الجمهوريـــــة الجزائريـــــة الديمقراطيـــــة الشـعبيـــــة
République Algérienne Démocratique et Populaire
وزارة التعليـــم العالـــي والبحـــث العلمـــي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعـــة غـــردايـــــــة
Université de GHARDAIA

N° d'enregistrement

:

/..../..../..../

كـــليـــة العـــلـــوم والتكنولـوجيـــا
Faculté des Sciences et de la Technologie
قســـم الرياضيـــات والاعـــلام الآلـــــي
Département des Mathématiques et d'Informatique
مخبر الرياضيـــات و العـــلـــوم التطبيقيـــة
Laboratoire des Mathématiques et Sciences Appliquées

T H È S E

Soumis en exigence pour l'obtention du diplôme de **Doctorat** (Troisième Cycle)

Spécialité : Systèmes intelligents et apprentissage automatique

Présentée par

Abdelfateh BEKKAIR

Approches profondes pour la détection de communautés dans le Big Data

Soutenue publiquement le 1er décembre 2025

Devant le jury composé de :

Dr. Abdelkrim KINA	MCA	Université de Ghardaia	Président
Pr. Abdelouahab MOUSSAOUI	Prof	Université de Setif 1	Examinateur
Pr. Hadda CHERROUN	Prof	Université de Laghouat	Examinateur
Dr. Attia NEHAR	MCA	Université de Djelfa	Examinateur
Dr. Slimane BELLAOUAR	MCA	Université de Ghardaia	Directeur
Dr. Slimane OULED-NAOUI	MCA	Université de Ghardaia	Co-Directeur
Pr. Djelloul ZIADI	Prof	Université de Normandie (France)	Invité

الجمهـوريـــة الجزائـريـــة الديمقـراطيـــة الشـعبيــة
People's Democratic Republic of Algeria
وزارة التعليـــم العالـــي والبحـــث العلمـــي
Ministry of Higher Education and Scientific Research
جامعـــة غـــردايــــــة
University of Ghardaia
كـــليــة العـــلـــوم والتكنولـوجيـــا
Faculty of Science and Technology
قســـم الرياضيـــات والاعــلام الآلـــــي
Department of Mathematics and Computer Science
مخبـر الرياضيـــات و العـــلـــوم التطبيقيـــة
Mathematics and Applied Science Laboratory

Registration n°:
/..../..../..../

THESIS

Submitted as a requirement for obtaining the degree of **Doctorate** Third Cycle

Specialty: Intelligent Systems and Machine Learning

Presented by

Abdelfateh BEKKAIR

Deep Approaches for Community Detection in Big Data

Publicly defended on December 1, 2025

Jury:

Dr. Abdelkrim KINA	MCA	Ghardaia University	Chair
Pr. Abdelouahab MOUSSAOUI	Prof	Setif 1 University	Examiner
Pr. Hadda CHERROUN	Prof	Laghouat University	Examiner
Dr. Attia NEHAR	MCA	Djelfa University	Examiner
Dr. Slimane BELLAOUAR	MCA	Ghardaia University	Supervisor
Dr. Slimane OULED-NAOUI	MCA	Ghardaia University	Co-Supervisor
Pr. Djelloul ZIADI	Prof	Normandie University (France)	Invited

بسم الله الرحمن الرحيم

قال تعالى

وَقُلْ رَبِّ زِدْنِي عِلْمًا

سورة طه - الأية 114

واضحة الدلالة على فضل العلم؛ لأن الله تعالى لم يأمر نبيه ﷺ بطلب الازدياد من شيء إلا من العلم.

ابن حجر العسقلاني

[فتح الباري 1/141]

إهداء

الحمد لله الذي بنعمته تتم الصالحات، وبفضله تتحقق النجاحات وتُثمر الجهود المباركة.

أهدي هذا النجاح

إلى جنتي أمي... نبض الدعاء ودفء الطريق، وعين الخير التي لا تنطفئ.
وإلى ضلعي الثابت أبي... الذي علّمني أن الطموح لا يكتمل إلا بصدق العمل وثبات النيّة.
دمتما لي عونًا وسندًا وبركة لا تنقطع.

وإلى إخوتي وأخواتي الأعزاء، شركاء الفرح الأول وداعمو القلب في كل مراحل المسير...
أسأل الله أن يحفظكم، ويرفع قدركم، ويجعل أيامكم عامرة بالنور والتوفيق.

وإلى كل من مرّ في حياتي وترك درسًا أو دعمًا، شكرًا لكم، وإن غاب ذكركم،
أسأل الله أن يجعله حجة لي لا عليّ، وبابًا لخير لا ينقطع.

Acknowledgments

I thank God Almighty first and foremost.

*I would like to express my deepest gratitude to my supervisor, **Slimane BELLAOUAR**, for his invaluable guidance, encouragement, and constant support throughout this research and my academic journey from the Bachelor's to the doctorate. This thesis would not have been possible without his motivation, inspiration, and insightful advice.*

*I also wish to express my sincere appreciation to my co-supervisor, **Slimane OULED-NAOUI**, for his guidance, encouragement, and continuous support throughout my academic journey from the Bachelor's to the doctorate and during the course of this work.*

I extend my sincere appreciation to the jury members for accepting to review and evaluate my thesis.

I also wish to thank my colleagues and friends for their support, motivation, and companionship during this journey.

Abdelfateh BEKKAIR

ملخص

يعد تحليل الشبكات أداة لا غنى عنها لفك شفرة العلاقات المعقدة والمبادئ التنظيمية الكامنة في الأنظمة المعقدة. في هذا المجال، تتناول هذه الأطروحة التحدي الأساسي المتمثل في اكتشاف التجمعات في الشبكات المعقدة، مع التركيز بشكل خاص على الرسوم البيانية المرفقة (attributed graphs) حيث تتوفر البنية الطوبولوجية وخصائص العقد على حد سواء. ويبقى دمج هذه المعلومات الغنية بشكل فعال لتحديد تجمعات متماسكة أمر بالغ الأهمية لفهم الأنظمة المعقدة. يساهم هذا العمل في هذا المجال من خلال تقديم فهم منهجي للمنهجيات الموجودة ومساهمات خوارزمية جديدة. أولًا، ننشئ تصنيف شامل لتقنيات كشف التجمعات عبر النماذج الكلاسيكية، ونماذج التعلم الآلي التقليدي، ونماذج التعلم العميق، مما يوفر مراجعة منظمة لأحدث ما توصلت إليه الأبحاث. و بناءً على ذلك، تحدد الدراسات المقارنة الدقيقة على نماذج شبكات الرسوم البيانية العصبية (GNN) البارزة، بما في ذلك المقاربات القائمة على الشبكات العصبية الالتفافية (CNNs) ومقاربات مُشَفِّرات الرسوم البيانية التلقائية (GAEs)، حدود الأداء الحالية. تشمل مساهماتنا المنهجية AA-LPA، وهي خوارزمية إرشادية كلاسيكية تعزز خوارزمية انتشار التسميات (LPA). بواسطة الاستفادة من مؤشر Adamic-Adar ودمج آليات حتمية لتحديد أولويات العقد وحل التعادلات، يحسن AA-LPA بشكل كبير الاستقرار والمتانة ضد العشوائية في الرسوم البيانية غير المرفقة. وتمثل المساهمة الأساسية في G2ACO، وهو مُشَفِّر رسوم بيانية تلقائي عميق جديد للشبكات المرفقة. يدمج G2ACO هدف التجميع K-means مع إعادة البناء، بهدف تعظيم المسافات بين التجمعات وتقليل المسافات داخل التجمعات. ويعد الإسهام الرئيس هو استراتيجيته الفريدة للتحسين، التي تفصل تحديثات نقاط مركز K-means عن انتشار الانحدار للتعامل بفعالية مع عدم القابلية للاشتقاق، مما يضمن تدريبًا مستقرًا وأداءً قويًا في تعلم تمثيلات العقد الموجهة نحو التجميع عبر الانتباه متعدد الرؤوس (multi-head attention). توضح التقييمات التجريبية على مجموعات بيانات مرجعية مختلفة أن أسالينا توفر حلولًا فعالة. يتم اختبار AA-LPA على مجموعات بيانات الشبكات الاجتماعية الكلاسيكية، حيث يعزز الاستقرار بشكل كبير مقارنة بـ LPA التقليدي. كما يتم تقييم G2ACO بدقة في كل من مجموعات بيانات شبكات الاقتباس والشبكات الاجتماعية، متفوقًا باستمرار على

أحدث خطوط الأساس. تساهم هذه الأطروحة بمعلومات وأدوات قيمة لتحليل الشبكات، وتقدم حلًا لكشف التجمعات في الرسوم البيانية المعقدة والمرفقة.

كلمات مفتاحية : كشف التجمعات، شبكات الرسوم البيانية العصبية، الشبكات المرفقة، مشفرات الرسوم البيانية التلقائية، تمثيل العقد، الشبكات الاجتماعية، شبكات الاقتباس، خوارزمية انتشار التسميات.

Résumé

L'analyse de réseaux est un outil indispensable pour découvrir les relations et les principes organisationnels inhérents aux systèmes complexes. Dans ce domaine, cette thèse aborde le défi fondamental de la détection de communautés dans les réseaux complexes, en se concentrant particulièrement sur les graphes attribués où la structure topologique et les caractéristiques des nœuds sont disponibles. Intégrer efficacement cette information riche pour identifier des communautés cohésives demeure une tâche essentielle à la compréhension des systèmes complexes. Ce travail contribue au domaine en proposant à la fois une compréhension systématique des méthodologies existantes et de nouvelles contributions algorithmiques. Nous établissons d'abord une taxonomie complète qui catégorise les techniques de détection de communautés selon les méthodes classiques, d'apprentissage automatique traditionnel et d'apprentissage profond, offrant une revue structurée de l'état de l'art. S'appuyant sur cela, des études comparatives rigoureuses sur des modèles de réseaux neuronaux sur graphes (GNN) proéminents, y compris les approches basées sur des CNN et des autoencodeurs de graphes (GAE), identifient les limitations de performance actuelles. Nos contributions méthodologiques incluent premièment AA-LPA, un algorithme heuristique classique qui améliore l'algorithme de propagation de labels (LPA). En tirant parti de l'indice Adamic-Adar et en intégrant des mécanismes déterministes pour la priorisation des nœuds et la résolution des égalités, AA-LPA améliore significativement la stabilité et la robustesse face à l'aléatoire dans les graphes non attribués. La seconde contribution principale est G2ACO, un nouvel autoencodeur de graphes profond pour les réseaux attribués, qui intègre un objectif de regroupement K-means à la reconstruction dans le but de maximiser les distances inter-communautés et de minimiser les distances intra-communautés. Une innovation clé est sa stratégie d'optimisation unique, qui dissocie les mises à jour des centroïdes K-means de la propagation du gradient pour gérer efficacement la non-différentiabilité, assurant une formation stable et une performance robuste dans l'apprentissage de représentations de nœuds orientées vers le regroupement via l'attention multi-têtes. Les évaluations empiriques sur divers ensembles de données de référence démontrent que nos méthodes fournissent des solutions efficaces. AA-LPA est testé sur des ensembles de données de réseaux sociaux classiques, où il améliore significativement la stabilité par rapport au LPA traditionnel. En outre, G2ACO est rigoureusement évalué sur des ensembles de données de réseaux de citation et de réseaux sociaux, surpassant de manière constante les approches de référence de l'état de l'art. Cette thèse apporte des connaissances et des outils précieux pour l'analyse de réseaux, offrant une solution complète pour la détection de communautés dans les graphes complexes et attribués.

Mots clés : Détection de communautés, Réseaux Neuronaux sur Graphes, réseaux attribués, autoencodeurs de graphes, représentation de nœuds, réseaux sociaux, réseaux de citation, algorithme de propagation de labels.

Abstract

The intricate relationships and organizational principles inherent in complex systems are effectively deciphered through network analysis, an indispensable tool in this regard. Within this field, this thesis addresses the fundamental challenge of community detection in complex networks, particularly focusing on attributed graphs where both topological structure and node features are available. Effectively integrating this rich information to identify cohesive communities remains a critical task in understanding complex systems. This work contributes the field by offering both a systematic understanding of existing methodologies and novel algorithmic contributions. We first establish a comprehensive taxonomy that categorizes community detection techniques across classical, traditional machine learning, and deep learning paradigms, providing a structured state-of-the-art review. Building on this, rigorous comparative studies on prominent graph neural network models, including CNN-based and Graph Autoencoder (GAE)-based approaches, identify current performance limitations. Our methodological contributions include AA-LPA, a classical heuristic algorithm that enhances the Label Propagation Algorithm (LPA). By leveraging the Adamic-Adar index and incorporating deterministic mechanisms for node prioritization and tie resolution, AA-LPA significantly improves stability and robustness against randomness in non-attributed graphs. The primary contribution is G2ACO, a novel deep graph autoencoder for attributed networks. G2ACO integrates a K-means clustering objective with reconstruction, with the aim of maximizing inter-community and minimizing intra-community distances. A key innovation is its unique optimization strategy, which decouples K-means centroid updates from gradient propagation to effectively handle non-differentiability, ensuring stable training and robust performance in learning clustering-oriented node representations via multi-head attention. Empirical evaluations on various benchmark datasets demonstrate that our methods provide effective solutions. AA-LPA is tested on classical social network datasets, where it significantly enhances stability compared to traditional LPA. Moreover, G2ACO is rigorously evaluated in both citation network datasets and social networks, consistently outperforming state-of-the-art baselines. This thesis contributes to valuable insights and tools for network analysis, providing a comprehensive solution for community detection in complex and attributed graphs.

Keywords : Community detection, Graph Neural Networks, attributed networks, graph autoencoders, node representation, social networks, citation networks, Label Propagation Algorithm.

Academic Publications

This section lists peer-reviewed publications derived from the research presented in this thesis.

Journal Publication

- Bekkair, A., Bellaouar, S., and Oulad-Naoui, S. (2025). *Unsupervised graph attention autoencoder clustering-oriented for community detection in attributed networks*. *International Journal of Data Science and Analytics*. Quality: Q2, Impact factor: 2.9. DOI: `https://doi.org/10.1007/s41060-025-00800-4`.

  ✓ Discussed in Chapter 4.

Conference Publications

- Bekkair, A., Oulad-Naoui, S., Bellaouar, S., Krimat, R.R., and Haddaoui, A. (2023). *CNN-based community detection: A comparison study*. In *Proceedings of the 2023 5th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*. IEEE. DOI: `https://doi.org/10.1109/PAIS60821.2023.10322072`.

  ✓ Discussed in Chapter 3.

- Bekkair, A., Bellaouar, S., Oulad-Naoui, S., and Zita, K. (2024). *A comparative study of graph autoencoder-based community detection in attributed networks*. In *Proceedings of the 2024 International Conference on Telecommunications and Intelligent Systems (ICTIS)*. IEEE. DOI: `https://doi.org/10.1109/ICTIS62692.2024.10893950`.

  ✓ Discussed in Chapter 3.

- Bekkair, A., Bellaouar, S., and Oulad-Naoui, S. (2025). *Community Detection via Enhanced Label Propagation with Adamic-Adar Similarity and Community Size Influence.* In *Proceedings of the 2025 7th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*. IEEE. DOI: https://doi.org/10.1109/PAIS66004.2025.11126491.

    ✓ Discussed in Chapter 4.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations and Notations

The following list provides a concise reference for all the abbreviations and symbols used throughout this thesis.

Abbreviations

AA	*Adamic–Adar index*
ARI	*Adjusted Rand Index*
BFS	*Breadth–First Search*
CNN	*Convolutional Neural Network*
CD	*Community Detection*
DFS	*Depth–First Search*
GAE	*Graph AutoEncoder*
GAT	*Graph Attention Network*
GCN	*Graph Convolutional Network*
GNN	*Graph Neural Network*
GraphSAGE	*Graph SAmple and aggreGatE*
LPA	*Label Propagation Algorithm*
NMF	*Nonnegative Matrix Factorization*
NMI	*Normalized Mutual Information*

PCA	*Principal Component Analysis*
SSL	*Semi-supervised Learning*
VAE	*variationnel Autoencodeurs*
VGAE	*Variational Graph Autoencoder*

Notations

$\mathbf{A}$	Adjacency matrix
$\mathbf{B}$	Modularity matrix
$\mathbf{X}$	Attribute matrix, size $N \times m$ (also represents data samples in ML tasks)
$\mathbf{Z}$	Latent representation matrix
$\mathbf{x}_i$	Attribute vector of node i (also represents individual data sample in ML contexts), $\mathbf{x}_i \in \mathbb{R}^m$
$\mathbf{z}_i$	Latent vector of node i
C	Community structure set
d	Embedding dimension
E	Edge set
$\mathcal{G}$	A graph
H	Number of attention heads
k	Number of communities
m	Number of attributes per node
N	Number of nodes
M	Number of edges

α	Attention coefficient
λ	Reconstruction weight hyperparameter
β	Clustering weight hyperparameter
ϕ_{enc}	Encoder function
ϕ_{dec}	Decoder function
$\mathcal{L}$	Overall loss function
$\mathcal{L}_r$	Reconstruction loss
$\mathcal{L}_c$	Clustering loss

Introduction

Context

Across the field of network analysis, one of the most essential and long-standing problems is community detection (also known as graph clustering or network partitioning). While the investigation of communities has roots as early as the 1920s in sociology and social anthropology (Rice, 1927), it is predominantly after the turn of the 21st century that advanced scientific tools were developed and applied primarily to real-world data (Newman and Girvan, 2004).

In this unsupervised task, the goal is to uncover cohesive clusters of nodes in a network, where the links connecting nodes inside a cluster are stronger and more frequent than those linking them to nodes outside that cluster. This pattern reflects the principle of homophily, which refers to the tendency for similar individuals to associate with one another, as expressed by the adage "Birds of a feather flock together" (McPherson et al., 2001).

The underlying premise is that these communities often correspond to functional modules, organizational units or semantically meaningful clusters within the system. The successful identification of such latent structures is not merely an academic exercise; it yields profound practical implications. For instance, in bioinformatics, it can reveal disease gene modules; in social sciences, it helps understand group dynamics and information flow; and in cybersecurity, it aids in detecting malicious botnets. Therefore, rapidly and reliably detecting these hidden communities is vital to fully leverage network data's analytical strengths and to guide effective decision-making in today's data-centric world.

Motivation

Despite the foundational importance of community detection, existing methodologies face significant limitations, particularly when confronted with the increasingly high-dimensional and complex nature of networks. Traditionally, many community detection approaches have focused almost exclusively on network topology. These include methods such as hierarchical clustering techniques, modularity optimization, spectral clustering, and various dynamic algorithms. While these established methods offer valuable insights into network structure and often yield interpretable results, they frequently suffer from issues such as sensitivity to initial conditions, leading to non deterministic solutions, and inherent difficulties in processing networks that evolve over time.

A more critical challenge arises from the growing availability of attributed network data, where nodes possess rich semantic and feature information beyond basic connectivity. Classical algorithms struggle to efficiently fuse this high dimensional, multi source information. Their inability to jointly leverage both network topology and node attributes often leads to suboptimal community detection, overlooking crucial semantic relationships embedded within the data.

To address these limitations and better harness the complexity of contemporary data, the field has seen a significant shift from older techniques towards more advanced models. This evolution has been particularly driven by the advent of deep learning, which has revolutionized the ability to process and learn from high dimensional data across various domains. In the context of network analysis, this advancement has culminated in the emergence of Graph Neural Networks (GNNs) (Wu et al., 2022). GNNs are specifically designed to operate directly on graph structured data, enabling them to learn powerful, low dimensional representations that effectively capture complex topological and semantic patterns.

Many GNN variants have demonstrated success in extracting meaningful features from graph structured data. Among these, Graph Autoencoders (GAEs) stand out as a particularly promising framework for unsupervised representation learning, as they extend the autoencoder paradigm to handle data in non Euclidean spaces (Thomas and Welling, 2016). GAEs learn a compact latent representation by reconstructing the input graph, thereby effectively reducing high dimensionality while preserving essential structural and attribute information. These models can seamlessly integrate diverse inputs, such as node features, edge attributes, and even precomputed embeddings, into their latent space.

A core component of many advanced GNN architectures is a self-attention mechanism, which assigns importance to relevant neighboring nodes to improve embedding quality. This mechanism is crucial to capture the local network structure and create more powerful node embeddings, and is a key component of several prominent GAE-based clustering methods (Veličković et al., 2018; Wang et al., 2019; Zhou et al., 2023). However, a single attention head may be insufficient to capture the multiple and diverse aspects of relationships among nodes that are essential for nuanced community detection.

To perform clustering within the latent space, many models adopt a dual objective. Several strategies have been proposed to integrate a clustering objective into the GAE's training. One common approach is to employ indirect methods that use probabilistic clustering loss, such as KL (Kullback-Leibler) divergence, to guide the latent space towards a desirable cluster distribution (Wang et al., 2019; Zhou et al., 2023). Although effective, this approach can produce soft assignments that may not always result in maximally separated or distinct communities. Alternatively, deep clustering approaches that seek to directly use a K-means objective to enforce cluster compactness, such as Simultaneous Deep Clustering (SDC) (Yang et al., 2017), often encounter a different set of problems. These methods, while innovative, are primarily designed for Euclidean data such as images and text. This limits their direct applicability to complex, graph structured data for the purpose of community detection, especially as they also struggle with the non differentiability inherent in hard cluster assignments.

This dichotomy highlights a critical research gap. The absence of a unified framework that can learn an enhanced representation and effectively address the clustering objective challenge underscores a significant limitation in the current state-of-the-art.

Contributions

The contributions of this thesis are organized into four main areas. Firstly, this thesis presents a comprehensive state-of-the-art review of community detection techniques. This review is structured around a novel organized taxonomy that categorizes these methods into three main classes: classical approaches, traditional machine learning-based models, and deep learning-based paradigms. This framework effectively highlights the methodological progress in the field, enabling a clear contrast of the advantages and limitations inherent to each class.

Secondly, this work includes two distinct and in-depth comparative studies focused on discovering cohesive communities using Graph Neural Network (GNN) techniques. The first study (Bekkair et al., 2023) rigorously compares CNN-based community detection methods, examining two taxonomized approaches (edge transformation and node transformation) in various non-attributed networks, including Karate, Email datasets and Dolphins (Zachary, 1977; Girvan and Newman, 2002; Lusseau et al., 2003). The second study (Bekkair et al., 2024) provides a thorough analysis and investigation of graph autoencoders (GAEs) for attributed network-based community detection. This study explores two main GAE categories (simple versus dual encoder), each focusing on GCN or GAT-based architectures, evaluating five distinct AE models in major citation networks (CiteSeer, Cora, and PubMed (Giles et al., 1998; McCallum et al., 2000; Sen et al., 2008)).

