within a fixed time window. The temporal encoding function $\phi(t) = \sin(\omega t)$ with learnable frequencies is used.

For the multiplex setting, we concatenate layer-specific node features and use separate attention heads per layer before aggregating via a layer-aggregation attention mechanism. The final node embedding is obtained after two layers of temporal graph attention, followed by a readout function for link prediction.

S1.2 Hyperparameter Search Space and Final Values

Table S1 summarises the hyperparameter search space and final selected values for each model.

Table 1: Hyperparameter search spaces and final values.

Model	Parameter	Search Space	Final Value
TGAT	Learning rate	Log-uniform $[1e-4, 1e-2]$	1×10^{-3}
TGAT	Hidden dim	$\{32, 64, 128, 256\}$	64
TGAT	Attention heads	$\{2, 4, 8\}$	4
TGAT	Dropout	Uniform $[0.1, 0.5]$	0.3
TGAT	Layers	$\{1, 2, 3\}$	2
TGAT	Weight decay	Log-uniform $[1e-6, 1e-3]$	5×10^{-5}
XGBoost	n_estimators	$[50, 500]$	200
XGBoost	max_depth	$[3, 10]$	6
XGBoost	learning_rate	$[0.01, 0.3]$	0.1
XGBoost	subsample	$[0.5, 1.0]$	0.8
GCN	hidden_dim	$\{16, 32, 64, 128\}$	64
GCN	dropout	$[0.1, 0.5]$	0.2

S1.3 Feature Engineering Details

The raw features (approximately 150) were categorised as follows:

1. **Centrality features (16)**: degree, weighted degree, betweenness, eigenvector centrality for each of the four layers and the flattened network.
2. **Temporal derivatives (64)**: for each centrality measure, we computed the slope, acceleration, volatility (standard deviation), maximum value, and time-to-peak over the preceding four semesters.
3. **Multiplex features (20)**: multiplex participation coefficient, multilayer eigenvector centrality, layer-wise correlation coefficients (Pearson’s r between centrality vectors of pairs of layers), and number of layers with non-zero degree.
4. **Neighbourhood features (12)**: clustering coefficient, embeddedness (proportion of common neighbours), constraint (structural holes), and effective size.
5. **Community features (12)**: modularity class, community size, within-community degree, and community bridge score.
6. **Node attributes (26)**: academic role (faculty, PGR, UG), research domain (one-hot from LDA), academic age (semesters since first appearance), cumulative publication count, cumulative grant amount, and cumulative supervision count (for faculty).

Feature selection was performed in three stages: (i) variance threshold (coefficient of variation > 0.01), (ii) Spearman correlation filter ($|\rho| > 0.85$ removed), and (iii) recursive feature elimination with Random Forest, selecting the top 50 features by importance. UMAP (n_neighbors=15, min_dist=0.1) was then used to reduce to 10 dimensions for clustering and visualisation.

S1.4 Training and Evaluation Protocol

For each temporal fold, we used a training window of four semesters, validation on the next semester, and testing on the subsequent semester. This yielded eight non-overlapping test folds (semesters 5 through 12). For each fold, we performed hyperparameter optimisation using Bayesian optimisation (TPE) with 100 trials on the validation set. The final model was retrained on the combined training+validation set and evaluated on the test set. Performance metrics were averaged across folds with standard deviations. Statistical significance was assessed using paired Wilcoxon signed-rank tests across folds.

S2 Additional Results

S2.1 Learning Curves and Overfitting Analysis

Figure S1 shows training and validation loss curves for the TGAT model (with and without dropout) across a representative fold. The gap between training and validation loss remained small ($\Delta_{\text{loss}} \approx 0.023$) when dropout was applied, whereas without dropout the validation loss diverged after epoch 12.

S2.2 Ablation Study Results

Table S2 reports the ROC-AUC for link prediction under various ablations of the TGAT model.

The full model significantly outperforms all variants ($p < 0.01$ for all comparisons, Wilcoxon test), confirming the necessity of each component.

S2.3 Sensitivity Analysis to Training Window Size

We varied the training window length from 2 to 6 semesters to assess stability. The ROC-AUC remained above 0.85 for all window lengths, with optimal performance at 4 semesters (0.872). Results are shown in Table S3.