Our third contribution is the **A**damic-**A**dar **L**abel **P**ropagation **A**lgorithm (AA-LPA) proposal (Bekkair et al., 2025b), a classical heuristic algorithm that improves the traditional Label Propagation Algorithm (LPA) (Raghavan et al., 2007). Based on the Adamic-Adar index measure, AA-LPA significantly improves LPA stability and effectively removes its inherent randomness through several deterministic mechanisms. These include node prioritization via AA-derived centrality to guide propagation order, AA-weighted label influence amplified by community size, and deterministic tie resolution using structural centrality to ensure consistent partitions. The algorithm was rigorously tested on three benchmark datasets (Karate, American Football, and Dolphins) (Zachary, 1977; Girvan and Newman, 2002; Lusseau et al., 2003)).

Ultimately, regarded as the main contribution of this thesis, we propose **G**raph **A**ttention **A**utoencoder **C**lustering-**O**riented (G2ACO) (Bekkair et al., 2025a), a novel deep learning model based on graph autoencoder architecture. G2ACO is specifically designed to detect communities and extract highly useful node embeddings for graph clustering tasks by jointly leveraging both the network topology and node attributes. It achieves this by orienting the learned representations using a K-means objective alongside the reconstruction loss. This K-means objective aims to maximize the distance between communities while minimizing the distance within communities. A key methodological innovation is its unique optimization strategy, which effectively handles the non-differentiability of K-means by decoupling centroid updates from gradient propagation, thereby ensuring stable training and robust performance. This model was fully tested on two different types of attributed networks: citation networks (CiteSeer, Cora, and

PubMed (Giles et al., 1998; McCallum et al., 2000; Sen et al., 2008)) and a social network (BlogCatalog (Li et al., 2015)).

Organization

The structure of this thesis is organized into two main parts:

Part I laid the essential groundwork. Chapter 1 provides a comprehensive background on fundamental concepts in graph theory, machine learning, deep learning, and their specialized applications to graph data. Chapter 2 then offers a review of existing community detection methodologies, culminating in a detailed taxonomy that categorizes approaches from classical methods to contemporary deep learning methods, thus establishing the context and highlighting the research gaps addressed in this work.

Part II constitutes the core contribution of this thesis, presenting original research designed to push the boundaries of community detection. Chapter 3 begins with an in-depth comparative study. This chapter rigorously evaluates prominent community detection techniques across different paradigms, including CNN-based methods for non-attributed networks and Graph Autoencoder (GAE)-based models for attributed networks. The insights derived from this comparative analysis directly inform and justify our subsequent methodological proposals. Chapter 4, details our contributions: AA-LPA, an enhancement for stable classical community detection in non-attributed networks, and G2ACO, a novel deep graph autoencoder with a unique clustering-oriented optimization strategy for attributed graphs.

Finally, Chapter 5 provides the general conclusion of this thesis by revisiting the research problem and objectives, summarizing the main contributions, and outlining future perspectives in the form of open questions and possible extensions.

First Part

Background and State of the Art

« أنَّ مَن آثر الراحة فاتته الراحة »

Ibn Qayyim al-Jawziyya,

Miftaah Daar as-Sa’aadah, 1/142

Chapter 1

Background

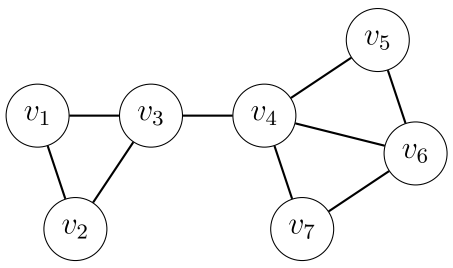
This Background chapter introduces the core concepts that form the foundation of this thesis, bringing together graph-theoretic fundamentals, machine learning paradigms, deep-learning architectures and their adaptations to graph-structured data. By establishing essential vocabulary and guiding intuition, the reader is equipped to fully understand and critically engage with the rest of the thesis.

1.1 Graph Theory

Graph theory was first introduced in 1735 when Leonhard Euler presented the problem of traversing each of the seven bridges of Königsberg exactly once without repetition. This problem laid the foundation for graph theory by modeling the city's layout as a network structure (Euler, 1741). In this network representation, landmasses were depicted as nodes (also known as vertices), and the bridges connecting them as edges. Graph theory provides a powerful mathematical tool for modeling and analyzing complex systems of interconnected elements. It is widely used to represent various real-world networks.

Formally, a *graph* $\mathcal{G}$ is defined by $\mathcal{G} = (V, E)$, where $V = \{v_1, v_2, \dots, v_N\}$ is a collection of $N = |V|$ nodes connected with a set of M edges $E = \{e_1, e_2, \dots, e_M\}$ (Figure 1.1).

An *attributed graph* (Zhang et al., 2018) can be represented as a triplet $\mathcal{G} = (V, E, \mathbf{X})$, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbb{R}^{N \times m}$ is the attribute matrix of size $N \times m$, and each vector $\mathbf{x}_i \in \mathbb{R}^m$ denotes the attribute vector of node i.



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Figure 1.1—A simple undirected graph with seven nodes and its adjacency matrix **A**, which encodes the graph topology.

A *bipartite graph* is one whose vertex set can be partitioned into two disjoint, non-empty subsets U and V, so that every edge joins a vertex in U to one in V (Figure 1.2). More generally, a *k-partite graph* partitions its vertices into k disjoint subsets, with no edges within the same subset. A *complete k-partite graph*, denoted $K_{n_1,n_2,\ldots,n_k}$, is one in which every possible edge between vertices in different subsets is present.

A *path* in a graph is a sequence of vertices in which each consecutive pair is joined by an edge, and no vertex (and hence no edge) is repeated. In contrast, a *walk* allows vertices and edges to be revisited. A *subgraph* of a graph $\mathcal{G} = (V, E)$ is any graph $\mathcal{G}' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$, such that each edge in E' connects vertices in V' (Sedgewick and Wayne, 2011; Barabasi and Pósfai, 2016).

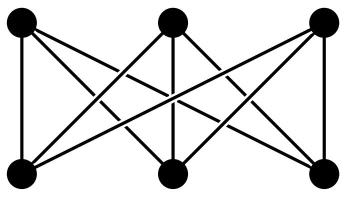


Figure 1.2—A complete bipartite graph $\mathbf{K_{3,3}}$.

Node Centrality

A key concept in graph theory and network analysis is the notion of *node centrality*, which quantifies the importance or influence of a node based on its position within the graph. One of the simplest measures is *degree centrality*, defined for a node v as follows:

$$C_D(v) = \frac{\deg(v)}{N-1}, \tag{1.1}$$

where deg(v) is the number of edges incident to v and $N-1$ is the maximum possible degree in a simple undirected graph. This normalization ensures $0 \leq C_D(v) \leq 1$.

A more global measure is *betweenness centrality*, which captures the extent to which a node lies on shortest paths between other node pairs. For each ordered pair of distinct nodes (s,t), let σ_{st} denotes the number of shortest paths from s to t, and let $\sigma_{st}(v)$ be the number of those paths that pass through v. The betweenness centrality of v is then

$$C_B(v) = \sum_{\substack{s,t \in V \\ s \neq t \neq v}} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1.2}$$

High betweenness indicates a node's critical role in connecting disparate parts of the graph. Other commonly used centrality measures include *closeness centrality*, which is based on the average shortest-path distance from a node to all others, and *eigenvector centrality*, which assigns higher scores to nodes connected to other high-scoring nodes. Collectively, these metrics provide complementary views of node importance, from local connectivity to global flow of information (Erciyes, 2021).

Graph Representations

Depending on their intended application, a graph $\mathcal{G} = (V, E)$ may be encoded in any of several formats. In the *adjacency matrix* representation, one associates to each pair of vertices $v_i, v_j \in V$ the entry

$$a_{ij} = \begin{cases} 1, & (v_i, v_j) \in E, \\ 0, & \text{otherwise}, \end{cases} \tag{1.3}$$

yielding an $N \times N$ binary matrix (Figure 1.1). This format permits $O(1)$ time adjacency queries at the cost of $O(n^2)$ storage. Alternatively, the *adjacency list* representation assigns to every node $v \in V$ the list of its neighbors $\mathcal{N}(v) = \{\, u \,|\, (v,u) \in E\}$, which uses $O(N+M)$ space and allows one to iterate efficiently over all edges incident to a given node. Finally, the *edge list* simply records the set $\{(u,v) \mid (u,v) \in E\}$, providing a minimal $O(|E|)$-size description of the graph's connections.

Graph Laplacians

In addition to the adjacency-matrix, adjacency-list, and edge-list representations, many algorithms rely on the *graph Laplacian* (Hamilton, 2020). Let $\mathcal{G} = (V, E)$ be an undirected graph, adjacency matrix $\mathbf{A} = [a_{ij}]_{i,j=1}^{N}$, and degree matrix $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$ where $d_i = \sum_j a_{ij}$. The (combinatorial) Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

By construction, $\mathbf{L}$ is symmetric positive semi-definite, and its spectrum (the set of eigenvalues) encodes connectivity properties. For instance, the multiplicity of the zero eigenvalue equals the number of connected components in $\mathcal{G}$. A commonly used variant is the symmetric *normalized Laplacian*.

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\,\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\,\mathbf{D}^{-1/2}, \tag{1.4}$$

where $(\lambda_i(\mathbf{L}{*}\text{sym}))$ denotes the (i)-th eigenvalue of the normalized Laplacian, satisfying $0 \leq \lambda_i(\mathbf{L}_{\text{sym}}) \leq 2$. In spectral clustering and graph convolutional networks, L_{sym} ensures that nodes with varying degrees are treated on an equal footing.

Another form is the random-walk (asymmetric) normalized Laplacian.

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}L = \mathbf{I} - \mathbf{D}^{-1}A, \tag{1.5}$$

whose eigenvalues lie in $[0, 2]$ and whose rows sum to zero. $\mathbf{L}_{\text{rw}}$ governs the transition dynamics of a simple random walk on $\mathcal{G}$.

Graph Traversal

Graph traversal is most commonly performed by two complementary algorithms, *breadth-first search* (*BFS*) (Moore, 1959) and *depth first search* (*DFS*) (Tarjan, 1971). Starting from a designated source vertex s, BFS proceeds "level by level," visiting first those vertices at distance one from s, then those at distance two, and so on. In contrast, DFS explores as deeply as possible along each branch before backtracking: starting from s, it selects an unvisited neighbor and continues along that path until no new vertices remain, then backtracks to explore alternative branches. DFS is often implemented recursively or with an explicit stack. Together, these representations and traversal strategies form the foundation for efficient storage, querying, and analysis of complex network structures.

Network Categories

In network analysis, data are commonly organized into categories such as social networks, citation networks, biological networks, and various other graph types, each characterized by unique structural properties and applications (Fortunato, 2010).

Social networks model the relationships and interactions among individuals or organizations, uncovering patterns of influence, information diffusion, and community structure. Prominent examples include Facebook's friendship graph, where vertices represent users and undirected edges denote mutual "friend" links, and interaction networks on platforms such as Instagram or TikTok, which researchers use to study how behaviors (for example, news sharing or product adoption) propagate through tightly-knit user clusters.

Citation networks represent scholarly documents as nodes and directed edges as citations, tracing the propagation of ideas, the emergence of research fronts, and the scholarly impact of individual works. Unlike social networks, citation graphs are inherently acyclic because a paper cannot cite itself directly or indirectly. The high-energy physics arXiv citation network[1], for example, has been analyzed to identify influential papers and to map the temporal evolution of scientific disciplines (Leskovec et al., 2005).

Biological networks capture molecular and cellular interactions such as protein–protein associations, gene regulatory relationships, or metabolic pathways, revealing functional modules, signaling cascades, and mechanisms underlying disease. The *Saccharomyces cerevisiae* protein–protein interaction (PPI) network, in which proteins are nodes and experimentally determined binding events form edges, serves as a canonical dataset for detecting overlapping complexes and predicting protein functions (Barabasi and Oltvai, 2004).

Other networks extend beyond social, citation, and biological domains to include transportation systems (for example, road and airline networks), critical infrastructure (e.g, power grids and water-distribution systems), and technological topologies (e.g, communication networks, the World Wide Web, and the Internet's autonomous-system graph). Analyses of these networks yield insights into system resilience, traffic optimization, and vulnerability to failure. For instance, examining how disruptions at key airports cascade through the U.S. domestic airline network.

[1]`http://snap.stanford.edu/data/` (Stanford Large Network Dataset Collection, SNAP). Accessed 2025-10-18.

1.2 Machine Learning

The quest to endow machines with human-like intelligence is a longstanding problem in the field of artificial intelligence. It can be traced back to Alan Turing's seminal question, "Can machines think?" (TURING, 1950). Despite decades of progress, artificial intelligence remains an open and multifaceted research area, comprising numerous subfields. Among these, machine learning—a data-driven methodology for extracting patterns and making predictions, has gained particular prominence. Machine learning itself is organized into several paradigms, including supervised learning, unsupervised learning, semi-supervised learning, and other paradigms like reinforcement learning and self-supervised learning.

1.2.1 Supervised Learning

Supervised learning refers to the subset of machine-learning algorithms that leverage labeled data during training to learn a mapping from inputs to outputs (Bishop, 1995). Concretely, given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, the goal is to infer from a hypothesis space $\mathcal{H}$, a function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes a chosen loss function $\mathcal{L}$:

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(f(\mathbf{x}_i), y_i\big). \tag{1.6}$$

When $\mathcal{Y}$ is a finite set of classes, the problem is known as *classification*, where the objective function typically employs a categorical loss such as the *binary cross-entropy* (*BCE*) for binary classification or *cross-entropy* loss for multiclass problems. Conversely, when $\mathcal{Y}$ is a set of real-valued vectors, the task corresponds to *regression*, commonly optimized using the *mean squared error* (*MSE*) loss.

Supervised learning dominates many practical applications, driven by the increasing availability of annotated data. Advances in data-collection platforms, crowdsourcing, and sensor networks have fueled vast labeled corpora in domains ranging from computer vision to natural language processing (LeCun et al., 2015).

This paradigm also extends naturally to graph-structured data. For example, in node-level tasks, machine learning models are trained to predict node labels by exploiting the structural information encoded within the graph. Classical approaches such as label propagation, graph kernels, and probabilistic relational models leverage topological similarities to infer labels and uncover underlying relational patterns.

1.2.2 Unsupervised Learning

Unsupervised learning refers to the class of machine-learning algorithms that operate without labeled data (Bishop, 1995). Formally, given a dataset $\mathcal{D} = \{\mathbf{x_i}\}_{i=1}^{N}$ with $\mathbf{x_i} \in \mathcal{X}$, the objective is to discover the underlying structure or distribution of the data in the absence of labels y_i. Unsupervised methods are especially powerful when annotated data is scarce or expensive to obtain. In graph-based learning, for example, they enable community detection, node embedding, and anomaly identification based solely on the topology and attributes of the graph (Fortunato, 2010; Perozzi et al., 2014; Grover and Leskovec, 2016; Hamilton, 2020; Su et al., 2022).

Two primary tasks in unsupervised learning are clustering and dimensionality reduction.

1.2.2.1 Clustering

Clustering aims at partitioning the data into K groups such that the points within each cluster are more similar to each other than to the points in other clusters. A widely used method is K-means clustering, which iteratively minimizes the intra-cluster sum of squared distances. Given a dataset $\mathcal{D} = \{\mathbf{x_i}\}_{i=1}^{N} \in \mathbb{R}^{N\times d}$ and a specified K, the K-means alternate between assigning each point to its nearest centroid and updating each centroid to the mean of its assigned points (Krishna and Murty, 1999). Formally, it solves the following problem.

$$\{\hat{C}, \hat{\mu}\} = \arg\min_{C,\mu} \sum_{j=1}^{K} \sum_{\mathbf{x_i}\in C_j} \left\|\mathbf{x_i} - \mu_j\right\|^2, \tag{1.7}$$

where $\hat{C} = \{\hat{C}_1, \ldots, \hat{C}_K\}$ is the set of clusters, $\hat{\mu} = (\hat{\mu}_1, \ldots, \hat{\mu}_K)$ are the cluster centroids, C_j denotes the set of points assigned to cluster j, $\mu_j \in \mathbb{R}^d$ is the centroid of cluster j, and $\mathbf{x}_i \in \mathbb{R}^d$ for $i = 1, \ldots, N$. Here K is the number of clusters, N the number of data points, d the data dimensionality, and $\|\cdot\|$ the Euclidean norm. As detailed in Algorithm 1. The time complexity of the standard K-means algorithm scales as $O(NKdT)$, where T is the number of iterations required for convergence.

Algorithm 1 K-means Clustering

```
**Require:** Dataset $\{\mathbf{x_i}\}_{i=1}^{N} \subset \mathbb{R}^d$, number of clusters $K$, tolerance $\epsilon$
**Ensure:** Cluster assignments $C_1, \dots, C_K$ and centroids $\mu_1, \dots, \mu_K$
 1: Initialize centroids $\{\mu_j^{(0)}\}_{j=1}^{K}$                ▷ Initial centroids at iteration 0
 2: for $t = 1, 2, \dots$ do                                     ▷ Assignment step
 3:     for each $i = 1, \dots, N$ do
 4:         $c_i^{(t)} \leftarrow \arg\min_{j \in \{1,\dots,K\}} \|\mathbf{x_i} - \mu_j^{(t-1)}\|^2$
 5:     end for                                                  ▷ Update step
 6:     for each $j = 1, \dots, K$ do
 7:         $$\mu_j^{(t)} \leftarrow \frac{1}{|C_j^{(t)}|} \sum_{i: c_i^{(t)} = j} \mathbf{x_i}$$
 8:     end for
 9:     if $\max_j \|\mu_j^{(t)} - \mu_j^{(t-1)}\| < \epsilon$ then
10:         break
11:     end if
12: end for
13: return $\{C_j^{(t)}\}, \{\mu_j^{(t)}\}$
```

1.2.2.2 Dimensionality Reduction

Dimensionality reduction techniques project high-dimensional data into a lower-dimensional space while preserving essential structural properties; these methods reduce dimensionality through a mapping:

$$f : \mathcal{X} \subseteq \mathbb{R}^N \to \mathbb{R}^d, \quad d \ll N. \tag{1.8}$$

where $\mathcal{X}$ represents the input feature space containing data samples $\mathbf{x}_i \in \mathcal{X}$, and f transforms each data sample to a low-dimensional representation.

Principal Component Analysis (PCA) (Jolliffe and Cadima, 2016) identifies orthogonal directions of maximal variance through eigen decomposition of the covariance matrix $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$ for mean-centered data $\mathbf{X} \in \mathbb{R}^{N \times m}$. Here, N denotes the number of samples, and m represents the dimensionality (number of features) of each sample. The projection matrix $\mathbf{W}_d \in \mathbb{R}^{m \times d}$ contains the top-d eigenvectors corresponding to the largest eigenvalues, where $d \ll m$ is the reduced dimensionality. The transformed representation is then obtained as $\mathbf{Y} = \mathbf{X}\mathbf{W}_d$, providing a lower-dimensional embedding that preserves the maximum variance in the data.

$$\mathcal{L}_{\text{t-SNE}} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{1.9}$$

with q_{ij} computed from Student's t-distributed embeddings optimized via gradient descent.

Autoencoders (Kramer, 1991) represent an advanced class of nonlinear dimensionality-reduction models, encompassing architectures such as convolutional, variational, and graph autoencoders. They naturally extend to non-Euclidean domains, including graphs, enabling the modeling of complex data distributions and the learning of task-specific embeddings; details are deferred to Section 1.3.4.

These methods enable visualization of complex data structures, noise suppression, and efficient feature extraction for downstream tasks; in graph domains, they are routinely utilized for community detection, node classification, link prediction, anomaly detection, and graph compression.

1.2.3 Semi-supervised Learning

Semi-supervised learning is a learning paradigm that bridges the gap between supervised and unsupervised learning. The objective is to build effective models using a small amount of labeled data combined with a large amount of unlabeled data (Van Engelen and Hoos, 2020). Formally, the training set is composed of two parts: a labeled dataset $\mathcal{D}_l = \{(\mathbf{x}_\mathbf{i}, y_i)\}_{i=1}^{n_l}$ and an unlabeled dataset $\mathcal{D}_u = \{\mathbf{x}_\mathbf{j}\}_{j=1}^{n_u}$, where typically $n_l \ll n_u$. The objective is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that generalizes well to unseen data.

The success of semi-supervised learning relies on several core assumptions (Zhu and Goldberg, 2009). ***The smoothness assumption*** states that points that are close in the input space are likely to have the same label. The ***cluster assumption*** suggests that the data form discrete clusters, and points within the same cluster tend to share labels. Furthermore, the ***manifold assumption*** posits that high-dimensional data lie on a lower-dimensional manifold, and learning is easier in this space.

Numerous methods have been proposed to exploit unlabeled data under these assumptions.

Semi-supervised learning is especially valuable in domains where labeled data is scarce or expensive to obtain, such as bioinformatics (Zhu et al., 2021) and medical imaging (Jiao et al., 2024), while unlabeled data is abundant. In the context of graph learning, semi-supervised approaches have proven effective for tasks such as node classification (Kipf and Welling, 2017), where only a few nodes are annotated and the label information must be propagated through the network structure.

1.3 Deep Learning

Deep learning encompasses a family of machine learning methods based on the composition of multiple layers of artificial neurons, capable of modeling highly complex and non-linear relationships within high-dimensional data. In this section, we present the fundamental concepts and terminology necessary for understanding the remainder of this thesis, with a particular emphasis on principles relevant to deep learning: we first introduce the basic building blocks underlying most deep architectures, then detail convolutional neural networks (CNNs) and autoencoders, illustrating their principles and applications.

1.3.1 Fundamental Architecture

We present the core architecture, covering feedforward neural networks, activation functions, the concepts of overfitting and underfitting, and regularization techniques.

A *feedforward neural network* (FFN) is a foundational architecture in deep learning that implements a sequence of affine transformations interleaved with pointwise nonlinearities (Rosenblatt, 1958; Rumelhart et al., 1986)

$$\mathbf{a}^{(0)} = \mathbf{x}, \quad \mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\,\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}, \quad \mathbf{a}^{(\ell)} = \phi\big(\mathbf{z}^{(\ell)}\big), \tag{1.10}$$

for layers $\ell = 1, \ldots, L$, where $\mathbf{W}^{(\ell)}$ and $\mathbf{b}^{(\ell)}$ are the weight matrix and bias vector, and ϕ is an activation function. The final output $y = \mathbf{a}^{(L)}$ may be passed through a task-specific transformation (e.g., softmax for classification).

Activation functions introduce the nonlinearity necessary for the network to approximate complex, non-linear mappings. Common choices include the rectified linear unit (ReLU) and its variants (Nair and Hinton, 2010; Maas et al., 2013):

$$\phi_{\text{ReLU}}(z) = \max(0, z), \tag{1.11}$$

$$\phi_{\text{LeakyReLU}}(z) = \max(\lambda z, z), \qquad \lambda \in (0, 1), \tag{1.12}$$

as well as the sigmoid and hyperbolic tangent functions (Bishop, 1995):

$$\phi_\sigma(z) = \frac{1}{1+e^{-z}} \tag{1.13}$$

$$\phi_{\text{tanh}}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{1.14}$$

For multiclass classification, softmax activation in the output layer converts logits $\mathbf{z}_i$ into a probability distribution (Bishop, 1995):

$$\phi_{\text{softmax}}(\mathbf{z})_i = \frac{e^{\mathbf{z}_i}}{\sum_j e^{\mathbf{z}_j}} \tag{1.15}$$

Overfitting and underfitting represent a fundamental trade-off in machine learning model generalization. The learning process typically involves fitting a model to a *training set* and evaluating its performance on a separate *test set* to estimate real-world performance. Overfitting occurs when a model learns the training data too well, including its noise and statistical fluctuations, resulting in high training performance but poor generalization to the test set. In contrast, underfitting occurs when a model is too simplistic to capture the underlying structure of the data, leading to suboptimal performance on both the training and test sets. These phenomena are formally characterized by the *bias-variance tradeoff* (Hastie et al., 2009), where underfitting corresponds to high bias and overfitting to high variance.

Regularization techniques are used to prevent overfitting and improve generalization. FFNs employ regularization techniques such as dropout and norm penalties. In *dropout* (Srivastava et al., 2014), each hidden activation is multiplied by a binary mask $\boldsymbol{\xi}^{(\ell)} \sim \text{Bernoulli}(1-p)$, where $p \in [0,1]$ is the dropout rate, during training:

$$\tilde{a}^{(\ell)} = \boldsymbol{\xi}^{(\ell)} \odot \phi\big(\mathbf{W}^{(\ell)}\,\tilde{\mathbf{a}}^{(\ell-1)} + \mathbf{b}^{(\ell)}\big), \tag{1.16}$$

where $\odot$ denotes element-wise multiplication. At test time, when all neurons are active, activations are scaled by the retention probability $1-p$ to maintain the expected magnitude of activations:

$$\mathbf{a}^{(\ell)} = (1-p) \cdot \phi\big(\mathbf{W}^{(\ell)}\,\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}\big). \tag{1.17}$$

Alternative implementations may use *inverted dropout*, where scaling is applied during training instead.

Norm penalties (weight decay) add a term to the loss that penalizes large weights. For example, with an L_2 penalty the objective becomes

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \sum_{\ell=1}^{L} \|\mathbf{W}^{(\ell)}\|_F^2, \tag{1.18}$$

where $\mathcal{L}_{\text{data}}$ is the data-fitting term and $\lambda > 0$ controls the strength of regularization (Krogh and Hertz, 1992). Similarly, L_1 penalties encourage sparsity in the weight matrices.

1.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a widely adopted deep-learning architecture designed to exploit the local spatial or temporal dependencies present in grid-structured data, such as images, audio signals, or text sequences (LeCun et al., 1998). By applying learnable convolutional filters, CNNs capture local features while sharing weights across the input, which dramatically reduces the total number of parameters compared to fully connected networks. These models naturally extend to one-dimensional (e.g., time series or text) and two-dimensional (e.g., images) inputs. A typical CNN consists of alternating convolutional layers, which extract hierarchical patterns through learned kernels, and pooling layers, which downsample feature maps to achieve translation invariance and further parameter efficiency. Together, these components enable CNNs to learn rich, spatially aware representations with relatively low computational cost.

Convolutional layers operate by applying learnable filters to small, spatially contiguous regions of the input. For example, in an image the first convolutional layer typically detects simple edge patterns, horizontal, vertical or diagonal, by convolving the image with a set of small kernels (e.g. 3×3 or 5×5 filters). Subsequent layers build upon these low-level features to recognize increasingly complex structures: the second layer may combine edge detectors into motifs such as corners or textures, and deeper layers can assemble these motifs into semantic parts (eyes, mouths, or other object components). Each convolutional layer uses multiple filters of the same spatial size in parallel, producing a corresponding set of feature maps that preserve the two-dimensional topology of the data.

The spatial dimension of the output after each convolutional layer is determined by Equation 1.19.

$$O = \left\lfloor \frac{W - f + 2P}{S} \right\rfloor + 1, \tag{1.19}$$

here, W is the size (height or width) of the input feature map, f is the size of the convolutional kernel, P is the amount of zero-padding applied to the input, and S is the stride with which the kernel is moved. The total number of output channels is equal to the number of filters applied in that layer.

Pooling layers are strategically inserted after convolutional operations to reduce the spatial dimensions of the feature maps. They function by aggregating values within local sliding windows, summarizing the statistics of each receptive region to produce a compact, lower resolution representation. The most common operations are max pooling, which extracts the maximum value to emphasize the most salient features, and average pooling, which computes the mean for smoother downsampling. Variants include L_2-norm pooling, L_p-pooling, stochastic pooling, adaptive pooling, and global pooling. Critically, the pooling is applied independently to each channel, preserving the channel count while compressing the spatial resolution. This downsampling provides key benefits: it reduces the computational cost and memory footprint for subsequent layers, mitigates overfitting by imparting translation invariance and reducing parameters, and increases the receptive field to integrate broader contextual information.

1.3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) were first introduced by Elman (1990) to model sequential data by maintaining a hidden state that evolves over time. At each time step i, an RNN takes an input vector $\mathbf{x}^{(i)}$ and the previous hidden state $\mathbf{h}^{(i-1)}$ to produce a new hidden state $\mathbf{h}^{(i)}$ and an output $\mathbf{y}^{(i)}$, as shown in Equation 1.20 and illustrated by Figure 1.3.

$$\begin{aligned} \mathbf{h}^{(i)} &= f\big(\mathbf{W}_h \mathbf{x}^{(i)} + \mathbf{U}_h \mathbf{h}^{(i-1)} + \mathbf{b}_h\big), \\ \mathbf{y}^{(i)} &= g\big(\mathbf{W}_y \mathbf{h}^{(i)} + \mathbf{b}_y\big) \end{aligned} \tag{1.20}$$

where f and g are non-linear activation functions, and $\mathbf{W}_h$, $\mathbf{U}_h$, and $\mathbf{W}_y$ are learnable weight matrices with biases $\mathbf{b}_h$ and $\mathbf{b}_y$. This recurrence enables the network to integrate information over arbitrary sequence lengths, making RNNs well-suited for tasks such as language modeling, time-series forecasting, and speech recognition.

However, RNNs suffer from vanishing and exploding gradients when learning long-term dependencies (Bengio et al., 1994). To address these issues, the Long Short-Term Memory (LSTM) architecture introduces memory cells along with input, forget, and output gates to regulate information flow and decide when to retain or discard information over extended sequences (Hochreiter and Schmidhuber, 1997). The Gated Recurrent Unit (GRU) simplifies this mechanism by merging the forget and input gates into a single update gate and combining the cell state with the hidden state, thereby reducing computational complexity without sacrificing performance (Cho et al., 2014a).

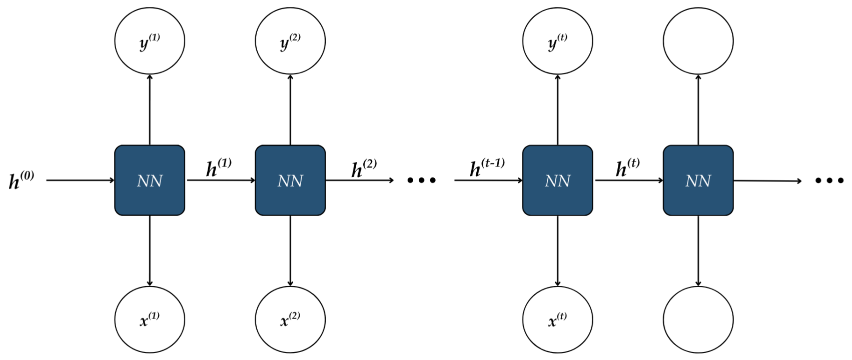


Figure 1.3—Recurrent neural network architecture

1.3.4 Autoencoders

Autoencoders (Kramer, 1991) are unsupervised neural networks that learn compressed representations of unlabeled data through a dual-phase compression-reconstruction process. Their architecture enables applications across diverse data modalities (images, text, graphs) including dimensionality reduction, community detection, anomaly detection, and feature extraction.

The fundamental architecture comprises three core components (Figure 1.4): an encoder that condenses high-dimensional inputs into compact low-dimensional embeddings, a decoder that reconstructs inputs from these embeddings, and a loss function that quantifies reconstruction fidelity.

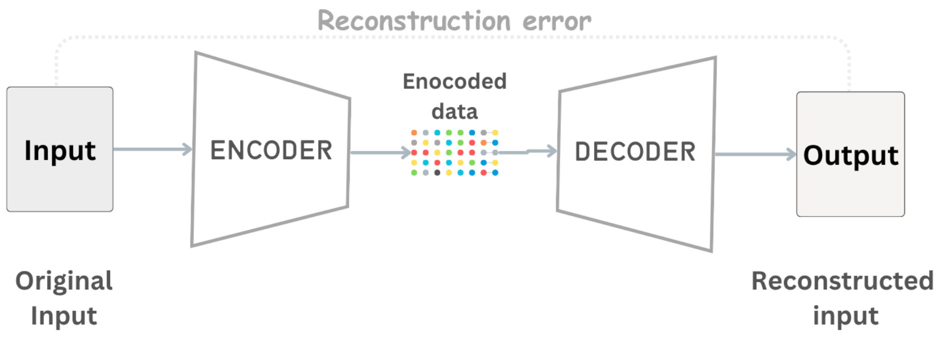


Figure 1.4—End-to-end autoencoder workflow illustrating how input data is compressed into a latent representation through the encoder and then reconstructed by the decoder, enabling unsupervised feature learning and data dimensionality reduction.

To illustrate, consider a photographic input **x** of an automobile. The encoder ϕ_{enc} functions as a feature extractor, distilling salient attributes such as silhouette profile, wheel configuration, and grille geometry into a latent representation **z**, while suppressing irrelevant details. Subsequently, the decoder ϕ_{dec} generates a reconstructed output $\hat{\mathbf{x}}$ exclusively from **z**, without reference to the original input. Reconstruction accuracy is optimized through a loss function $\mathcal{L}$ that penalizes deviations in structural integrity or chromatic properties.

An autoencoder implements the pipeline:

$$\mathbf{x} \in \mathbb{R}^D \xrightarrow{\phi_{\text{enc}}} \mathbf{z} \in \mathbb{R}^d \xrightarrow{\phi_{\text{dec}}} \hat{\mathbf{x}} \in \mathbb{R}^D, \quad d \ll D, \quad d, D \in \mathbb{N}. \tag{1.21}$$

where d/D defines the *compression ratio*. The bottleneck constraint forces information distillation:

$$\mathbf{z} = \phi_{\text{enc}}(\mathbf{x}; \Theta_e) = \sigma(\mathbf{W}_e\mathbf{x} + \mathbf{b}_e), \quad \Theta_e = \{\mathbf{W}_e \in \mathbb{R}^{d\times D}, \mathbf{b}_e \in \mathbb{R}^d\} \tag{1.22}$$

with σ denoting nonlinear activation such as ReLU, sigmoid, etc. The decoder reconstructs the input:

$$\hat{\mathbf{x}} = \phi_{\text{dec}}(\mathbf{z}; \Theta_d) = \psi(\mathbf{W}_d\mathbf{z} + \mathbf{b}_d), \quad \Theta_d = \{\mathbf{W}_d \in \mathbb{R}^{D\times d}, \mathbf{b}_d \in \mathbb{R}^D\} \tag{1.23}$$

where ψ is output activation.

Loss functions and optimization include Mean Squared Error (MSE; Equation 1.24), which is convex, sensitive to outliers, and assumes Gaussian noise; Binary Cross-Entropy (BCE; Equation 1.25), which provides a probabilistic interpretation and is typically paired with a sigmoid output; and Kullback-Leibler (KL) divergence (Equation 1.26), which measures the divergence between two Gaussian distributions P and Q.

$$\mathcal{L}_{\text{MSE}} = \frac{1}{D}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \qquad \text{(Mean Squared Error)} \tag{1.24}$$

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{D}\sum_{i=1}^{D}[x_i \log(\hat{x}_i) + (1 - x_i)\log(1 - \hat{x}_i)] \qquad \text{(Binary Cross-Entropy)} \tag{1.25}$$

$$\mathcal{L}_{\text{KL}(P\|Q)} = \sum_i P(i)\log\left(\frac{P(i)}{Q(i)}\right) \qquad \text{(Kullback-Leibler divergence)} \tag{1.26}$$

The global objective minimizes:

$$\min_{\Theta_e,\Theta_d} \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}) + \lambda\Omega(\Theta) \tag{1.27}$$

where $\Omega(\Theta) = \|\mathbf{W}\|_F^2$ is weight decay regularization, λ controls overfitting, and N is the number of samples. Gradients are computed via chain rule:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_e} = \underbrace{\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{x}}}}_{\text{Reconstruction error}} \cdot \underbrace{\frac{\partial\hat{\mathbf{x}}}{\partial\mathbf{z}}}_{\text{Decoder Jacobian}} \cdot \underbrace{\frac{\partial\mathbf{z}}{\partial\mathbf{W}_e}}_{\text{Encoder Jacobian}} \tag{1.28}$$

Here, $\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{x}}}$ represents the gradient of the loss with respect to the reconstructed input, reflecting the reconstruction error. The term $\frac{\partial\hat{\mathbf{x}}}{\partial\mathbf{z}}$ captures the sensitivity of the decoder output to the latent representation $\mathbf{z}$, while $\frac{\partial\mathbf{z}}{\partial\mathbf{W}_e}$ accounts for how changes in the encoder weights affect the latent code. This formulation outlines the full backpropagation path through the autoencoder architecture.

Variants and Specialized Architectures

The basic autoencoder framework has been extended through various specialized architectures tailored to different learning objectives. Below, we describe key variants of autoencoders, each designed to enhance specific aspects of representation learning.

Sparse Autoencoders: Olshausen and Field (1996) enforce sparsity in neuron activations through population-level constraints such as L1 regularization or KL divergence penalties applied to hidden units. These biologically inspired constraints ensure that only a small subset of neurons is activated for any given input, aligning with the information bottleneck principle observed in neural systems. The resulting representations are highly interpretable and specialized, effectively eliminating redundancy in high-dimensional data. Sparse autoencoders have been successfully applied in unsupervised pretraining for image recognition (Vincent et al., 2008), the discovery of neuroscience-aligned visual features (e.g., orientation-selective filters) (Olshausen and Field, 1996), and dimensionality reduction in genomics by isolating statistically salient latent factors (Eraslan et al., 2019).

Denoising Autoencoders: Vincent et al. (2010) aim to reconstruct clean inputs from artificially corrupted versions (e.g., via Gaussian noise or masking), thereby encouraging the model to learn noise-invariant features. The reconstruction loss between the original input and its denoised version promotes robustness to input perturbations. These models are widely used in tasks such as data imputation for incomplete records (e.g., medical datasets), robust feature extraction in noisy environments (e.g., speech signals), and self-supervised pretraining for downstream tasks like semantic segmentation.

Variational Autoencoders (VAEs): Kingma and Welling (2014) introduce a probabilistic framework wherein latent variables are modeled as Gaussian distributions (Figure 1.5). The encoder outputs both the mean and variance for each latent dimension, enabling stochastic sampling through the reparameterization trick. Training optimizes a joint objective that balances accurate reconstruction with regularization of the latent space toward a predefined prior distribution. VAEs are particularly effective for generative modeling, uncertainty-aware anomaly detection, and disentangled feature learning.

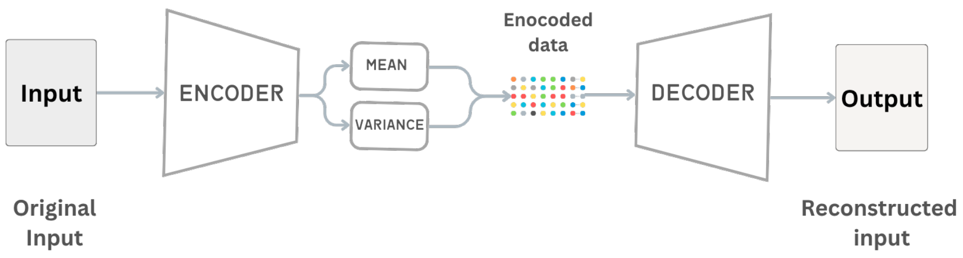


Figure 1.5—Variational Autoencoder (VAE) architecture showing input encoding to latent distribution (mean and variance), sampling, and reconstruction through the decoder.

Convolutional Autoencoders (CAEs): Masci et al. (2011) employ convolutional layers in the encoder and transpose-convolutional (deconvolutional) layers in the decoder to exploit spatial locality and translation invariance. The encoder progressively reduces spatial dimensions via convolutions and pooling operations while extracting hierarchical features. The decoder reverses this process using learned upsampling kernels.

A typical example on the MNIST dataset (Guo et al., 2017) uses three consecutive 3×3 convolutions with ReLU activations and stride 2 (with 32, 64, and 128 filters), reducing the $28\times28\times1$ input to a compact $3\times3\times128$ tensor, which is flattened to a 10-dimensional code. The decoder mirrors this structure using three 3×3 transpose-convolutions (stride 2; filters 128, 64, 32), followed by a 1×1 deconvolution with sigmoid activation to restore the original resolution. Despite the narrow bottleneck, this architecture achieves high-fidelity reconstructions.

Graph Autoencoders: Thomas and Welling (2016) adapt the autoencoder framework to non-Euclidean domains by incorporating GNNs within the encoder and/or the decoder. These models are designed to learn node or graph-level embeddings that preserve the topological structure of the graph. Graph autoencoders are especially valuable when the relationships between entities carry essential information, such as in social networks, citation graphs, or molecular structures. This thesis emphasizes this class of models and explores their use in graph-structured data. A detailed discussion of their architecture and operational mechanisms is presented in the subsequent sections.

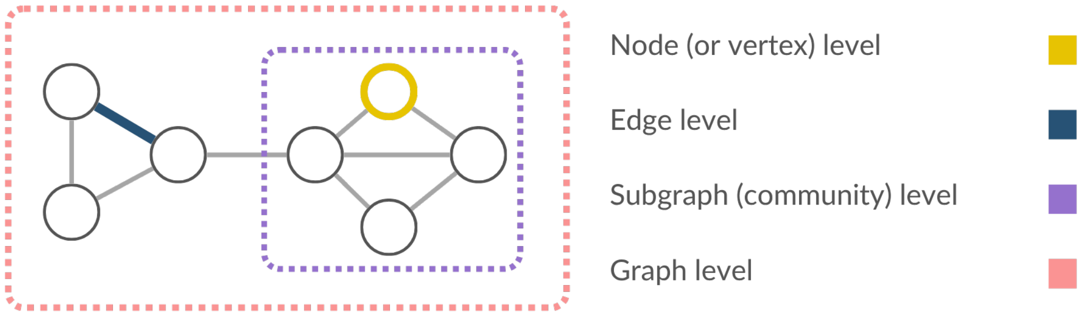


Figure 1.6—Overview of the different levels of learning tasks on graphs, including node-level, edge-level, subgraph (community)-level, and graph-level prediction tasks.

1.4 Machine Learning on Graphs

Graph-structured data naturally emerge in diverse domains, including social networks, biochemical compounds, knowledge graphs, and molecular structures. These data types pose unique challenges for machine learning models due to their non-Euclidean nature, irregular topology, and variable size. Deep learning on graphs aims to build neural architectures that capture and exploit the relational information embedded in such structures. The prediction tasks on graphs can be broadly categorized by their level of granularity: node-level, edge-level, subgraph-level (e.g., community-level), and graph-level tasks (see Figure 1.6).

Node-level models aim to infer attributes or labels for individual nodes in a graph. A prominent example is protein structure prediction, as demonstrated by AlphaFold 3 (Abramson et al., 2024). In this setting, amino acids in a polypeptide chain are represented as nodes, with edges encoding spatial or sequential proximity. The model learns to predict the 3D structure of proteins by propagating information across the graph using attention mechanisms and relational reasoning (see Figure 1.7). The key idea is to ensure that each node embedding reflects both its local biochemical context and global structural constraints.

Edge-level, such as edge classification and link prediction, concentrate on forecasting the existence or specific type of relationship between pairs of nodes (Wu et al., 2021). Link prediction is widely applied in social networks, recommendation systems, and biomedical domains. In recommendation systems, for instance, PinSage (Ying et al., 2018) operates on bipartite user–item graphs where existing edges signify historical interactions. The model learns to estimate the likelihood of unobserved user–item links,

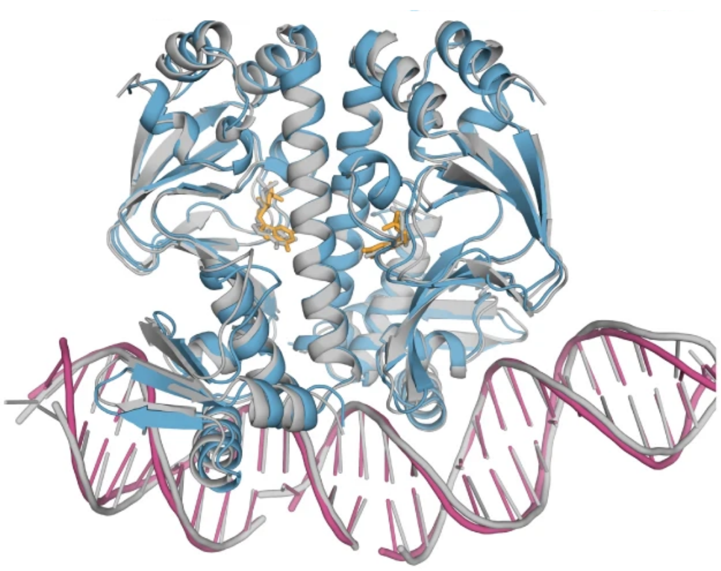


Figure 1.7—AlphaFold predicts a protein's 3D structure from its amino-acid sequence. Example structures predicted using AlphaFold 3 for bacterial proteins (Abramson et al., 2024).

directly facilitating the generation of personalized recommendations. A parallel application is seen in drug–drug interaction prediction, where models like Decagon (Zitnik et al., 2018) represent drugs as nodes and interaction types as distinct edge labels; here, the model's primary function is to predict missing links to uncover potential therapeutic synergies or adverse side effects.

Subgraph-level (Atastina et al., 2017; Ur Rehman et al., 2021) focuses on discovering patterns or assigning labels to substructures within a graph. Community detection (Fortunato, 2010) in social networks exemplifies this task: nodes represent users, and edges indicate interactions. The objective is to identify densely connected groups of nodes that correspond to real-world communities or interest groups.

Graph-level tasks involve predicting properties of the entire graph, such as drug discovery or physics simulation. In drug discovery, for instance, a molecule is modeled as a graph where atoms are nodes and chemical bonds are edges. The goal is to learn a representation of the whole molecule to predict pharmacological properties such as toxicity or binding affinity (Konaklieva and Plotkin, 2023). These tasks typically involve aggregating node and edge information into a global embedding vector through pooling or readout operations.