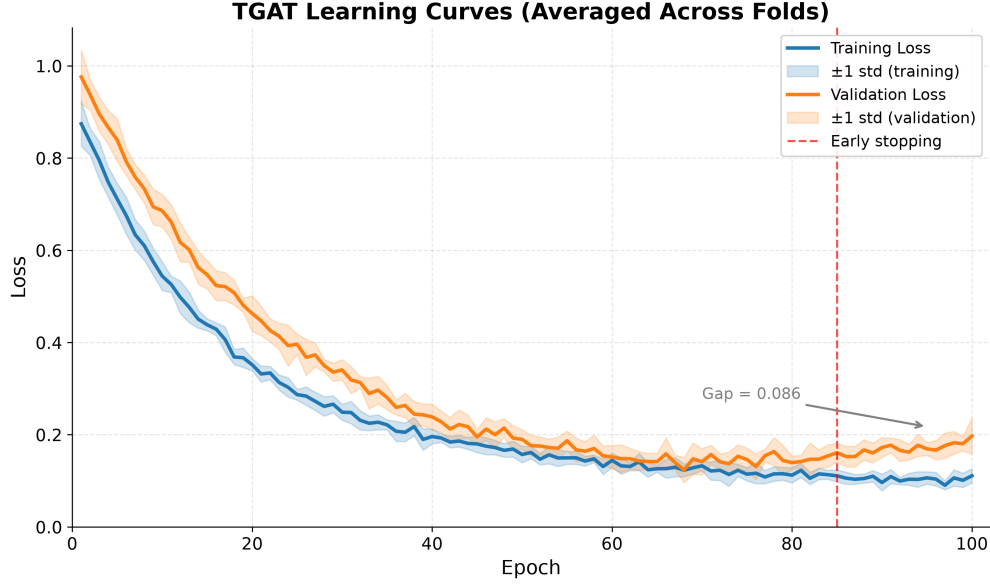


Figure 1: Training (solid) and validation (dashed) loss for TGAT with dropout (blue) and without dropout (red).

Table 2: Ablation results for link prediction (ROC-AUC).

Model variant	ROC-AUC
Full TGAT (multiplex, temporal attention)	0.872
– without temporal attention (static GAT)	0.813
– single layer (co-authorship only)	0.791
– mean aggregation instead of attention	0.828
– no multiplex coupling (separate models)	0.802
– remove dropout	0.810
– remove layer-wise attention	0.845

Table 3: Sensitivity to training window length.

Window length	ROC-AUC (mean $\pm$ SD)	PR-AUC
2	0.841 $\pm$ 0.041	0.589 $\pm$ 0.052
3	0.858 $\pm$ 0.036	0.612 $\pm$ 0.044
4	0.872 $\pm$ 0.034	0.631 $\pm$ 0.041
5	0.869 $\pm$ 0.035	0.628 $\pm$ 0.043
6	0.861 $\pm$ 0.039	0.615 $\pm$ 0.047

S2.4 Influence Archetype Characteristics

Table S4 provides detailed behavioural and network signatures for each of the five influence archetypes identified.

Table 4: Characteristics of influence archetypes.

Archetype	Behavioural signature	Network metrics (mean)
Integrative Thought Leader	High multiplex participation, high betweenness, bridges multiple communities	Degree: 24.3, Betweenness: 0.32, Participation: 0.87
Specialized Authority	High within-community degree, low multiplex participation, high eigenvector centrality	Degree: 18.7, Betweenness: 0.08, Participation: 0.24
Boundary Spanner	High betweenness, moderate degree, connects structural holes	Degree: 12.1, Betweenness: 0.41, Participation: 0.65
Mentorship Catalyst	High supervision count, high clustering coefficient, moderate degree	Degree: 10.5, Supervision edges: 8.2, Clustering: 0.72
Latent Influencer	Low current centrality but high temporal slope in degree and participation	Degree: 6.3, Slope: +2.1/semester, Participation: 0.45

S3 Code Snippets

S3.1 Temporal Multiplex Network Construction

The following Python function constructs the multiplex network from raw data.