Another important concept in graph machine learning is the distinction between transductive and inductive learning: transductive learning trains and predicts on the exact nodes or edges observed during training, whereas inductive learning builds a model that can generalize to entirely new, unseen nodes or edges.

In the following sections, we detail two foundational approaches in deep learning on graphs: graph embedding and graph neural networks (GNNs). While graph embedding focuses on learning compact vector representations of nodes, edge or entire graphs, GNNs enable end-to-end training using neighborhood aggregation and message passing.

1.4.1 Graph Embedding

Most real-world networks are complex, and processing and analyzing their data is challenging. Graph embedding methods address this challenge by mapping the network into a low-dimensional space that preserves essential structural and feature information.

Graph embedding encompasses a family of techniques designed to represent graph entities—nodes, edges, or entire subgraphs in a continuous vector space of much lower dimensionality ($d \ll |V|$), while retaining the most salient structural and attribute information. Let $\mathcal{G} = (V, E)$ be a graph with adjacency matrix $A \in \mathbb{R}^{N\times N}$ and optional node-feature matrix $X \in \mathbb{R}^{N\times m}$. A node embedding seeks to ensure that two structurally or semantically similar nodes in the original graph are mapped to nearby points in the latent space. Formally, an encoder

$$\text{ENC} : V \rightarrow \mathbb{R}^d$$

assigns to each node v_i a vector $\mathbf{z}_i = \text{ENC}(v_i)$ such that, for any pair of nodes v_i, v_j, a graph proximity measure $S_{\mathcal{G}}(v_i, v_j)$ is approximated by a similarity function in the embedding space, $S_E(\mathbf{z}_i, \mathbf{z}_j)$ (see Figure 1.8).

1.4.1.1 Classical Graph Embedding Methods

Classical graph embedding methods typically involve two steps: constructing a similarity graph from node features, then applying dimensionality reduction to preserve structural properties (Cai et al., 2018).

Locally Linear Embedding (LLE) (Roweis and Saul, 2000) first computes optimal reconstruction weights for each node based on its k-nearest neighbors. It then learns low-dimensional embeddings that preserve these local reconstruction relationships. While

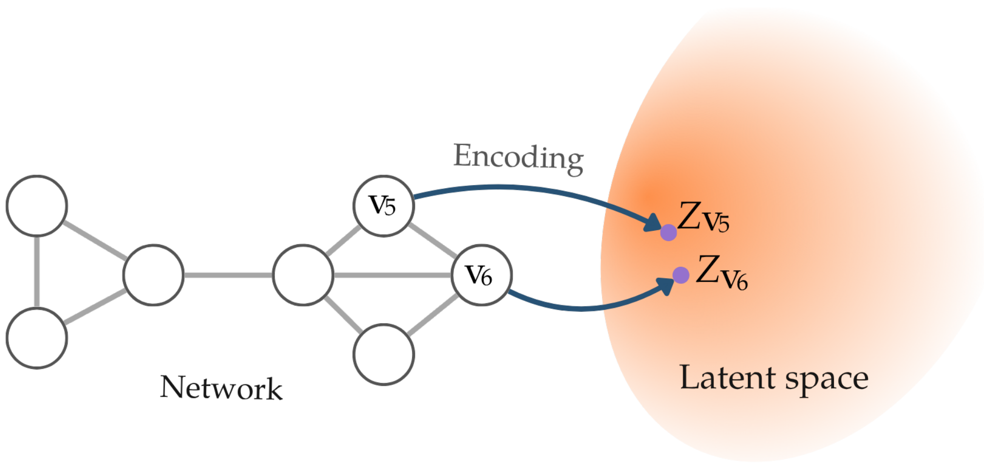


Figure 1.8—Visualization of node embedding where adjacent nodes in the graph are mapped to close points in the latent space, preserving structural similarity.

effective for maintaining local geometry, LLE is sensitive to neighbor selection noise and requires solving an $N \times N$ eigenproblem.

Multidimensional Scaling (***MDS***) (Borg and Groenen, 2007) directly embeds nodes to preserve given pairwise distances. The classical variant uses eigen-decomposition of the double-centered distance matrix, providing an analytic solution for global distance preservation but scaling cubically with node count.

Isometric Mapping (***Isomap***) (Tenenbaum et al., 2000) first estimates geodesic distances via the shortest paths on a k-NN graph, then applies MDS to preserve these manifold distances. This captures global manifold structure effectively, but inherits MDS's cubic complexity and sensitivity to graph connectivity.

Laplacian Eigenmaps (Belkin and Niyogi, 2003) solve a generalized eigenvalue problem using the normalized graph Laplacian. The solution embeds nodes such that adjacent vertices remain close in the latent space, emphasizing local smoothness over global distance preservation.

1.4.1.2 Modern Graph Embedding Methods

Modern graph embedding methods aim to preserve both local or global structural properties of the original graph by modeling various orders of node similarity. First-order proximity ensures that directly connected nodes remain close in the embedding space, while second-order (and higher-order) proximity captures similarity in nodes' neighborhood distributions or more distant relational patterns. In this section, we explore several essential approaches for graph representation.

DeepWalk (Perozzi et al., 2014) is an unsupervised graph-embedding algorithm that learns low-dimensional node representations by combining random walks on the graph with neural-language-modeling techniques. Its objective is to embed each node $v \in V$ into a vector $\Phi(v) \in \mathbb{R}^d$ (with $d \ll |V|$) such that nodes that are structurally similar in the original graph remain close in the embedding space.

DeepWalk proceeds in two main phases. First, it treats truncated random walks on the graph $\mathcal{G}$ as sentences and nodes as words, thereby transforming the graph into a corpus of node-sequences. In this Random-Walk Sequence Generation process, for each node v, *DeepWalk* generates T independent random walks $W = (v_0, v_1, \ldots, v_t)$ of fixed length t. Each walk begins at $v_0 = v$ and proceeds by moving at each step to a uniformly sampled neighbor. These walks capture both first-order (direct) and higher-order (multi-step) proximity patterns among nodes.

In the second phase, *DeepWalk* learns continuous vector representations for each node by training a language model on the generated walks. It employs the two classic *Word2Vec* architectures: *Skip-Gram* and *Continuous Bag-of-Words* (*CBOW*). These models aim to preserve the information of the graph structure by capturing the co-occurrence relations between nodes in the graph, thereby maximizing the likelihood of node co-occurrences within a sliding window of size w. Consequently, these models capture the relational and semantic information conveyed by node neighborhoods.

The ***Skip-Gram*** (Mikolov et al., 2013b) variant aims, for each target node v_i , to predict its surrounding context nodes within the window. Formally, if we generate a corpus of node sequences of total length T and use a context window of size w, the objective is to maximize the log-likelihood of observing context nodes given each center node:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-w\le j\le w,\ j\neq 0}\log p(v_{t+j}|\Phi(v_t)) \tag{1.29}$$

A simple neural network with one hidden layer computes this probability. Denoting the one-hot encoding of v_i by $\mathbf{x} \in \{0,1\}^{|V|}$, we have the following:

$$\mathbf{h} = \mathbf{W}^{\top}\mathbf{x} \in \mathbb{R}^d, \quad \mathbf{u} = \mathbf{W}'^{\top}\mathbf{h} \in \mathbb{R}^{|V|}, \quad P(u \mid v_i) = \frac{\exp(u_u)}{\sum_{x\in V}\exp(u_x)} \tag{1.30}$$

where $\mathbf{W}, \mathbf{W}'$ are learned weight matrices and u_u denotes the component of $\mathbf{u}$ corresponding to node u. For example, after training, inputting a node associated with the word 'climate' might produce a high probability for the node 'weather', illustrating that the model has captured meaningful semantic relationships between them.

To reduce the $O(|V|)$ cost of the full softmax in Equation (1.30), a Skip-gram variant employs hierarchical softmax (Scarselli et al., 2009), organizing nodes into a binary Huffman tree (Salakhutdinov and Hinton, 2009), which reduces the per-training-example complexity to $O(\log |V|)$ (Perozzi et al., 2014).

The ***CBOW*** variant reverses the *Skip-Gram* objective by predicting each target node from the average embedding of its context nodes. For a random walk $W = (v_0, \ldots, v_t)$ and a window size w, the context vector $\mathbf{c}_i$ for a target node v_i is defined as:

$$\mathbf{c}_i = \frac{1}{2w} \sum_{\substack{j=-w \\ j\neq 0}}^{w} \Phi(v_{i+j}) \in \mathbb{R}^d. \tag{1.31}$$

The CBOW objective is to maximize the log-probability of the target node given its context:

$$\max_{\Phi} \sum_{i=0}^{t} \log P\big(v_i \mid \mathbf{c}_i\big), \tag{1.32}$$

This is typically implemented using the same hierarchical softmax or negative sampling techniques as Skip-Gram. Architecturally, CBOW mirrors Skip-Gram but uses the averaged context vector $\mathbf{c}_i$ as its input instead of a single one-hot vector.

Figure 1.9 presents the Skip-Gram and CBOW architectures side by side, showing how they invert the direction of prediction while relying on the same neural-language-model framework. In the sentence "the city is beautiful today" with window size $w = 2$, Skip-Gram treats each central word as input and learns to predict its neighbors. For example, "city" predicts "the," "is," and "beautiful," and "is" predicts "the," "city," "beautiful," and "today," as detailed in Table 1.1. By contrast, CBOW combines the embeddings of the surrounding words, such as "the," "is," "beautiful," and "today" when the target is "city" and learns to predict the missing central word.

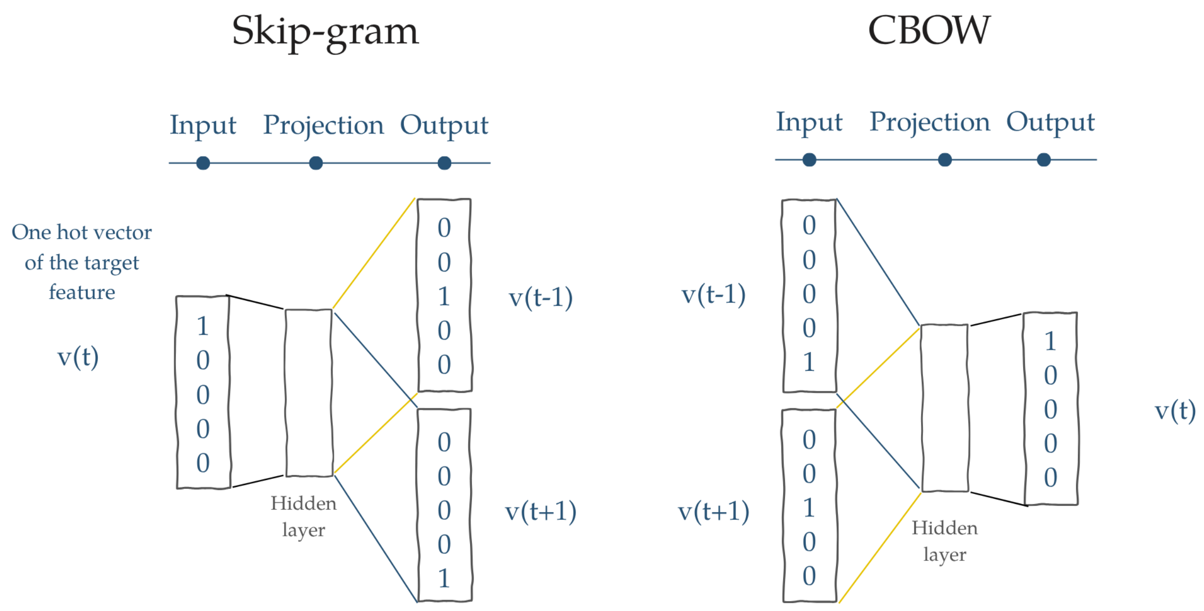


Figure 1.9—Comparison of the Skip-Gram and CBOW models used in Word2Vec: Skip-Gram predicts surrounding context nodes given a central target node, whereas CBOW predicts the target node from its surrounding context (Mikolov et al., 2013a).

Table 1.1—Example training pairs for Skip-Gram and CBOW on the sentence "the city is beautiful today" with a window size $w = 2$.

Model	**Sentence**	**Training Pairs**
Skip-Gram	the city is beautiful today	(city, the), (city, is), (city, beautiful)
	the city is beautiful today	(is, the), (is, city), (is, beautiful), (is, today)
CBOW	the city is beautiful today	([the, is, beautiful], city)
	the city is beautiful today	([the, city, beautiful, today], is)

Node2Vec is a deep-learning algorithm introduced by researchers at Stanford University (Grover and Leskovec, 2016) that, like DeepWalk, aims to learn low-dimensional representations of graph nodes. It advances the first phase of DeepWalk by employing a second-order random walk that incorporates information about the previous step rather than relying solely on a first-order walk. To generate neighborhood samples for each node in V, Node2Vec interpolates between BFS and DFS strategies via two user-defined parameters, p and q. At each step of the walk, the transition probability from the current node v to a neighbor x is given by Equation 1.33.

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \dfrac{\pi_{vx}}{Z} & \text{if } (v, x) \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{1.33}$$

where π_{vx} is an unnormalized transition weight, and Z is a normalization constant. In an unweighted graph, one typically sets $\pi_{vx} = w_{vx} = 1$.

To bias the walk according to the previous node t , Node2Vec defines the transition probabilities as follows:

$$\pi_{vx} \;=\; \alpha_{p,q}(t, x) \;\cdot w_{vx},$$

where the bias factor $\alpha_{p,q}(t, x)$ depends on the shortest-path distance d_{tx} between the previous node t and the prospective next node x. Specifically,

$$\alpha_{p,q}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0, \\ 1 & \text{if } d_{tx} = 1, \\ \frac{1}{q} & \text{if } d_{tx} = 2. \end{cases}$$

When $d_{tx} = 0$, the walk returns to the immediately preceding node with a probability proportional to $1/p$. When $d_{tx} = 1$, it moves to a neighbor of v that is adjacent to t; when $d_{tx} = 2$, it explores nodes two hops away from t, with that probability scaled by $1/q$. By adjusting p and q, practitioners can control whether the sampling favors local neighborhoods (small q), emphasizing community structure or broader exploration (large q) capturing more global roles. For example, setting $p = 1$ and $q = 2$, this choice biases the walk toward nodes closer to the current node, enhancing local structural learning, while $p = 1$ and $q = 0.5$ encourage exploration of more distant nodes, and thus better capture global community patterns. The resulting biased random walks are then fed into a Word2Vec-style optimization, as in DeepWalk, to produce continuous vector embeddings that reflect both local and global graph topology. As illustrated in Figure 1.10, this biasing mechanism effectively balances exploration and exploitation during the sampling process.

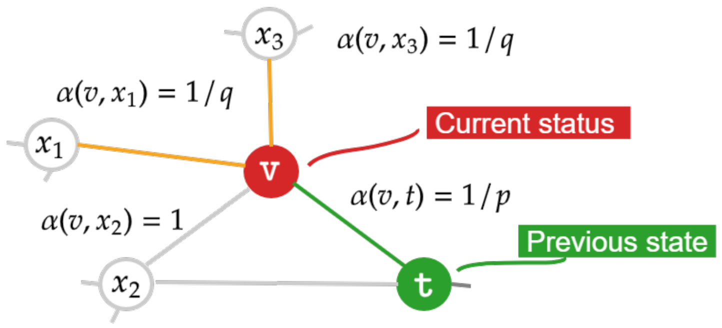


Figure 1.10—Illustration of the Node2Vec random walk strategy controlled by the parameter α, which guides the exploration trade-off between local and global neighborhoods (Grover and Leskovec, 2016).

Building on DeepWalk's equivalence to matrix factorization, ***Text-Associated DeepWalk*** (***TADW***) (Yang et al., 2015) extends this paradigm by jointly factorizing a random walk-derived node co-occurrence matrix M. Its objective function is formalized as:

$$min_{W,H} \, \|M - W^{\top}H\|_F^2 \; + \; \tfrac{\lambda}{2}\|X - H\|_F^2 \, . \tag{1.34}$$

Crucially, rather than relying solely on the graph's structural information encoded by its adjacency matrix A, TADW innovatively infuses each node's latent representation (captured in H) with rich semantic content. This semantic information is extracted from the nodes' associated textual data via a pre-processed text–feature matrix X. The optimization of this formulation is efficiently achieved through an iterative process of alternating least squares and truncated Singular Value Decomposition (SVD). This integrated approach not only meticulously preserves the graph's topological properties but also effectively encodes nuanced node semantics, leading to notable performance gains.

Large-scale Information Network Embedding (***LINE***) (Tang et al., 2015) stands as a prominent probabilistic edge-reconstruction model engineered to derive continuous vector representations from large graphs. Its design innovatively captures both local and global network structures by simultaneously preserving two distinct notions of proximity. Furthermore, LINE exhibits remarkable versatility, accommodating directed or undirected, and weighted or unweighted graphs. In contrast to random-walk-based methodologies like DeepWalk and Node2Vec, which implicitly sample node sequences, LINE adopts an explicit approach: it directly constructs pairs of adjacent nodes (representing one-hop relationships) and optimizes two separate, yet complementary, objectives.

The **first-order proximity** mechanism models the joint probability of observing an undirected edge (i, j) with an associated weight w_{ij} as shown in Equation 1.35.

$$p_1(i, j) \;=\; \sigma\big(\mathbf{z}_i^\top \mathbf{z}_j\big), \tag{1.35}$$

where $\sigma(x)$ is the sigmoid function. The core of this objective is to minimize the Kullback-Leibler divergence between the empirically observed edge-weight distribution and the model's predicted distribution. This rigorous optimization process ensures that strongly connected node pairs are ultimately mapped to proximate positions in the embedding space.

In contrast, **second-order proximity** is designed to capture the conditional probability of a context node j given a source node i. This objective treats each node i as a "source" and its neighbors j as "contexts" and is maximized by the following formulation:

$$\mathcal{L}_2 = \sum_i \sum_j w_{ij} \log \frac{\exp\big(\mathbf{z}_j'^\top \mathbf{z}_i\big)}{\sum_k \exp\big(\mathbf{z}_k'^\top \mathbf{z}_i\big)}, \tag{1.36}$$

Here, $\mathbf{z}_i$ and $\mathbf{z}'_j$ denote distinct vector representations, specifically assigned when a node acts as a source and a context, respectively. This particular objective function requires nodes that share similar structural neighborhoods to converge toward proximity within the embedding space.

Following independent optimization of these two objectives, typically achieved by stochastic gradient descent, often enhanced with negative sampling for accelerated convergence, LINE consolidates the learned representations. It does this by concatenating the derived first-order and second-order embeddings for each node, yielding a comprehensive final embedding that encapsulates both direct connectivity and broader contextual similarities.

Metapath2Vec (Dong et al., 2017) extends the embedding of random walk to heterogeneous information networks by guiding walks along predefined metapaths, sequences of node types that capture schema-level semantics. For example, in a bibliographic network, an author-paper-venue metapath highlights coauthorship patterns and publishing venues. Metapath2Vec first generates node sequences strictly following the chosen metapath pattern, ensuring that each step transitions to the correct node type. These context-preserving walks are then fed into a heterogeneous Skip-gram model, which learns type-aware embeddings by maximizing the probability of observing correct meta-type neighbors. The result is a set of embeddings that reflect both structural proximity and semantic relationships within a network.

Struc2vec (Ribeiro et al., 2017) embeds nodes by capturing *structural equivalence* rather than mere proximity. In contrast to methods such as DeepWalk or Node2vec, which assume that nodes sharing neighbors are similar, Struc2vec formalizes structural equivalence through the principle that nodes v_1 and v_2 share equivalent roles if their extended neighborhoods exhibit isomorphic degree distributions at multiple scales. Specifically, v_1 and v_2 are structurally equivalent when for every hop distance k, the multisets of k-hop neighbors $R_k(v_1)$ and $R_k(v_2)$ possess similar degree distributions. To operationalize this, Struc2vec constructs a multilayer auxiliary graph where each layer k corresponds to a hierarchy of structural comparisons. The meta-distance $f_k(v_1, v_2)$ between nodes is defined recursively as:

$$f_{-1}(v_1, v_2) = 0, \qquad f_k(v_1, v_2) = f_{k-1}(v_1, v_2) + \mathcal{D}\big(s(R_k(v_1)),\, s(R_k(v_2))\big) \quad (k \geq 0) \quad (1.37)$$

Here, $s(R_k(v))$ denotes the sorted degree sequence of the k-hop neighborhood and $\mathcal{D}(s_1, s_2) = \text{DTW}(s_1, s_2)$ computes the hierarchical dynamic time warping distance between sequences, measuring minimal-cost alignment under sequence warping constraints.

These meta-distances induce edge weights in layer k:

$$w_k(v_1, v_2) = \exp\big(-f_k(v_1, v_2)\big) \quad (1.38)$$

with vertical edges connecting each node's instances across adjacent layers. The exponentially decaying weights ensure random walks preferentially transition between structurally isomorphic nodes. Random walks of length ℓ, repeated γ times per node, move within layer k to neighbor x with probability proportional to $W_k(v, x)$, or shift up/down a layer with probability $1-p$. Finally, the collection of walks trains a Skip-Gram model (window size ω, embedding dimension d, negative sampling) exactly as in DeepWalk, yielding embeddings that reflect structural roles rather than mere adjacency.

Other Methods such as ***ComE***(Cavallari et al., 2017) introduces community-aware embeddings by jointly learning node vectors and probabilistic community membership distributions. Unlike traditional methods representing nodes with single points, ComE models overlapping community structure by capturing affiliations across multiple communities. Training alternates between inferring community assignments via variational inference optimizing categorical distributions over k communities per node and updating embeddings via a Skip-gram objective augmented with community regularization.

Community2vec(Martin, 2017) refines community-aware representation by incorporating hierarchy. Building on first- and second-order proximities, it constructs a community hierarchy via modularity-based detection. Random walks alternate between node-level and community-level transitions: a walker may traverse within a community to capture local structure or jump to parent/child communities to encode broader context. A unified Skip-gram framework processes these mixed sequences, producing embeddings that reflect intra-community cohesion and inter-community relationships, enhancing performance in community detection, link prediction, and node classification.

Each method balances expressiveness, scalability, and preservation of distinct graph properties, enabling practitioners to select embeddings suited for tasks ranging from link prediction to graph classification.

1.4.2 Graph Neural Networks (GNNs)

Since the advent of deep learning, remarkable progress has been made in Euclidean data, such as images in computer vision and sequences in natural language processing, by exploiting their grid-like structure. However, many real-world datasets naturally possess non-Euclidean structures best represented as graphs or manifolds, and classical deep neural networks do not readily generalize to such irregular domains. Researchers have identified several key challenges when applying deep learning to graph-structured data: the irregular topology of graphs defies standard operations like convolution and pooling; graphs vary widely in size, connectivity, and node/edge attributes, often requiring bespoke architectures; the scale of modern graphs, such as social networks with billions of nodes demands highly efficient, scalable algorithms; and diverse application domains (from molecular biology and chemistry to 3D modeling) call for domain-informed feature-extraction strategies. Addressing these challenges gave rise to the field of Graph Neural Networks (GNNs), which were first formalized by Gori et al. (2005) and Scarselli et al. (2009) and are now studied broadly under the umbrella of geometric deep learning.