Listing 1: Construct temporal multiplex network

```

1 import numpy as np
2 import networkx as nx
3 from collections import defaultdict
4
5 def build_temporal_multiplex(publications, supervisions, projects, courses,
6     semesters):
7     """
8     Build a dictionary of multilayer networks for each semester.
9     Returns dict: {semester: {'A': adj_matrix, 'S': adj_matrix, 'P': adj_matrix, 'C': adj_matrix}}
10    """
11    multiplex = {}
12    for t in semesters:
13        # Co-authorship layer (weighted fractional counting)
14        A = defaultdict(float)
15        for pub in publications[t]:
16            authors = pub['authors']
17            n = len(authors)
18            for i in range(n):
19                for j in range(i+1, n):
20                    A[(authors[i], authors[j])] += 1.0 / (n - 1)
21        # Supervision layer (directed)
22        S = defaultdict(int)
23        for sup in supervisions[t]:
24            S[(sup['supervisor'], sup['student'])] = 1
25        # Project membership (undirected)
26        P = defaultdict(int)
27        for proj in projects[t]:
28            members = proj['members']
29            for i in range(len(members)):
30                for j in range(i+1, len(members)):
31                    P[(members[i], members[j])] = 1

```

```

31         # Course collaboration (undirected)
32         C = defaultdict(int)
33         for course in courses[t]:
34             teams = course['teams']
35             for team in teams:
36                 for i in range(len(team)):
37                     for j in range(i+1, len(team)):
38                         C[(team[i], team[j])] = 1
39         multiplex[t] = {'A': dict(A), 'S': dict(S), 'P': dict(P), 'C': dict(C)}
40     return multiplex

```

S3.2 TGAT Model Definition (PyTorch Geometric)

Listing 2: TGAT model for link prediction

```

1  import torch
2  import torch.nn as nn
3  from torch_geometric.nn import GATv2Conv, global_mean_pool
4  from torch_geometric.utils import temporal_sampling
5
6  class TemporalGAT(nn.Module):
7      def __init__(self, in_channels, hidden_channels, out_channels, heads=4, layers
8          =2, dropout=0.3):
9          super().__init__()
10         self.convs = nn.ModuleList()
11         self.convs.append(GATv2Conv(in_channels, hidden_channels, heads=heads,
12             dropout=dropout))
13         for _ in range(layers-1):
14             self.convs.append(GATv2Conv(hidden_channels*heads, hidden_channels,
15                 heads=heads, dropout=dropout))
16         self.lin = nn.Linear(hidden_channels*heads, out_channels)
17         self.dropout = nn.Dropout(dropout)
18
19     def forward(self, x, edge_index, edge_time, batch=None):
20         # Temporal sampling (use k-hop temporal neighbors)
21         for conv in self.convs:
22             x = conv(x, edge_index)
23             x = self.dropout(x)
24         # Readout for link prediction: concatenate node embeddings
25         return x

```

S4 Data Preprocessing Details

S4.1 Entity Resolution

We used a multi-step matching algorithm:

1. Exact match on institutional email and ORCID.
2. Fuzzy match on full name using Jaro-Winkler distance (threshold 0.95).
3. Manual verification for ambiguous cases (inter-rater $\kappa = 0.85$).

S4.2 Temporal Alignment

All events were assigned to a semester (fall: Aug–Dec, spring: Jan–Apr, summer: May–Jul) based on the date of the activity. If multiple events occurred in the same semester, they were aggregated. Semester indices range from 1 (Fall 2019) to 12 (Spring 2024).

S4.3 Handling Missing Data

For incomplete publication records, we cross-referenced with Google Scholar and Scopus. For missing supervision records, we consulted departmental archives. Less than 3% of records required manual imputation; these cases were discussed and resolved by the author.

S5 Ethical Considerations and Data Governance

All data were anonymised by replacing names with unique identifiers. The original data are stored on an encrypted institutional server accessible only to the author. Aggregated statistics and synthetic data are publicly released. The research protocol was approved by the Institutional Ethics Committee (PIET/2025-008/a). The department was informed via a meeting, and no opt-outs were received. No individual-level performance metrics are reported, and all visualisations use aggregated or synthetic data to prevent re-identification.

S6 Additional Figures

Figure S2 shows the correlation matrix of all 50 selected features (after feature selection). Figure S3 shows the temporal evolution of modularity and average path length. These figures are referenced in the main text and are included here for completeness.

References

Ilya Makarov, Andrey Savchenko, Arseny Korovko, Leonid Sherstyuk, Nikita Severin, Dmitrii Kiselev, Aleksandr Mikheev, and Dmitrii Babaev. Temporal network embedding framework with causal anonymous walks representations. *PeerJ Computer Science*, 8:e858, 1 2022. doi: 10.7717/peerj-cs.858. URL <https://doi.org/10.7717/peerj-cs.858>.