Graph Neural Networks (GNNs) demonstrate strong capabilities in processing graph-structured data by effectively extracting and learning non-linear feature representations. They enable the projection of complex networks into lower-dimensional spaces while preserving the structural and semantic properties of the original graph. This powerful paradigm has been applied to a wide range of tasks, including community detection, anomaly detection, recommender systems, natural language processing, drug discovery, signal and image processing, and bioinformatics (Wu et al., 2022).

Graph Neural Networks (GNNs) update node representations iteratively at each layer. In this general framework, the embedding of a node is refined by two fundamental functions: **aggregation**, which gathers information from its neighbors, and **combination**, which merges this aggregated information with the node's own previous representation (Xu et al., 2019). This process transforms raw graph-structured data into structure-aware latent embeddings that capture the relational dependencies inherent in non-Euclidean systems.

Formally, let $h_v^{(k-1)}$ be the embedding of node v at layer $k-1$. The update at layer k proceeds in two steps. First, the aggregation step gathers representations from each neighbor $u \in \mathcal{N}(v)$:

$$a_v^{(k)} = \mathrm{AGGREGATE}_k\big(\{\, h_u^{(k-1)} : u \in \mathcal{N}(v)\}\big) \tag{1.39}$$

Second, the combination step merges the aggregated information with the node's previous state:

$$h_v^{(k)} = \mathrm{COMBINE}_k\big(h_v^{(k-1)},\, a_v^{(k)}\big) \tag{1.40}$$

In the next parts, we review the most widely used GNN architectures.

1.4.3 GNN Architectures

In this section, we review the principal families of Graph Neural Network architectures, focusing on their defining characteristics and representative models.

1.4.3.1 Graph Convolutional Networks (GCNs)

Graph convolutional networks (Kipf and Welling, 2017) extend the convolution operation from regular grids to graph-structured data. They are among the most widely used architectures, owing to their simplicity and strong performance across a variety of tasks. Most GCN variants can be classified into two families: spectral methods and spatial methods (Wu et al., 2021). Spectral GCNs are defined in the spectral domain based on the graph Fourier transform, whereas spatial GCNs operate in the vertex domain by aggregating features from local neighborhoods.

Spectral GCNs: Spectral methods construct graph convolutions by analogy with the classical Fourier transform. The graph Fourier transform decomposes an input signal into orthogonal basis functions associated with the eigenvectors of the graph Laplacian. Given an undirected graph with adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$ (where $D_{ii} = \sum_j A_{ij}$), one can define the combinatorial Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ and its symmetric normalization $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\,\mathbf{D}^{-1/2}$.

Bruna et al. (2014) first introduced spectral graph convolutions via the eigendecomposition $\mathbf{L}_{\text{sym}} = \mathbf{U}\mathbf{U}^\top$, which yields an orthonormal basis U of graph Fourier modes and a diagonal matrix of eigenvalues. A learnable spectral filter g_θ acting on a graph signal $\mathbf{x} \in \mathbb{R}^N$ is defined by:

$$g_\theta * \mathbf{x} = \mathbf{U} g_\theta() \mathbf{U}^\top \mathbf{x}, \tag{1.41}$$

where $g_\theta() = \text{diag}(\theta)$ specifies trainable spectral coefficients. Because full eigendecomposition costs $\mathcal{O}(n^3)$, Defferrard et al. (2016) approximate $g_\theta(\Lambda)$ using Chebyshev polynomials in the rescaled Laplacian:

$$\widetilde{\mathbf{L}} = \frac{2}{\lambda_{\text{max}}} \mathbf{L}_{\text{sym}} - \mathbf{I}, \tag{1.42}$$

yielding fast, localized convolutions without explicit spectral decomposition. Kipf and Welling's first-order approximation further simplifies the filter to

$$g_\theta * \mathbf{x} \approx \theta\big(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\big)\mathbf{x}, \tag{1.43}$$

which gives rise to the standard GCN layer (Kipf and Welling, 2017).

Spatial GCNs Spatial methods define convolution directly in the vertex domain by aggregating and combining feature messages from each node's local neighborhood. Let $\mathbf{H}^{(l-1)} \in \mathbb{R}^{n\times d}$ denote the matrix of node embeddings from layer $l-1$. At layer l, the update for node i is given by the following formula:

$$\mathbf{H}_i^{(l)} = \sigma\Bigg(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{\widetilde{A}_{ij}}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} \mathbf{H}_j^{(l-1)} \mathbf{W}^{(l)}\Bigg), \tag{1.44}$$

as in Equation 1.44, where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, one can alternatively separate neighbor and self contributions:

$$\mathbf{H}_i^{(l)} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \frac{A_{ij}}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} \mathbf{H}_j^{(l-1)}\mathbf{W}^{(l)} + \frac{1}{\widetilde{D}_{ii}} \mathbf{H}_i^{(l-1)}\mathbf{W}^{(l)} \Big), \tag{1.45}$$

highlighting the distinct roles of neighbor and self-loop messages (Equation 1.45). Here, σ is a nonlinear activation function and $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times d'}$ is a learnable weight matrix shared across all nodes.

A concrete instance of this framework is Kipf and Welling's GCN (Kipf and Welling, 2017), which employs the symmetric normalized aggregation and combine step in matrix form:

$$\mathbf{H}^{(l)} = \sigma\big(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\big), \tag{1.46}$$

sharing parameters across edges and scaling efficiently to large graphs via sparse matrix multiplication.

1.4.3.2 Graph Attention Networks (GATs)

Graph Attention Networks (GATs) (Veličković et al., 2018) extend GCNs by incorporating a learnable attention mechanism (Vaswani et al., 2017) that dynamically weights neighbor contributions based on feature relevance. Widely adopted in areas such as natural language processing and computer vision, this mechanism enables each node to focus on the most informative neighbors. In contrast to GCNs, which treat all neighbors equally.

GATs incorporate the self-attention mechanism into graph learning by enabling each node to selectively aggregate information from its neighbors based on learned importance weights. This is achieved by computing pairwise attention scores between a node and its immediate neighbors. The unnormalized attention score between nodes i and j in the layer l is computed as in Equation 1.47, where feature vectors from the previous layer are first linearly projected into a latent space using a shared weight matrix $\mathbf{W}^l$ and then passed through a shared attention mechanism a^l followed by a LeakyReLU non-linearity. These scores are then normalized using the *softmax* function, as shown in Equation 1.48, to obtain attention coefficients $\alpha_{ij}^{h,l}$ that sum to one over the neighbors of node i.

$$e_{ij}^{l} = \text{LeakyReLU}\left(a^{(l)} \left(\mathbf{W}^{(l)}\mathbf{z}_i^{(l-1)}, \mathbf{W}^{(l)}\mathbf{z}_j^{(l-1)} \right) \right) \tag{1.47}$$

$$\alpha_{ij}^{l} = \frac{\exp(e_{ij}^{l})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik}^{l})} \tag{1.48}$$

where $\mathcal{N}_i$ denotes the set of neighbors of node i. These coefficients serve as adaptive weights that quantify the relative importance of each neighbor during the aggregation process.

The updated representation of node i is then obtained by computing the weighted sum of the transformed features of its neighbors, followed by a non-linear activation function, as shown in Equation 1.49.

$$\mathbf{z}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{l} \mathbf{W}^{(l)} \mathbf{z}_j^{(l-1)}\right), \tag{1.49}$$

where σ is a non-linear activation function.

Multi-head attention is employed to jointly capture diverse interaction patterns and to project node features into multiple latent subspaces. Each attention head independently learns a distinct similarity function over a node and its neighbors, enabling the model to attend to different aspects of the local structure (Veličković et al., 2018). The individual outputs from each head are then concatenated to form the final node representation, as defined in Equation 1.50.

$$\mathbf{z}_i^{(l)} = \Big\Vert_{h=1}^{H} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{h,l} \mathbf{W}^{h,l} \mathbf{z}_j^{(l-1)}\right), \tag{1.50}$$

where $\|$ denotes concatenation over the H attention heads, $\alpha_{ij}^{h,l}$ is the attention coefficient computed by the h-th head at layer l, and $\mathbf{W}^{h,l}$ is the corresponding learnable weight matrix.

While concatenation is commonly used to combine the outputs of multiple attention heads, an alternative approach at the final layer is to apply a **pooling operation** by averaging the representations from each head (Veličković et al., 2018). This strategy reduces the dimensionality while retaining the aggregated information across heads. The corresponding formulation is given in Equation 1.51:

$$\mathbf{z}_i^{(l)} = \sigma\left(\frac{1}{H} \sum_{h=1}^{H} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{h,l} \mathbf{W}^{h,l} \mathbf{z}_j^{(l-1)}\right), \tag{1.51}$$

where all terms follow the same definitions as in Equation 1.50, and the final node representation is obtained through average pooling across the H attention heads.

1.4.3.3 GraphSAGE

Hamilton et al. (2017) introduce GraphSAGE (Graph SAmple and aggreGatE), an inductive graph convolutional framework designed for large-scale networks. Unlike transductive GCNs, GraphSAGE generalizes to unseen nodes by learning how to aggregate feature information from a fixed-size, sampled neighborhood. At each hop—denoted by the superscript "hop" for every node $v \in V$, a set of neighbors $\mathcal{N}(v)$ is sampled, and their representations from the previous hop are aggregated into a single vector. This aggregated neighborhood vector is concatenated with the node's own representation from the previous hop and transformed via a shared weight matrix $\mathbf{W}^{(\text{hop})}$ followed by a nonlinear activation ϕ, as in Equation 1.52. Figure 1.11 provides a schematic overview of GraphSAGE.

$$\mathbf{h}_v^{(\text{hop})} = \phi\Big(\big[\mathbf{h}_v^{(\text{hop}-1)} \,\Big\|\, \text{AGGREGATE}\big(\{\,\mathbf{h}_u^{(\text{hop}-1)} : u \in \mathcal{N}(v)\}\big)\big]\,\mathbf{W}^{(\text{hop})}\Big), \tag{1.52}$$

where $\|$ denotes concatenation.

GraphSAGE offers several choices for the AGGREGATE function. In the mean aggregator, one computes the element-wise average of the sampled neighbor representations, yielding an inductive analogue of a spectral GCN, as shown in Equation 1.53. The LSTM aggregator treats the sampled neighbors as a sequence and feeds their representations through an LSTM network. The final hidden state of this LSTM serves as the aggregated vector, thereby capturing ordering-dependent interactions among neighbors (Equation 1.54).

$$\mathbf{h}_v^{(\text{hop})} = \phi\Bigg(\big[\mathbf{h}_v^{(\text{hop}-1)} \,\|\, \tfrac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(\text{hop}-1)}\big]\,\mathbf{W}^{(\text{hop})}\Bigg), \tag{1.53}$$

$$\mathbf{h}_v^{(\text{hop})} = \phi\Bigg(\big[\mathbf{h}_v^{(\text{hop}-1)} \,\|\, \text{LSTM}\big(\mathbf{h}_{u_1}^{(\text{hop}-1)}, \ldots, \mathbf{h}_{u_m}^{(\text{hop}-1)}\big)\big]\,\mathbf{W}^{(\text{hop})}\Bigg). \tag{1.54}$$

Finally, the pooling aggregator first applies a nonlinear transformation to each neighbor representation and then performs element-wise max-pooling over the set, as in Equation 1.55.

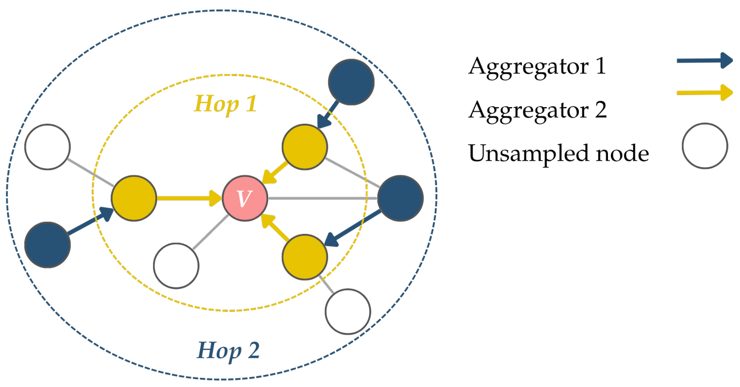


Figure 1.11—Illustration of the GraphSAGE idea: sampling neighbors, aggregating their representations, and updating node embeddings (Hamilton et al., 2017).

$$\mathbf{h}_v^{(\text{hop})} = \phi\Big(\big[\mathbf{h}_v^{(\text{hop}-1)} \,\|\, \max_{u \in \mathcal{N}(v)} \phi\big(\mathbf{h}_u^{(\text{hop}-1)} \mathbf{W}_{\text{pool}}^{(\text{hop})} + \mathbf{b}\big)\big]\Big). \tag{1.55}$$

GraphSAGE can be trained in a supervised fashion, e.g., by minimizing cross-entropy loss for node classification or in an unsupervised regime, where embeddings are optimized with a contrastive objective that encourages similarity between sampled neighbor pairs and discourages similarity with random negative pairs:

$$\mathcal{L} = - \sum_{(u,v) \in \mathcal{P}} \left[\log \sigma\left(\mathbf{z}_u^\top \mathbf{z}_v\right) + Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log \sigma\left(-\mathbf{z}_u^\top \mathbf{z}_{v_n}\right)\right], \tag{1.56}$$

where, the summation $\sum_{(u,v)\in\mathcal{P}}$ is over all positive pairs, where a pair consists of a node u and its neighbor v. The vectors $\mathbf{z}_u$ and $\mathbf{z}_v$ are the learned embeddings for nodes u and v. The sigmoid function, σ, maps the dot product to a similarity score between 0 and 1. The term Q is the negative sampling ratio, which controls how many non-neighbor nodes are sampled. The expression $\mathbb{E}_{v_n \sim P_n(v)}$ denotes the expectation over a random negative sample v_n drawn from a distribution $P_n(v)$.

1.4.3.4 Message Passing Neural Networks (MPNN)

The Message Passing Neural Network (MPNN) framework provides a general formulation for GNNs based on iterative message and update functions (Gilmer et al., 2017). In this paradigm, each iteration t consists of two core steps. First, every node v aggregates messages from its topological neighborhood $\mathcal{N}(v)$ using a learnable message function M_t:

$$\mathbf{m}_v^{(t)} = \sum_{u \in \mathcal{N}(v)} M_t\big(\mathbf{h}_v^{(t-1)}, \mathbf{h}_u^{(t-1)}, \mathbf{e}_{uv}\big), \tag{1.57}$$

where $\mathbf{e}_{uv}$ denotes optional edge features. The node's state is then refreshed by an update function U_t, which integrates the aggregated message $\mathbf{m}_v^{(t)}$ with the node's previous state:

$$\mathbf{h}_v^{(t)} = U_t\big(\mathbf{h}_v^{(t-1)}, \mathbf{m}_v^{(t)}\big). \tag{1.58}$$

After T such iterations, a readout function R pools the final set of node embeddings $\{\mathbf{h}_v^{(T)} : v \in V\}$ to produce a graph-level representation for downstream prediction tasks:

$$\hat{y} = R\big(\mathbf{h}_v^{(T)} : v \in V\big). \tag{1.59}$$

This architecture, which aligns with the general framework introduced in this section, captures the essential message-passing mechanism through Equations 1.57 and 1.58, with Equation 1.59 enabling graph-wide predictions. The MPNN framework thus serves as the foundational template for many influential variants, including GraphSAGE (Hamilton et al., 2017) and Graph Attention Networks (Veličković et al., 2018).

1.4.3.5 Gated Graph Neural Network

Gated Graph Neural Networks (GGNNs) extend standard message-passing frameworks by integrating the Gated Recurrent Unit (GRU) mechanism (Cho et al., 2014b), which allows each node to control how much information it receives from neighbors while retaining relevant past context, thus effectively modeling long-range dependencies and evolving interactions (Li et al., 2016).

The iterative node update at step t refines the state of node i through a structured sequence. The process begins with message aggregation, where a contextual message $\mathbf{m}_i^{(t)}$ is computed for node i by summing the previous hidden states of its neighbors:

$$\mathbf{m}_i^{(t)} = \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(t-1)}. \tag{1.60}$$

This aggregated message then feeds into the gated information control phase. Here, two gating vectors are computed to manage the state transition. First, the update gate $\mathbf{z}_i^{(t)}$ determines the balance between retaining the previous state and incorporating new information:

$$\mathbf{z}_i^{(t)} = \sigma\big(\mathbf{W}_z \mathbf{m}_i^{(t)} + \mathbf{U}_z \mathbf{h}_i^{(t-1)}\big). \tag{1.61}$$

Concurrently, the reset gate $\mathbf{r}_i^{(t)}$ controls how much of the past state contributes to a candidate update:

$$\mathbf{r}_i^{(t)} = \sigma\big(\mathbf{W}_r \mathbf{m}_i^{(t)} + \mathbf{U}_r \mathbf{h}_i^{(t-1)}\big). \tag{1.62}$$

Finally, the state update produces the new hidden state. A candidate activation $\tilde{\mathbf{h}}_i^{(t)}$ is first generated by combining the aggregated message with a gated version of the previous state:

$$\tilde{\mathbf{h}}_i^{(t)} = \tanh\big(\mathbf{W}_h \mathbf{m}_i^{(t)} + \mathbf{U}_h (\mathbf{r}_i^{(t)} \odot \mathbf{h}_i^{(t-1)})\big). \tag{1.63}$$

The new state $\mathbf{h}_i^{(t)}$ is then formulated as a gated interpolation between the previous state and this candidate, weighted by the update gate:

$$\mathbf{h}_i^{(t)} = (1 - \mathbf{z}_i^{(t)}) \odot \mathbf{h}_i^{(t-1)} + \mathbf{z}_i^{(t)} \odot \tilde{\mathbf{h}}_i^{(t)}. \tag{1.64}$$

In this formulation, σ denotes the logistic sigmoid function, $\odot$ the element-wise product, and $\mathbf{W}_*$ and $\mathbf{U}_*$ are learnable weight matrices. This gated update mechanism empowers GGNNs to stabilize learning over long sequences of propagation steps, leading to their successful application in data-rich domains such as program verification (Li et al., 2016).

1.4.3.6 Graph Autoencoders (GAEs)

Building on the definition provided earlier, where graph autoencoders "adapt the autoencoder framework to non-Euclidean domains by incorporating GNNs within the encoder and/or decoder" and learn embeddings that preserve topological structure, this section delves into the specific architecture and motivations behind GAEs. In essence, graph autoencoders (GAEs) pair a GNN-based encoder (e.g., GCN, GAT, GraphSAGE) with a decoder, often a simple inner product or a more complex neural network, that reconstructs the original graph structure, typically by predicting topology and optionally node features. This reconstruction objective provides a strong unsupervised training signal. By forcing the model to capture the graph's connectivity, GAEs learn dense, low-dimensional node representations that are well suited to tasks where relational information and structural similarity are critical, such as link prediction, community detection, and downstream semi-supervised classification (Su et al., 2022; Wu et al., 2022).

Concretely, we consider undirected graph $\mathcal{G} = (V, E)$ with N nodes. Let $\mathbf{X} \in \mathbb{R}^{N\times m}$ denote the node feature matrix, where m is the dimension of the feature vector for each node. Let $\mathbf{A} \in \{0,1\}^{N\times N}$ be the adjacency matrix. The encoder maps $(\mathbf{X}, \mathbf{A})$ into a compressed latent embedding matrix $\mathbf{Z} \in \mathbb{R}^{N\times d}$, where d is the embedding dimension. This is achieved via one or more GNN layers:

$$\mathbf{Z} = \text{Encoder}(\mathbf{X}, \mathbf{A}) = \text{GNN}(\mathbf{X}, \mathbf{A}), \tag{1.65}$$

where each row $\mathbf{z}_v$ is the embedding of node v. The decoder then reconstructs the adjacency (or its probabilistic variant) from these embeddings, most commonly using an inner-product decoder:

$$\hat{\mathbf{A}} = \sigma\big(\mathbf{Z}\mathbf{Z}^\top\big), \tag{1.66}$$

where σ is an activation function, and training minimizes an objective function such as the binary cross-entropy reconstruction loss.

Recent advances have further enhanced GAEs. By imposing a Gaussian prior on $\mathbf{Z}$ and adding a Kullback-Leibler divergence term. **Variational Graph Autoencoders (VGAEs)** (Thomas and Welling, 2016) learn smoother latent spaces that facilitate generative modeling of graphs and provide stronger regularization. Meanwhile, for heterogeneous or attributed graphs, domain-specific decoders have been developed to reconstruct multiple relation types or node and edge attributes, extending the applicability of GAEs to domains such as social networks, citation graphs, and molecular structures. These innovations, combined with demonstrated success on link prediction, community detection, and dynamic graph modeling, underscore the foundational role of GAEs in unsupervised graph representation learning.

1.5 Conclusion

In summary, this background chapter has laid the groundwork by defining graphs, outlining the key learning paradigms, surveying deep-learning architectures and showing how these ideas extend to graph data. In the next chapter, we will formally define the community-detection problem, examine its applications and standard benchmarks, and present a state-of-the-art review of community-detection methods organized according to the taxonomy developed in this thesis, thereby setting the stage for the novel contributions that follow.

Chapter 2

State of the Art

Community detection in complex networks has become a pivotal research area due to its wide range of applications in domains such as social network analysis, biology, and recommendation systems. This chapter provides a comprehensive overview of the state of the art in community detection, offering insights into the evaluation metrics used to assess algorithm performance, the datasets that serve as benchmarks, and the practical applications that motivate this research. We begin by introducing key evaluation metrics and datasets, including synthetic and real-world networks, followed by a discussion of the domains where community detection plays a crucial role.

The chapter then systematically explores the evolution of community detection algorithms, beginning with classical approaches based on graph theory and optimization techniques. Next, we transition to traditional machine learning–based techniques. Building upon this, we delve into the recent surge in deep learning–based community detection.

2.1 Community Detection

Community detection is a fundamental task in network analysis, aiming to partition the nodes of a graph into k communities $C = \{C_1, C_2, \ldots, C_k\}$. Although there is no universally accepted definition, the most widely used characterizations emphasize both structural and semantic cohesion. Semantically, a community is a set of nodes that share similar attributes, such as belonging to related categories, exhibiting analogous behavioral patterns, or performing similar functional roles. Structurally, a community is a subgraph whose nodes are densely interconnected yet sparsely connected to nodes outside the subgraph. Thus, a well-defined community maximizes internal cohesion, reflected by high intra-community edge density and node degrees, and minimizes external ties, consistent with the principle that intra-community connections should outweigh inter-community ones (Fortunato, 2010).

Communities may be classified as disjoint, overlapping, or hierarchical. Disjoint communities partition nodes such that $C_i \cap C_j = \emptyset$ for all $i \neq j$. Overlapping communities allow nodes to belong to multiple groups simultaneously, so there exist i and j with $C_i \cap C_j \neq \emptyset$. Hierarchical communities form nested structures in which larger, loosely connected clusters encompass smaller, densely interconnected subcommunities, often visualized as a dendrogram to reveal multilevel organization.

The notion of a community can vary by domain. For example:

- **Sociology:** A social unit (group) defined by its members' shared identity and "we-feeling" (sentiment), often bounded by geography (community of place) and/or common interests (community of interest), and characterized by the patterns of social interaction and shared institutions that allow the members to sustain a common life (Fortunato, 2010)
- **Ecology:** A set of interacting species coexisting within a particular environment (Lusseau et al., 2003).
- **Economics:** Individuals sharing a common market and engaging in exchange of goods and services (Reichardt and Bornholdt, 2007).
- **Political Science:** A population defined by geographic or political boundaries sharing a collective political identity, such as a nation or state (Zhang et al., 2008).

- **Biology:** Interdependent organisms inhabiting the same geographic area and interacting with each other (Barabasi and Oltvai, 2004).
- **Computer Science:** Users or organizations that interact and share common interests or objectives (Newman and Girvan, 2004).

2.1.1 Evaluation Metrics

To evaluate methods for the task of community detection, we describe the most commonly used evaluation metrics: NMI, ARI, F1-Score, and Modularity Q.

Normalized Mutual Information (*NMI*) is a widely used metric for evaluating community detection performance. It quantifies the similarity between the detected community structure (C^*) and the ground truth (C) by leveraging mutual information and entropy of the partitions. Higher NMI values indicate a stronger alignment between the two community assignments. NMI is defined by Equation 2.1 (Amelio and Pizzuti, 2015).

$$\mathrm{NMI}(C, C^*) = \frac{-2\sum_{i=1}^{k}\sum_{j=1}^{k^*} n_{ij}\log\frac{n_{ij}N}{n_i n_j}}{\sum_{i=1}^{k} n_i \log\frac{n_i}{N} + \sum_{j=1}^{k^*} n_j \log\frac{n_j}{N}}. \tag{2.1}$$

The Adjusted Rand Index (*ARI*) measures the similarity between detected and ground-truth partitions by comparing all pairs of nodes, counting agreements and disagreements, and correcting for chance. It yields values in the range $[-1, 1]$, where higher scores indicate stronger alignment with the ground truth (Hubert and Arabie, 1985).

$$ARI(C, C^*) = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}}. \tag{2.2}$$

In Equations (2.1) and (2.2), k and k^* denote the number of ground-truth and detected communities, respectively. C and C^* represent the true and predicted community assignments. n_{ij} is the count of nodes that belong to the community i in C and the community j in C^* simultaneously.

Modularity, introduced by Newman and Girvan, measures how much the density of edges within communities exceeds what would be expected in a random graph with the same degree distribution. It takes values between –1 and 1, with higher values indicating stronger community structure. The modularity is given by Equation 2.3.

$$Q = \sum_{c=1}^{k} (e_{cc} - a_c^2), \tag{2.3}$$

where k is the number of communities, e_{cc} is the fraction of all edges that lie within community c, and a_c is the fraction of all edge-ends (stubs) attached to nodes in community c. Here, e_{cc} captures the observed intra-community connectivity, while a_c^2 represents its expected value under the null model.

Newman reformulated modularity for spectral optimization. For an undirected graph $G = (V, E)$, adjacency matrix $\mathbf{A}$, and node degrees $k_i = \sum_j A_{ij}$, the spectral definition is given by Equation 2.4.

$$Q = \frac{1}{4M} \sum_{i,j} \left(\mathbf{A}_{ij} - \frac{k_i k_j}{2M} \right) \delta(c_i, c_j), \tag{2.4}$$

where $\delta(c_i, c_j) = 1$ if the nodes i and j belong to the same community and 0 otherwise. The term A_{ij} is the edge (possibly weighted) between i and j, and $\frac{k_i k_j}{2M}$ is the expected edge weight in the null model.

The F1-score integrates precision and recall through their harmonic mean, providing a single balanced measure of the performance of an algorithm (Yang et al., 2013). It is defined by the Equation 2.5.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

$$\text{Precision} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}$$

$$\text{Recall} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}$$

NMI measures the agreement between ground truth and predicted community assignments, while ARI evaluates clustering quality by correcting for chance agreements. The F1-Score harmonizes precision and recall to assess both the accuracy and completeness of individual community assignments. Modularity, in turn, quantifies the coherence of detected communities by comparing the observed density of intra-community edges to that expected in a random graph. Together, these four metrics provide a comprehensive evaluation: NMI and ARI capture overall partition agreement and clustering fidelity, the F1-Score highlights assignment accuracy versus coverage, and modularity reflects the structural strength of the communities.

2.1.2 Datasets

Both real-world networks and synthetic benchmarks are considered, covering a range of sizes, densities, and community structures.

2.1.3 Real-World Datasets

We explore several well-known networks drawn from social, co-authorship and citation domains. Nodes represent entities (users, authors, documents), edges denote interactions (friendships, collaborations, citations), and many include node features and ground-truth community labels (possibly overlapping). Table 2.1 summarizes key statistics for the real-world datasets we explore.

Table 2.1—Summary of real–world community detection datasets. Columns: F = feature available, S = feature dimension, C = number of communities, O = overlapping membership (✓= yes, X = no). A ✓indicates the presence of node features or overlapping communities.

Category	Dataset	Nodes	Edges	F	S	C	O
Social	Zachary's Karate Club	34	78	X	–	2	X
	Dolphins	62	159	X	–	2	X
	American Football	115	613	X	–	12	X
	Email–Eu–core	1,005	16,706	X	–	42	X
	Facebook 1684	792	28,048	✓	15	17	✓
	BlogCatalog	5,196	171,743	✓	8,189	6	X
Co-authorship	ENG	14,927	98,610	✓	4,800	16	✓
	CS	21,957	193,500	✓	7,800	18	✓
Citation	Cora	2,708	5,429	✓	1,433	7	X
	CiteSeer	3,327	4,732	✓	3,703	6	X
	PubMed	19,717	44,338	✓	500	3	X

2.1.3.1 Co-authorship and Social Networks

Co-authorship and social networks are commonly used in graph mining and community detection tasks. In these networks, nodes often represent individuals such as authors or users, and edges represent interactions like collaborations or friendships. Some datasets also include node features derived from user profiles or textual content, and many provide ground-truth community labels, including overlapping memberships.

Zachary's Karate Club: This classic dataset represents social interactions among 34 members of a university karate club, with 78 edges denoting friendships. The network is frequently used to test community detection algorithms, with ground-truth communities corresponding to a real-life split in the club (Zachary, 1977).

Dolphins Network: This network captures the social structure among 62 bottlenose dolphins observed in Doubtful Sound, New Zealand. The 159 edges represent frequent associations, and the network is naturally divided into two communities (Lusseau et al., 2003).

American Football: This graph contains 115 college football teams from the 2000 season, where edges represent scheduled games. The 613 edges define communities corresponding to 12 athletic conferences (Girvan and Newman, 2002).

Email-Eu-core: This dataset models email communication between 1,005 members of a large European research institution. It includes 16,706 email links and 42 ground-truth departments as communities (Leskovec et al., 2007).

Facebook 1684: An ego-centric social network of 792 users and 28,048 links. Communities are based on user-defined circles (e.g., family, friends, coworkers). The network includes 15-dimensional binary profile features (Mcauley and Leskovec, 2014).

Computer Science Co-authorship (CS): A large co-authorship network with 21,957 authors and 193,500 collaboration links. Each author is described by a 7,800-dimensional feature vector derived from the publication data. The network includes 18 overlapping research communities (Shchur and Günnemann, 2019).

Engineering Co-authorship (ENG): Similar to the CS dataset, the ENG network comprises 14,927 authors and 98,610 co-authorship links, with 4,800-dimensional feature vectors. It contains 16 research-based communities with significant overlap (Shchur and Günnemann, 2019).

BlogCatalog: The BlogCatalog network is a social graph of 5,196 users linked by 171,743 social relationships. Each user is associated with an 8,189-dimensional feature vector constructed from blog metadata and tag information. The dataset includes six overlapping community labels that represent user interests. BlogCatalog is widely used to evaluate attributed community detection methods due to its high dimensionality of characteristics and realistic social structure (Li et al., 2015).

2.1.3.2 Citation Networks

Citation networks are structured graphs in which the nodes represent scientific documents and the edges denote the citation relationships. These datasets provide structured node features derived from textual content such as paper titles and abstracts. However, raw text is not included; only feature vectors (e.g., binary word presence or TF–IDF representations) are available.

Cora: The Cora dataset consists of 2,708 scientific publications classified into seven topics related to machine learning. Each node represents a publication, and the edges denote citation relationships, resulting in 5,429 undirected links. Every document is characterized by a binary feature vector, indicating the presence or absence of 1,433 unique terms extracted from the text. This dataset is widely used for benchmarking graph-based learning tasks due to its moderate size and rich feature representation (McCallum et al., 2000).

CiteSeer: CiteSeer is a citation network comprising 3,327 computer science publications grouped into six categories. The network includes 4,732 citation links, which form an undirected graph. Each document is described by a 3,703-dimensional binary word vector, constructed by mapping the title and abstract to a predefined vocabulary. CiteSeer offers a more challenging structure due to its higher feature sparsity and noisier class distribution compared to Cora (Giles et al., 1998).

PubMed: PubMed, a database curated by the National Library of Medicine of the United States, hosts a wealth of scientific literature, particularly in the life sciences and biomedical fields. Its citation dataset on diabetes includes 19,717 research articles, each classified into one of three categories (Diabetes Mellitus: Experimental, Type 1, or Type 2). The associated citation graph contains 44,338 connections. Each article is represented by a 500-dimensional feature vector based on TF–IDF scores of medical terms. The PubMed dataset provides a large-scale and semantically rich test to evaluate graph algorithms in the biomedical domain (Sen et al., 2008).

2.1.3.3 Synthetic Benchmarks

Synthetic benchmarks serve as controlled testbeds for evaluating community detection algorithms under varying network conditions. Unlike real-world data, synthetic graphs allow precise control over structural properties (e.g., degree distribution, community size, mixing) so that the algorithm behavior can be systematically assessed.

LFR benchmarks: The Lancichinetti–Fortunato–Radicchi (LFR) benchmark graphs (Lancichinetti et al., 2008) are widely considered the standard for evaluating community detection algorithms under realistic network heterogeneity. They generate synthetic networks with power-law distributed degrees and community sizes, allowing precise control over the mixing parameter μ (the fraction of a node's links connecting outside its planted community). By tuning μ and the exponents of the degree and community-size distributions, LFR benchmarks simulate diverse network topologies, from well-separated communities to highly overlapping structures, making them indispensable for stress-testing algorithmic robustness across detection regimes.

Girvan–Newman (GN) benchmarks: The Girvan–Newman benchmark (Girvan and Newman, 2002) is one of the earliest synthetic models for assessing clustering performance. It generates networks with $N = 128$ nodes divided into four equal-sized communities (32 nodes each), where each node has an average degree of $k = 16$. The ratio of intra- to inter-community links is controlled by a tunable parameter $(d_{\text{in}}/d_{\text{out}})$. $\mathbf{d}_{\text{in}}$ and $\mathbf{d}_{\text{out}}$ are the expected number of links inside a node's community (intra-community degree) and the expected number of links connecting to other communities (inter-community degree), respectively. Despite its simplicity (uniform degrees and community sizes), it remains useful for illustrating classical divisive algorithms (e.g., edge-betweenness removal) in idealized settings before progressing to complex benchmarks like LFR.

2.1.4 Application Domains

Community detection techniques have proven useful in a variety of fields, offering insight into latent groupings and improving downstream tasks (Su et al., 2022). Typical areas are discussed in the following parts:

Recommender Systems

By identifying user communities with shared interests or behaviors, recommender systems can suggest items that are popular within a user's community, thereby improving personalization and relevance (Hao et al., 2023).

Community Search

Given a query node or set of seed nodes, community search aims to quickly find the most relevant community subgraph containing them. This is useful in social networks, citation graphs, and web navigation (Fang et al., 2020).

Anomaly Detection

Anomaly detection in networks uses community structure as a baseline for "normal" interactions: most connections occur within densely linked node communities. Edges that span unrelated communities or deviate from expected patterns are flagged as anomalies (Safdari and Bacco, 2022). This approach can uncover fraud rings in financial networks, misinformation campaigns on social media, and failures in critical infrastructure by highlighting behavior that diverges from the established community layout.

Online Social Networks

Community detection in online social networks (OSNs) underpins a wide range of practical applications. By uncovering clusters of users with similar interests or behaviors, it enables more accurate personalized recommendations and targeted advertising on platforms such as Facebook and LinkedIn (Kumar et al., 2006; Konstantinidis et al., 2013). In cybersecurity, community-based methods have been employed to identify coordinated botnets and fraudulent accounts on Twitter, thereby curbing misinformation and malicious campaigns (Ferrara et al., 2016). During emergencies, detecting emergent community structures in real time helps map interdependent user groups, improving the allocation of resources and the efficiency of crisis response efforts (Sakaki et al., 2010). In terrorist networks, "friend-of-a-friend" influence models further illustrate how latent communities can reveal covert operational cells (Waskiewicz, 2012).

Biochemistry

Community detection techniques have become indispensable in the analysis of biochemical networks, where they reveal functional modules and elucidate complex biological processes. In protein–protein interaction (PPI) networks, densely connected subgraphs often correspond to multi-protein complexes or signaling cascades, so identifying these communities helps predict protein function and prioritize drug targets (Spirin and Mirny, 2003). In metabolic networks, community structure uncovers pathways or reaction clusters that share common substrates and regulators, highlighting key metabolic modules and their cross-talk (Guimerà and Amaral, 2005). In gene co-expression networks derived from transcriptomic data, community detection groups genes with correlated expression profiles, facilitating the discovery of co-regulated gene sets and putative transcriptional regulators (Langfelder and Horvath, 2008).

Functional Brain Networks

Community detection in brain networks uncovers modules, such as default mode, visual, and sensorimotor systems, by grouping nodes with dense intra-module and sparse inter-module connections, thereby capturing the balance between segregation and integration in healthy cognition (Power et al., 2011). In disorders like Alzheimer's and schizophrenia, altered community structure, manifested as reduced intra-module connectivity or shifted module boundaries, serves as a biomarker of dysfunction and helps track disease progression and treatment effects (He et al., 2008).

Given the wide range of techniques developed for community detection, ranging from early graph-theoretic heuristics to modern neural models, it is essential to structure the field into coherent methodological categories. In this survey, we adopt a taxonomy that classifies community detection approaches into three major families: (i) **classical methods** (Section 2.2), encompassing modularity optimization, spectral clustering, and label propagation algorithms; (ii) **traditional machine learning–based methods** (Section 2.3), including statistical inference models, nonnegative matrix factorization, and topic modeling; and (iii) **deep learning–based methods** (Section 2.4), which leverage neural architectures such as graph convolutional networks, attention mechanisms, and generative models.

To visually summarize this classification, Figure 2.1 presents our adopted taxonomy and representative methods within each category. In addition, Figure 2.2 provides a chronological overview of key developments in the field, illustrating the evolution from classical algorithms to learning-based and deep neural approaches.

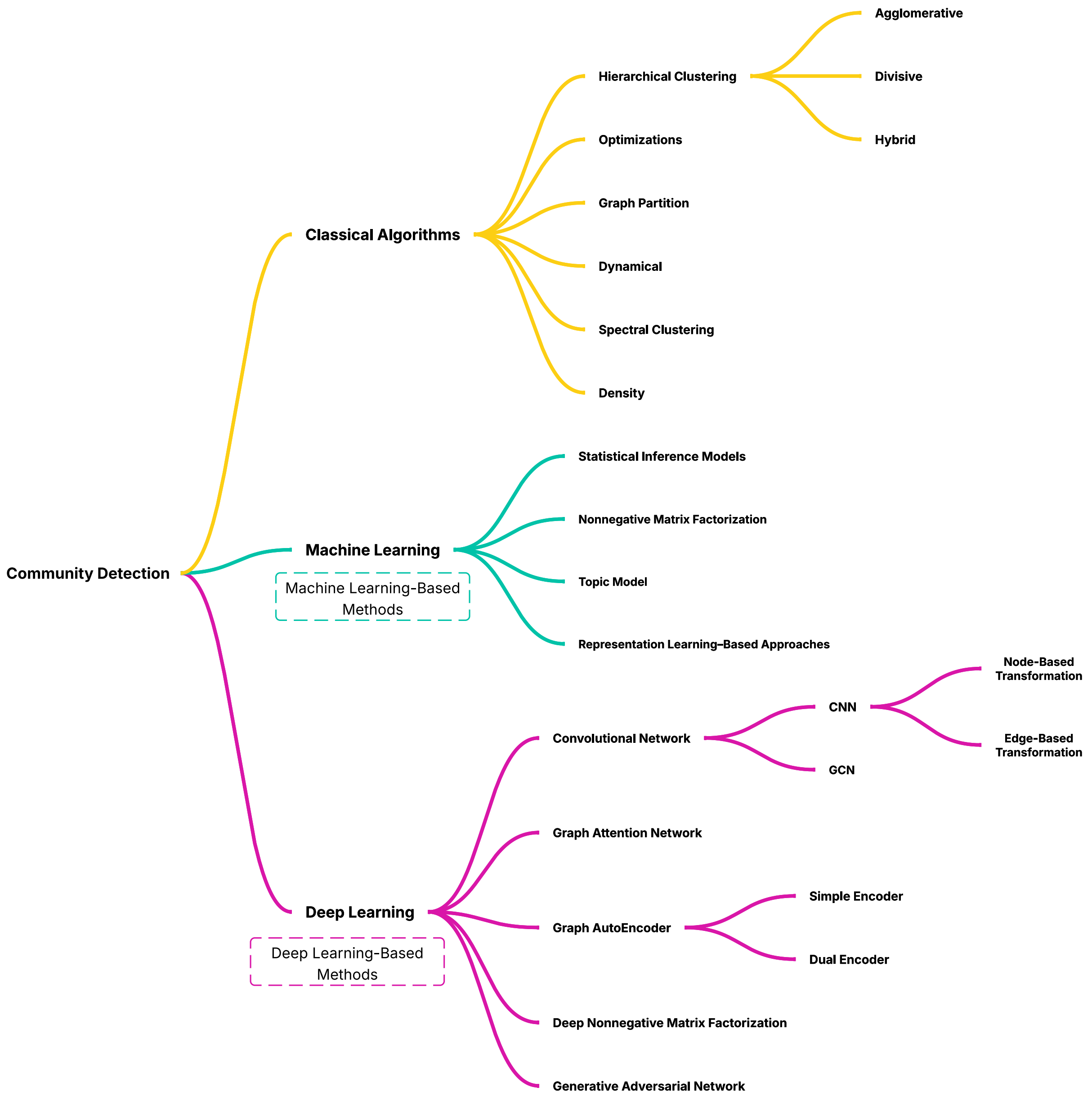


Figure 2.1—Taxonomy of community detection methods adopted in this survey. The field is organized into three main categories: classical algorithms, traditional machine learning–based approaches, and deep learning–based models. Each category contains representative techniques.

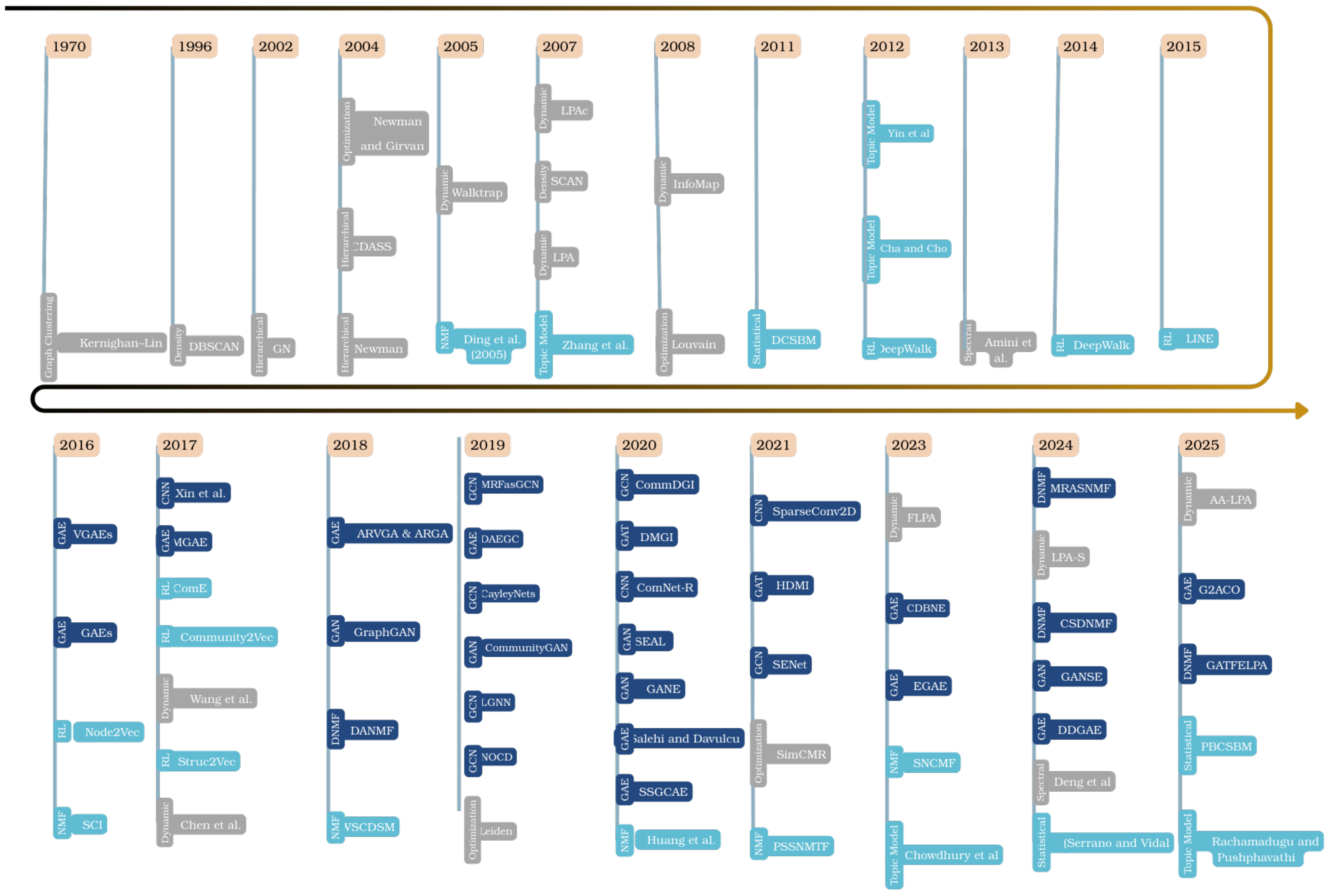


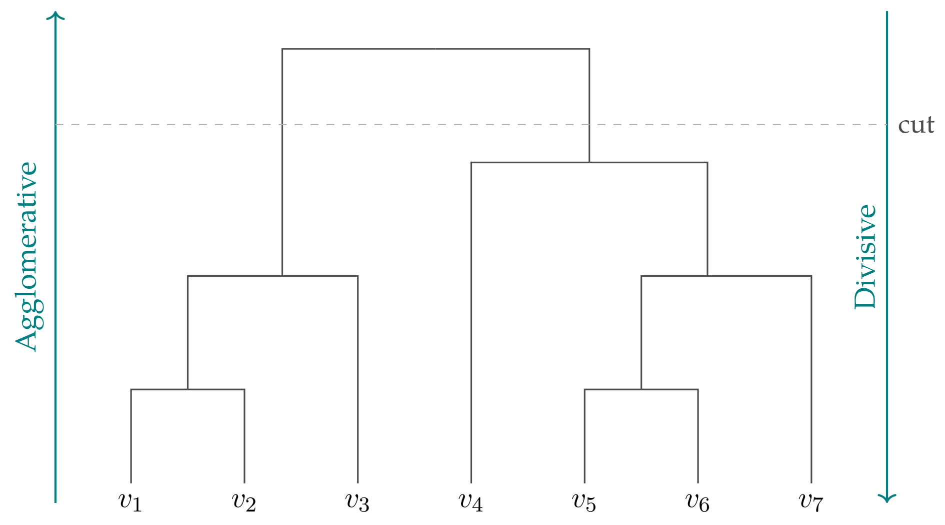
Figure 2.2—Timeline of key developments in community detection. Early research focused on classical methods (colored gray), followed by traditional machine learning models (light blue), and deep learning–based approaches (blue).

2.2 Classical Community Detection Algorithms

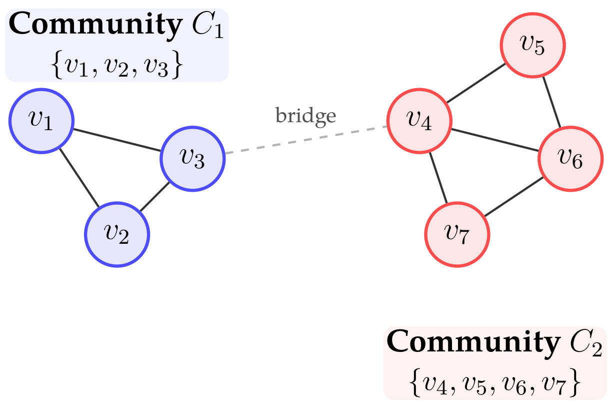
There are many classical approaches to uncover community structures in graphs, which can be grouped into several main categories. These include hierarchical methods, dynamic approaches, density-based techniques, spectral clustering, and optimization-based methods.

2.2.1 Hierarchical Methods

Hierarchical clustering methods work by forming nodes into nested partitions, often illustrated as a dendrogram (Figure 2.3). In such a dendrogram, leaves correspond to individual nodes and internal nodes denote successively larger communities, exhibiting multiple levels of vertex groupings: small clusters contained within larger clusters, which themselves are contained within even larger clusters, and so on. A key advantage of hierarchical clustering is that it requires no prior information about the number or size of clusters. However, it produces numerous possible partitions without providing a clear criterion for selecting the one that best reflects the graph's true community structure. Hierarchical clustering takes one of three forms: divisive, agglomerative, or hybrid, each offering a distinct strategy for splitting or merging nodes to form nested community levels (Su et al., 2022).



(a) Example of a dendrogram cut at a certain level to form communities in a seven-node network $[v_1, \ldots, v_7]$. Arrows on the left and right illustrate bottom-up (agglomerative) versus top-down (divisive) clustering directions.



(b) Example of detecting two communities (C_1 and C_2) in a seven-node network, laid out so that each community's internal structure is clear. The dashed gray "bridge" edge between v_3 and v_4 corresponds exactly to the height at which the dendrogram was cut.

Figure 2.3—Illustration of hierarchical clustering: (a) the dendrogram showing how nodes merge or split at each level, and (b) the resulting graph with two detected communities.

2.2.1.1 Divisive Approach

Divisive methods, also known as top-down hierarchical approaches, operate on a fundamental principle: communities emerge by systematically identifying and removing edges that bridge distinct clusters, thereby disconnecting network regions. This philosophy adapts hierarchical clustering techniques to graph structures, but with a critical distinction. The core challenge lies in establishing a robust criterion to distinguish these inter-community connections (bridges) from intra-community edges. These methods begin with the entire network as a single community of size N. At each iteration, one of the existing communities is selected and split into two subcommunities. This splitting procedure continues recursively until every community contains exactly one node.

Edge betweenness centrality serves as a widely used criterion for identifying inter-community edges, first formally defined by Freeman (1977) and adapted by Girvan and Newman (2002) for community detection. For an edge e, it is expressed as:

$$B(e) = \frac{\sum_{s \neq t} \sigma_{st}(e)}{\sum_{s \neq t} \sigma_{st}}, \tag{2.6}$$

where σ_{st} denotes the total number of shortest paths (geodesics) between nodes s and t, and $\sigma_{st}(e)$ is the number of those shortest paths that traverse edge e. Intuitively, an edge with high betweenness lies on many shortest paths and therefore serves as a bridge between different regions of the network. By iteratively removing edges of highest betweenness, one effectively disconnects the network along its weakest inter-community links, thereby revealing a nested hierarchy of partitions.

The Girvan–Newman (GN) algorithm proceeds as follows:

1. Compute the betweenness centrality $B(e)$ for every edge e in the current network.
2. Identify and remove the edge e^* with maximum betweenness; if multiple edges tie for maximum, break the tie arbitrarily.
3. Recompute betweenness centrality for all remaining edges in the modified network.
4. Repeat steps 2–3 until no edges remain (or until the desired number of communities has been obtained).

The computational bottleneck of Girvan–Newman lies in recomputing edge-betweenness after each edge removal. Calculating edge-betweenness on a graph requires $O(MN)$ time, and when the graph is sparse ($M = O(N)$), this reduces to $O(N^2)$. Because this computation must be repeated for up to M edge removals, the overall complexity becomes $O\big(M \times (MN)\big) = O\big(M^2N\big)$, which, in the sparse regime where $M = O(N)$, simplifies to $O(N^3)$.

2.2.1.2 Agglomerative Approach

Agglomerative hierarchical clustering for community detection initializes with N singleton partitions (one per node) and iteratively merges the pair of communities yielding the highest modularity gain, as formalized by $\Delta Q = \frac{1}{2M}\left[2\mathbf{A}_{ij} - \frac{k_i k_j}{M}\right]$ where $\mathbf{A}_{ij}$ denotes adjacency, k_i degree of node i, and M total edges. This greedy optimization framework was pioneered by Newman (2004), who established modularity maximization as the merging criterion to reveal optimal community structures. The approach constructs a dendrogram through successive merges until all nodes coalesce into a single community, with the optimal partition identified at the global modularity maximum $Q_{\text{max}} = \max \sum_c \left[\frac{l_c}{M} - \left(\frac{d_c}{2M}\right)^2\right]$ where l_c and d_c represent intra-community links and total community degree, respectively. While theoretically principled, the original algorithm's $O(N^2)$ complexity on sparse graphs limited scalability. This bottleneck was addressed by Clauset et al. (2004) through the CNM (Clauset-Newman-Moore) algorithm, which achieves $O(N \log^2 N)$ complexity via three innovations: (1) a max-heap storing ΔQ values for all adjacent community pairs, enabling $O(1)$ retrieval of optimal merges; (2) a balanced tree tracking community memberships; and (3) incremental updating where only ΔQ values for neighbors of merged communities require recalculation. By reducing merge operations from quadratic to near-linear efficiency, CNM enabled community detection in networks with over 10^6 nodes, demonstrating 100-fold speedups over the baseline approach while maintaining equivalent modularity optimization. Subsequent refinements have further optimized CNM's heap structures, but its core architecture remains foundational for scalable agglomerative clustering in network science.

2.2.1.3 Hybrid Approach

Hybrid methods combine both divisive and agglomerative strategies. CDASS (Community Detection in Complex Networks Using Structural Similarity) (Zarandi and Rafsanjani, 2018) is a representative example of a hierarchical hybrid approach. The algorithm simultaneously applies edge removal and community merging. Specifically, it begins by randomly removing low-similarity edges, fragmenting the graph into several disconnected components, each considered a preliminary community. These initial communities are then merged based on a modularity-inspired evaluation function to approximate the actual community structure. Finally, an evolution-based function is used to select the partition that best represents the global community configuration.

2.2.2 Graph Clustering

Graph clustering (or graph partitioning) involves dividing the nodes into k groups to minimize the number of edges between groups. The parameter k determines the number of communities. In practice, partitioning a network into k communities requires cutting edges between groups. The total number of edges crossing communities is called the *cut size* (Fortunato, 2010). Kernighan–Lin algorithm (Kernighan and Lin, 1970), originally developed to optimize electronic circuit layouts, minimizes interconnections between components on separate boards. This foundational heuristic splits a graph into two equally sized subsets using a gain function Q which quantifies the net benefit of moving a node between subsets by comparing intra- and inter-community edges. After initializing two disjoint sets (randomly or with prior information), node pairs (one from each set) are evaluated for swapping, with each node moved at most once per pass. The sequence of swaps yielding the highest cumulative gain Q is applied. To escape local optima, Q is recalculated after each tentative swap, and the configuration with maximal gain is retained. This pairwise-exchange procedure iterates until all nodes are evaluated.

Beyond local-swap heuristics, spectral bisection (Barnes, 1982) leverages the eigenstructure of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ (where $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ the adjacency matrix). The nodes are partitioned according to the Fiedler vector (the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}$), with components divided by sign or rank to achieve a balanced bipartition that approximately minimizes the size of the cut. Other graph clustering approaches include geometric cuts, multilevel algorithms, and level-structure partitioning (Karypis, 1997; Pothen, 1997).

2.2.3 Density-Based Techniques

Data distributions can exhibit diverse geometries, and non-convex structures may assume arbitrary shapes. Conventional methods such as k-means often struggle with these irregular clusters. In contrast, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) leverages density connectivity principles to detect arbitrarily shaped clusters using only two parameters, ε (the radius of the neighborhood around a data point) and MinPts (the minimum number of points required within ε to form a dense region), while explicitly identifying noise.

Formally, for a dataset V and a point $p \in V$, the ε-neighborhood is defined as:

$$N_\varepsilon(p) = \{\, r \in V \mid \text{dist}(p, r) \leq \varepsilon \,\} \tag{2.7}$$

A point p qualifies as a core point if $|N_\varepsilon(p)| \geq \text{MinPts}$. As shown in Figure 2.4, DBSCAN groups points into core points, border points, and noise based on the two parameters ε and $MinPts$ and is able to recover arbitrarily-shaped clusters.

DBSCAN then categorizes each point as follows: if $|N_\varepsilon(p)| \geq \text{MinPts}$, p is a core point; if p does not satisfy this criterion but lies within the ε-neighborhood of some core point, p is a border point; otherwise, p is classified as noise. Clusters emerge as maximal sets of density-connected core points, with border points assigned to the clusters of their associated cores. This density-based framework has inspired numerous extensions (You et al., 2016; Xu et al., 2007). For example, SCAN (Structural Clustering Algorithm for Networks) identifies disjoint communities, bridge nodes (which link clusters), and outliers in large networks by analyzing edge density between nodes (Xu et al., 2007).

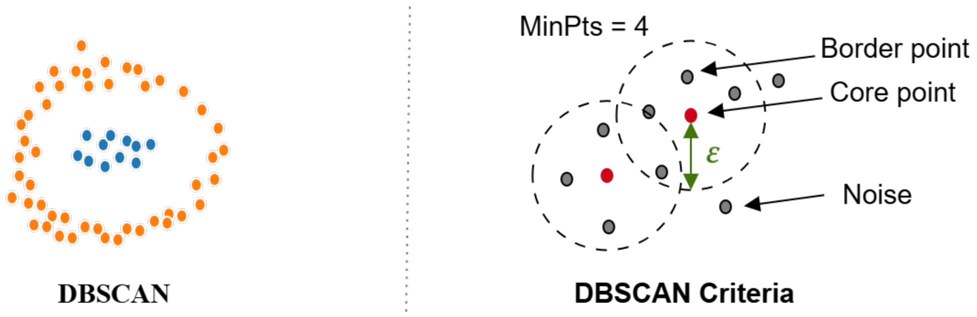


Figure 2.4—Illustration of the DBSCAN algorithm. (*Left*) Clustering result on an illustrative two-ring dataset. (*Right*) The DBSCAN criteria: a point p is a *core point* if $|N_\varepsilon(p)| \geq \text{MinPts}$; a *border point* if it lies in the ε-neighborhood of a core point but has $|N_\varepsilon(p)| < \text{MinPts}$; otherwise it is a *noise*.

2.2.4 Spectral Clustering

Spectral clustering is a widely adopted method for grouping data points or nodes within a graph, leveraging the spectral characteristics of the data or the graph Laplacian matrix. The core principle of spectral clustering involves representing data points or graph nodes in a reduced-dimensional embedding, obtained from the eigenvectors of the Laplacian, such that clusters become more distinguishable in this lower-dimensional space. Once the embedding is computed, a conventional clustering algorithm (e.g., k-means) is applied to partition the points based on their coordinates in the spectral domain (Amini et al., 2013; Deng et al., 2024).

Metaheuristic methods detect community structure by exploring the solution space in a systematic way without guaranteeing a global optimum. In contrast to heuristics that depend on problem-specific rules, metaheuristics offer general search frameworks inspired by natural processes. Examples include Genetic Algorithms (Holand, 1975), Particle Swarm Optimization (Kennedy and Eberhart, 1995), and Simulated Annealing (Kirkpatrick et al., 1983). There is extensive research on metaheuristic approaches for community detection (Bara'a et al., 2021).

Genetic Algorithms improve candidate partitions through selection, crossover, and mutation while a fitness function directs the search toward better solutions. Particle Swarm Optimization relies on collective movement patterns to update solution positions. Simulated Annealing uses controlled randomness to escape local optima and approach high-quality solutions. Hybrid and hierarchical variants extend these principles by combining metaheuristic search with recursive graph partitioning. Saoudi and Moussaoui (Saoud and Moussaoui, 2018) introduce a hierarchical genetic approach that applies evolutionary optimization at each bipartition to maximize modularity. The graph is recursively divided once an optimal split is found, producing a dendrogram that reflects the hierarchical organization of the network. The method is evaluated on small classical social-network datasets, such as Zachary's Karate Club, Dolphins, and American Football.

2.2.5 Optimization-Based Methods

Optimization broadly refers to the process of determining input parameters or arguments that maximize or minimize a given objective function. In the context of community detection, a prominent task is identifying community structures in networks by optimizing a quality function, most commonly modularity Q (a formal definition of modularity is provided in Section 2.1.1). Modularity-based algorithms address this problem by evaluating the quality of graph partitions.

Newman and Girvan introduced the concept of modularity and proposed a greedy algorithm for community detection using modularity as a hierarchical clustering criterion (Newman and Girvan, 2004). The algorithm initializes with each node as its own community and iteratively merges community pairs to achieve the maximum modularity gain at each step. This agglomerative process continues until no further merges improve modularity, resulting in a hierarchical dendrogram of communities.

Modularity optimization gained broader attention with the Louvain method by Blondel et al. (2008), which operates in a series of passes, each consisting of two phases. In the first phase, every node is treated as a separate community. For each node v_i, the algorithm calculates the modularity gain from moving v_i into the community of each neighbor v_j, with gains ranging from -1 to $+1$. If the highest gain is positive, v_i is reassigned to the corresponding community; otherwise, it remains unchanged. This process continues until no single-node move yields a modularity increase. Although the node update order does not significantly affect final modularity, it can influence computational cost, and heuristic node ordering may improve convergence.

Once phase one concludes, the second phase constructs a new network by collapsing each community into a "super-node". Edges between super-nodes represent the aggregate weight of edges between their constituent nodes, and self-loops capture intra-community connections. This coarse network becomes the input for the next pass. The two-phase process repeats until modularity can no longer be improved. Through iterative coarsening and refinement, the Louvain method efficiently uncovers hierarchical community structures.

Building upon the Louvain method, the *Leiden* algorithm (Traag et al., 2019) introduces a refinement step between the two original Louvain phases. This step ensures that each community is internally connected, thus preventing fragmented clusters. Leiden also improves convergence speed and uses a more effective aggregation strategy that reduces the number of levels and overall runtime.

The *SimCMR* algorithm (Tunali, 2021) introduces a novel similarity-based approach to community detection. It fundamentally shifts focus from traditional node-node similarity metrics, instead operating through an iterative optimization framework centered on two uniquely defined similarity indices: node-community similarity (Snc) quantified as $\text{Snc}(v, C) = \frac{|N(v) \cap C|}{\sqrt{|C|}}$ where $N(v)$ denotes the neighbor set of node v and $|C|$ represents community size, and community-community similarity (Scc) defined as $\text{Scc}(C_i, C_j) = \frac{e(C_i, C_j)}{|C_i| \cdot |C_j|}$ with $e(C_i, C_j)$ counting edges between communities C_i and C_j. The algorithm initiates with singleton communities. It then iteratively reassigns nodes to maximize Snc and merges communities when Scc exceeds a dynamic threshold τ. Crucially, Leiden-inspired connectivity constraints are enforced during merging to prevent fragmentation. This process repeats until modularity gains fall below a convergence threshold ϵ.

Metaheuristic methods detect community structure by exploring the solution space in a systematic way without guaranteeing a global optimum. In contrast to heuristics that depend on problem-specific rules, metaheuristics offer general search frameworks inspired by natural processes. Examples include Genetic Algorithms (Holand, 1975), Particle Swarm Optimization (Kennedy and Eberhart, 1995), and Simulated Annealing (Kirkpatrick et al., 1983). There is extensive research on metaheuristic approaches for community detection (Bara'a et al., 2021).

Genetic Algorithms improve candidate partitions through selection, crossover, and mutation while a fitness function directs the search toward better solutions. Particle Swarm Optimization relies on collective movement patterns to update solution positions. Simulated Annealing uses controlled randomness to escape local optima and approach high-quality solutions. Hybrid and hierarchical variants extend these principles by combining metaheuristic search with recursive graph partitioning. Saoudi and Moussaoui introduce a hierarchical genetic approach (Saoud and Moussaoui, 2018) that applies evolutionary optimization at each bipartition to maximize modularity. The graph is recursively divided once an optimal split is found, producing a dendrogram that reflects the hierarchical organization of the network. The method is evaluated in terms of NMI and Modularity Q on small classical social-network datasets, such as Zachary's Karate Club, Dolphins, and American Football.

2.2.6 Dynamic Approaches

Dynamic Methods leverage simulated dynamic processes, particularly random walks or information flow, to uncover community structure by analyzing how entities interact and influence each other through iterative steps within the network.

Walktrap (Pons and Latapy, 2005) is a dynamic community detection method that uses random walks to capture the community structure. The idea is that short random walks tend to stay within the same community, and nodes visited in similar walks are likely to belong to the same group. The algorithm computes pairwise node similarities based on random walk distances and performs agglomerative clustering to uncover a hierarchical community structure.

Another dynamic approach is the *InfoMap* algorithm (Rosvall and Bergstrom, 2008), which applies the map equation of information theory to reveal the structure of the community by minimizing the Shannon entropy. The aim is to obtain an optimal partition without specifying the number of communities. A random walker either stays within its current community or moves to a neighboring community. The map equation measures the expected description length of this process:

$$L(M) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{k} p_{\circlearrowright}^{i} H\big(P^{i}\big). \tag{2.8}$$

Here, k is the number of communities, $H(\cdot)$ denotes Shannon entropy, and $q_{\curvearrowright} = \sum_{i=1}^{k} q_{i\curvearrowright}$ is the total probability of exiting any community (where $q_{i\curvearrowright}$ is the probability of leaving community i), and $p_{\circlearrowright}^{i}$ is the fraction of all node visits that occur within community i. The term $H(P^{i})$ measures the average code length for identifying nodes within community i, and the community-level entropy $H(Q)$, defined as:

$$H(Q) = -\sum_{i=1}^{k} \frac{q_{i\curvearrowright}}{\sum_{j=1}^{k} q_{j\curvearrowright}} \log\!\left(\frac{q_{i\curvearrowright}}{\sum_{j=1}^{k} q_{j\curvearrowright}}\right), \tag{2.9}$$

quantifies the average code length for community names. Minimizing $L(M)$ balances the cost of encoding movements between communities against the cost of encoding movements within them.

In practice, InfoMap builds transition probabilities via random walks, generating sequences of node visits. These sequences are encoded with a two-layer Huffman scheme: the first layer assigns a code to each community, and the second layer assigns codes to nodes within that community. By grouping sequences with similar prefixes, InfoMap encourages nodes with comparable random-walk behavior to cluster together. Iteration continues until no further reduction in the map equation is possible, yielding a partition that reflects the dynamic flow of information.

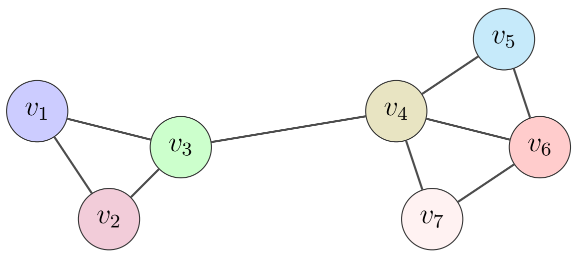
The Label Propagation Algorithm (LPA), introduced by Raghavan et al., represents a foundational approach to community detection that operates exclusively on network topology without requiring predefined parameters or optimization objectives, achieving remarkable efficiency with near-linear time complexity $O(M)$, where M is the number of edges, which enables scalable application to massive networks (Raghavan et al., 2007). The algorithm starts by assigning each node a unique label, then iteratively updates labels through asynchronous local propagation where nodes sequentially adopt the most frequent label among their neighbors resolving ties through random selection, causing tightly connected node clusters to rapidly converge toward consensus labels as labels propagate preferentially through dense regions, with the process terminating when no label changes occur and connected components sharing identical labels forming communities.The complete procedure is formalized in Algorithm 2, while Figure 2.5 illustrates its stepwise execution on a simple graph, demonstrating how nodes like v_4 adopt neighbors' labels (v_3) and v_6 aligns with v_5 during propagation until stabilization reveals distinct communities.

Despite these strengths, LPA suffers from well-documented limitations including outcome instability due to random tie-breaking (Raghavan et al., 2007), the "monster community" problem where dominant labels absorb disproportionate regions (Leung et al., 2009), oscillatory behavior in bipartite structures, resolution limits in sparse regions, and absence of an objective function for quality validation. These issues have spurred numerous enhancements: Deterministic variants, such as the *entropy-driven label propagation algorithm*, eliminate randomness through entropy-based rules instead of random selection (Chen et al., 2017)., while LPAm incorporates modularity maximization to avoid trivial solutions and improve community quality; edge-weighted approaches such as LPAc leverage edge clustering coefficients to prioritize connections within dense local neighborhoods, substantially improving detection accuracy in networks with heterogeneous community structures (Zhang et al., 2007); similarity-informed methods incorporate Jaccard similarity between neighbor sets to weight label influence, enhancing performance in networks with ambiguous community boundaries (Wang et al., 2017b); scalability-focused innovations like FLPA (Fast Label Propagation) achieve orders-of-magnitude speedups through queue-based processing of active nodes and optimized local moves, enabling real-time community detection in billion-edge graphs (Traag and Šubelj, 2023); finally, centrality-driven extensions exemplified by LPA-S integrate node centrality metrics and common neighbor analysis to prevent monster communities and stabilize results, particularly in scale-free networks where high-degree nodes dominate propagation (Douadi et al., 2024). This evolutionary trajectory demonstrates how modern variants preserve LPA's core advantages while systematically addressing fundamental limitations through topological heuristics and computational optimizations, solidifying label propagation's role as an adaptable framework for contemporary network science challenges.

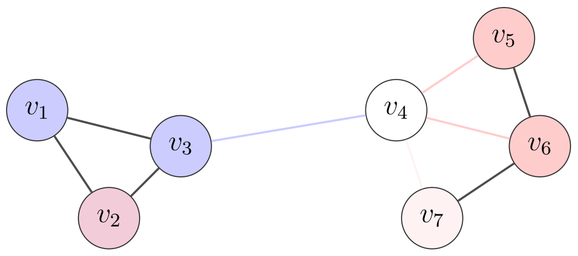
Experimental evaluations of LPA-S on standard community detection benchmarks (Karate Club, Football, and Dolphins) demonstrate its strong performance against established methods including Walktrap, Infomap, LPA, LPAC, FLPA, and Louvain. The results reveal that LPA-S consistently improves upon standard LPA, frequently matches or exceeds the performance of Walktrap and Infomap on NMI and F1 metrics, and achieves modularity scores competitive with Louvain, despite Louvain's specialized focus on modularity optimization.

Algorithm 2 Label Propagation Algorithm (LPA)

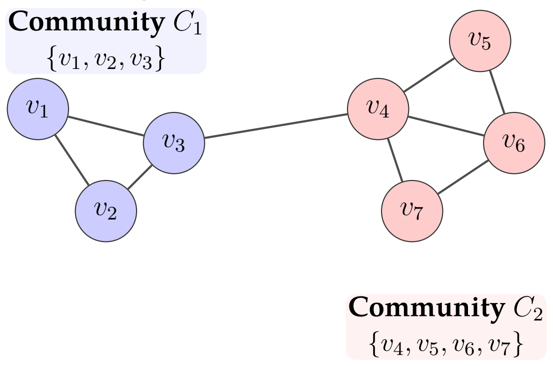
```
1: Input: Graph $G = (V, E)$
2: Output: Community assignments $C$ for all nodes
3: Initialize each node $v \in V$ with a unique label $C_v = v$
4: repeat
5:     Shuffle the node set $V$ in random order
6:     for all $v \in V$ do
7:         $\mathcal{M} \leftarrow \{C(u) \mid u \in N(v)\}$          ▷ multiset of neighbor labels
8:         Count frequency of each label in $\mathcal{M}$
9:         $C_{\text{new}} \leftarrow$ most frequent label in $\mathcal{M}$          ▷ break ties uniformly at random
10:        if $C_{\text{new}} \neq C(v)$ then
11:            $C(v) \leftarrow C_{\text{new}}$
12:        end if
13:    end for
14: until no label changes in an entire iteration
15: return $C$
```



(a) **Step 1: Initialization.** Each node is assigned a unique label (represented with distinct colors). At this stage, no information about communities is known, and nodes are equally likely to influence others.



(b) **Intermediate step.** More labels are added. Node v_4 adopts the label of its neighbor v_3 (blue), node v_7 (pink), and node v_6 aligns with v_5 (red), showing how label influence begins to spread across local neighborhoods.



(c) **Step 3: Label convergence.** After several passes, labels stabilize into two distinct communities. Nodes within each group share a common label.

Figure 2.5—Stepwise illustration of the Label Propagation Algorithm (LPA) on a small graph. The process begins with unique labels and proceeds iteratively until stable communities emerge.

2.3 Traditional Machine Learning–based Community Detection

Traditional machine learning–based methods for community detection have gained prominence due to their ability to uncover latent structure in complex networks without relying solely on heuristic or optimization-based rules. Using probabilistic models, matrix decompositions, and generative approaches originally developed for text and signal processing, these methods can flexibly model node attributes, topological patterns, and overlapping memberships. In this section, we survey three major classes of machine learning models for community detection: statistical inference methods, nonnegative matrix factorization, and topic models.

2.3.1 Statistical Inference Models

The Stochastic Block Model (SBM) (Holland et al., 1983; Abbe, 2017) provides a principled generative framework for networks with latent community structure. In this model, each node is assigned to one of k blocks (communities) via a hidden membership vector $\mathbf{z} \in \{1, \ldots, k\}^N$. Edge formation is governed by a symmetric $k \times k$ probability matrix $\mathbf{P}$, where the existence of an edge between nodes i and j follows an independent Bernoulli distribution:

$$\Pr(A_{ij} = 1 \mid z_i = c, z_j = d) = P_{cd}.$$

Here, $p_{\text{in}} = P_{cc}$ denotes the intra-block edge probability, while $p_{\text{out}} = P_{cd}$ $(c \neq d)$ represents the inter-block edge probability. By modulating the ratio $p_{\text{in}}/p_{\text{out}}$, the SBM generates graphs spanning from clearly delineated communities to near-random topologies, establishing it as a benchmark for community detection algorithms.

Parameter inference in SBM typically proceeds by maximizing the likelihood of the observed adjacency matrix $\mathbf{A}$ or by estimating the Bayesian posterior under priors on $\mathbf{z}$ and P. Common estimation methods include expectation-maximization (EM) (Daudin et al., 2008), variational Bayes, and Markov chain Monte Carlo (MCMC) (McDaid et al., 2013). Building on degree-corrected SBMs (DCSBM) that introduce node-specific parameters $\{\theta_i\}$ to model heavy-tailed degree distributions and thus better capture real-world networks' heterogeneity (Karrer and Newman, 2011), Serrano and Vidal (2024) deliver the first exact inference algorithms for maximum-likelihood community detection under DCSBM. They first cast the likelihood maximization as a mixed-integer non-linear program (MINLP) and then, via careful linearization of the log-terms, as a mixed-integer linear program (MILP). To tame the MILP's combinatorial blow-up, they integrate dynamic valid-inequality generation, symmetry-breaking constraints, and tight bounds on variables, alongside EM-based warm starts to accelerate convergence.

The Parameter-Weighted Betweenness-Clustering SBM (PBCSBM) (Wang et al., 2025) significantly enhances community detection in attributed networks by integrating topological node properties directly into the connectivity function. This approach extends degree-corrected SBMs through a weighted likelihood formulation $\lambda_{ij} = \theta_{z_i z_j} \cdot (\alpha d_i d_j + \beta b_i b_j + \eta c_i c_j)$, where betweenness centrality b_i and clustering coefficient c_i complement degree heterogeneity, with tunable parameters (α, β, η) balancing their contributions. On attributed networks (e.g. Cora), PBCSBM yields over 12% higher NMI than classical SBMs (Wang et al., 2025), especially for nodes with non-hub roles.

2.3.2 Nonnegative Matrix Factorization

Nonnegative matrix factorization (NMF), first introduced by Lee and Seung (1999), approximates a nonnegative data matrix as the product of two lower-rank nonnegative factors. This technique finds applications in diverse domains such as image analysis, text mining and network science. For graph-structured data, the adjacency matrix $\mathbf{A} \in \mathbb{R}_{\geq 0}^{n \times n}$ is factorized to uncover interpretable, parts-based latent structure by finding matrices $\mathbf{U} \in \mathbb{R}_{\geq 0}^{N \times k}$ and $\mathbf{P} \in \mathbb{R}_{\geq 0}^{k \times N}$, with $k \ll N$, such that $\mathbf{A} \approx \mathbf{UP}$.

Each row of $\mathbf{U}$ represents a node's nonnegative embedding in a k-dimensional latent space. In other words, $\mathbf{U}$ models the projection between the graph and the community-membership space. Each column of $\mathbf{P}$ provides reconstruction weights for adjacency relationships, indicating the probability of community membership for each node. To determine $\mathbf{U}$ and $\mathbf{P}$, one typically minimizes one of two commonly used objective functions: the Frobenius-norm reconstruction error (Equation 2.10) or the Kullback–Leibler divergence (Equation 2.11).

$$\mathcal{L}(\mathbf{U}, \mathbf{P}) = \tfrac{1}{2}\left\|\mathbf{A} - \mathbf{UP}\right\|_F^2 = \tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(A_{ij} - [\mathbf{UP}]_{ij}\right)^2, \tag{2.10}$$

$$D_{\mathrm{KL}}\left(\mathbf{A} \,\|\, \mathbf{UP}\right) = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(A_{ij} \log \tfrac{A_{ij}}{[\mathbf{UP}]_{ij}} - A_{ij} + [\mathbf{UP}]_{ij}\right). \tag{2.11}$$

Ding et al. (2005) establish a formal equivalence between symmetric NMF and two cornerstone clustering techniques, Kernel k-means and Laplacian-based spectral clustering. They demonstrate how non-negativity sharpens community boundaries. Building on this theoretical foundation, Pairwise-Constrained Symmetric NMF (PCSNMF) (Shi et al., 2015) injects partial supervision via must-link and cannot-link constraints derived from known labels, steering the factorization toward ground-truth relations. Wang et al. (2016) introduce the Semantic Community Integration (SCI) framework, which fuses adjacency and attribute matrices through a unified NMF-style loss to jointly capture topological and semantic information. Huang et al. (2020) propose a dynamic joint-factorization model that adaptively weights graph structure and node attributes within a unified NMF objective, yielding crisper, semantically coherent communities. Concurrently, SNCMF (Liu et al., 2022) enriches symmetric NMF with multiple latent factor matrices, a symmetry-alignment regularizer, and a Laplacian-based geometric term to preserve higher-order network structure often missed by single-matrix models.

In semi-supervised settings, methods diverge in how they weave side information into the core NMF decomposition. WSCDSM (unified Weakly Supervised framework for Community Detection and Semantic Matching) (Wang et al., 2018b) uncovers communities by jointly factorizing the adjacency and membership matrices and dynamically incorporating node attributes or pairwise hints. By contrast, PSSNMTF (Semi-Supervised Non-negative Matrix Tri-Factorization with node Popularity) (Jin et al., 2021) employs an NMF tri-factorization that explicitly models node-popularity effects alongside structural and attribute information. Each approach exemplifies a distinct strategy for augmenting classical NMF with auxiliary signals to improve both accuracy and interpretability.

2.3.3 Topic Modeling

Topic modeling seeks to uncover the latent thematic structure of document collections by assuming that each document arises from a mixture of topics, where each topic is itself a probability distribution over words. Early probabilistic approaches include Probabilistic Latent Semantic Analysis (PLSA), which models word–document co-occurrences via mixture components and employs an expectation–maximization procedure for parameter estimation (Hofmann, 1999), and Latent Dirichlet Allocation (LDA), which introduces Dirichlet priors over per-document topic distributions and per-topic word distributions to yield a fully generative Bayesian model, typically learned via collapsed Gibbs sampling or variational inference (Blei et al., 2003). Despite its widespread use, LDA requires the number of topics to be specified a priori; this limitation is addressed by nonparametric Bayesian extensions such as the Hierarchical Dirichlet Process (HDP), which infers an unbounded number of topics through a stick-breaking construction and Chinese Restaurant Franchise metaphor (Teh et al., 2006).

Recent work shows that community detection and network embedding approaches can be powerfully derived by treating networks as "documents" and structural elements (nodes, edges, walks) as "words," leveraging topic-modeling analogies to uncover communities. Zhang et al. (2007) pioneered this direction by applying standard LDA to large-scale social graphs by treating each node's ego-network as a document to discover latent community memberships through word-topic assignments. Yin et al. extended this with their Latent Community Topic Analysis (LCTA), a unified probabilistic framework that jointly infers per-node topic mixtures and network structures, enhancing community coherence and link-prediction accuracy (Yin et al., 2012). Cha and Cho demonstrated this mapping's utility for information retrieval, using topic modeling on social-network subgraphs to improve document ranking at SIGIR (Cha and Cho, 2012). More recently, Chowdhury et al. applied community-detection algorithms to word-association graphs built from corpora, extracting semantically coherent term clusters with gains in interpretability and downstream NLP performance (Chowdhury et al., 2023). Building on this, researchers combined Local Search Smart Local Moving (LS-SLM) and Probabilistic Community Coherence-based Latent Dirichlet Allocation (PCC-LDA) to perform community detection on user-article interaction graphs, thereby framing recommendations as a joint topic-network modeling problem (Rachamadugu and Pushphavathi, 2025).

2.3.4 Representation Learning—Based Approaches

The use of representation learning techniques to discover communities has gained popularity due to their ability to capture structural and proximity-based patterns in graphs. Most methods in this category learn low-dimensional *node embeddings* that require additional clustering techniques (e.g., K-means (Krishna and Murty, 1999) or GMM (Rasmussen, 1999)) for community detection. Several approaches introduced earlier in Section 1.4.1.2, such as DeepWalk (Perozzi et al., 2014), Node2Vec (Grover and Leskovec, 2016), and Struc2Vec (Ribeiro et al., 2017), follow this two-step paradigm. In contrast, Community2Vec (Martin, 2017) focuses on embedding *entire communities* rather than individual nodes, though it typically requires pre-detected communities as input. Significantly different is ComE (Community Embedding) (Cavallari et al., 2017), which uniquely performs *joint embedding and community detection* through a unified optimization framework that directly outputs communities without requiring separate clustering steps. While initially designed for general graph representation, these methods have shown strong performance when adapted for community detection tasks. Experimental evaluations in the original ComE paper (Cavallari et al., 2017) demonstrate that the framework outperforms established baseline methods including LINE, Node2Vec, and DeepWalk on social network benchmarks. The authors employ Gaussian Mixture Models (GMM) (Rasmussen, 1999) to directly fit communities over the learned embeddings, leveraging this probabilistic approach to derive final community assignments.

2.4 Deep Learning–based Community Detection

Deep learning–based approaches have revolutionized community detection by learning high-capacity, hierarchical representations of nodes and edges directly from data. These methods leverage convolutional, attention, and generative architectures to capture complex structural patterns and attribute information, often outperforming traditional algorithms on large, noisy networks. In the following subsections, we review five key families of deep models for community discovery.

2.4.1 Convolutional Networks

In this section, we first review the classical Convolutional Neural Network paradigm, which was originally formulated to exploit the local spatial structure of grid-based data via shared convolutional filters and hierarchical feature extraction (LeCun et al., 1998). We then turn to Graph Convolutional Networks (GCNs), which extend the notion of convolution to irregular, non-Euclidean domains by leveraging spectral or spatial formulations of graph signal processing (Bruna et al., 2014; Kipf and Welling, 2017). Together, these two architectural paradigms illustrate how convolutional architectures can be adapted from regular grids to arbitrary graph topologies, enabling powerful representations for tasks such as image recognition, graph classification, and of particular relevance here, community detection in networks.

2.4.1.1 Convolutional Neural Network

CNNs are deep learning architectures originally designed for grid-structured data, such as images, text, and audio. Their strength lies in capturing local spatial or sequential patterns through shared convolutional filters, making them highly effective in domains like computer vision and natural language processing. The general CNN architecture is detailed in Section 1.3.2.

Although CNNs are traditionally applied to grid-like data, their principles have been extended to handle more complex, non-Euclidean structures such as graphs. In the context of network analysis, CNN-based methods have been adapted to learn from graph-structured data, enabling powerful approaches to tasks such as community detection, link prediction, and node classification. These adaptations preserve the core strengths of CNNs while allowing them to operate effectively on irregular topologies typical of real-world networks. Because graphs are inherently non-Euclidean, these methods must first transform graph data into a Euclidean form suitable for convolutional operations. Accordingly, we categorize existing CNN-based community detection methods into two broad transformation approaches, Node-based Transformation Approach and Edge-based Transformation Approach. This taxonomy will serve as the basis for the comparative study in Section 3.2, where we will analyze and benchmark representative methods from each category.

In the *Node-based Transformation Approach*, the community detection task is cast as a node classification problem: nodes sharing the same label are grouped into the same community. Rather than relying solely on the adjacency matrix, which marks direct connections with 1 and non-connections with 0, each target node v is enriched by encoding its multi-hop neighborhood. The s-hop proximity between v and n is defined by Equation 2.12, where the hop count s runs from 1 up to S_0, a user-defined threshold, and $\sigma \in (0, 1)$ is an attenuation factor that down-weights more distant neighbors (Xin et al., 2017). This encoding highlights the local structural characteristics of each node. Incorporating multi-hop proximities can enhance detection accuracy by capturing indirect adjacency relations, but raising the hop-count threshold S_0 beyond an optimal range may introduce noise and blur community boundaries. Figure 2.6 provides an illustrative example of the input details. Consequently, S_0 must be selected empirically, through dataset-specific ablation studies and validation experiments to balance the benefit of additional contextual information against the risk of degrading community separability.

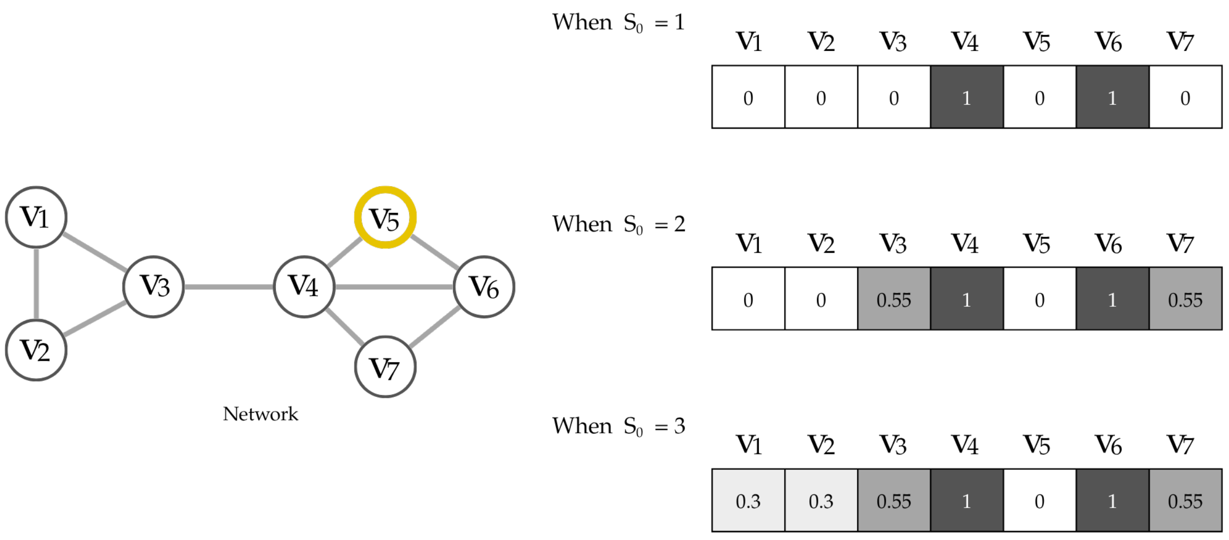


Figure 2.6—Illustrative example of information about the input data for node $V_5(\sigma = 0.6)$.

$$Entry = e^{\sigma(1-S)} \tag{2.12}$$

To leverage standard 2D convolutional kernels, methods in this category (Xin et al., 2017; Santo et al., 2021; Sperlí, 2019) reshape each node's feature vector of length N into a 2D array of dimensions $w \times h$, with $w\,h = N$. This mapping preserves all original adjacency relations while organizing the entries into a matrix on which convolutional filters can operate. As a result, each of the N nodes is represented by its own $w \times h$ matrix, enabling the direct application of off-the-shelf CNN architectures.

Xin et al. (2017) tackle the challenge of finding communities when a substantial fraction of edges is missing, the so-called topologically incomplete networks (TINs), they propose that CNN filters capture higher-order connectivity patterns, with deeper layers aggregating increasingly broad structural contexts. The model reinterprets each node's local adjacency pattern as a two-dimensional "image" and applies a sequence of convolutional and pooling operations to extract increasingly rich structural features.

In the first convolutional layer, c_1 learnable kernels operate over the node's adjacency submatrix, each kernel **W** together with its bias b_W generates a corresponding feature map entry:

$$v_{n,xy} = \sigma\Big(b_W + \sum_{i=0}^{w'-1}\sum_{j=0}^{h'-1} W_{ij}\, p_{n,(x+i)(y+j)}\Big), \tag{2.13}$$

where $p_{n,(x+i)(y+j)}$ denotes the presence or absence of edges in the local patch of node n and σ is the sigmoid activation. A max-pooling operation of size $m_1 \times m_2$ then subsamples each feature map by retaining the maximum value within non-overlapping regions, reducing resolution while preserving salient patterns. The second convolutional layer repeats this process with c_2 kernels on each downsampled map, yielding a total of $c_1 \times c_2$ feature maps that capture higher-order connectivity motifs.

The feature maps from the second convolutional layer, with spatial dimensions $f_1/m_1 \times f_2/m_2$ (where f_1 and f_2 are the original feature map dimensions from the first convolution, reduced by max-pooling factors m_1 and m_2), are flattened into a vector $\mathbf{q}_n \in \mathbb{R}^{c_1 c_2\, f_1/m_1\, f_2/m_2}$ and passed to a fully connected output layer with K neurons (one per community) whose activations are defined as:

$$o_n^k = \sigma\big(b_f^k + \mathbf{W}_f^k\, \mathbf{q}_n\big) \tag{2.14}$$

represent the confidence that node n belongs to community k. During training, the parameters $P = \{\mathbf{W}, \mathbf{W}_f, b, b_f\}$ are optimized by minimizing the mean squared error:

$$J(P) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{k=1}^{K}\big(o_n^k - \ell_n^k\big)^2, \tag{2.15}$$

where $\ell_n^k \in \{0, 1\}$ is the ground-truth indicator of membership. Standard back-propagation adjusts all convolutional kernels and output weights until convergence.

The model shows good performance in terms of NMI on both synthetic and real-world datasets, particularly under Topologically Incomplete Network (TIN) scenarios simulated through controlled edge removal or sparsity variations, and in supervised settings with varying label rates, surpassing traditional machine-learning community-detection baselines.

Sparsity is an ubiquitous property of real-world graphs, particularly social networks, and some methods exploit this by operating directly on the non-zero entries of the adjacency matrix. By considering only active connections, these approaches dramatically reduce both memory requirements and computational cost (Santo et al., 2021; Sperlí, 2019). For example, Santo et al. propose a semi-supervised CNN architecture that classifies nodes into K categories on sparse adjacency patches. Their method replaces dense convolutions and pooling with *SparseConv2D* and sparse *max-pooling* layers, which traverse only active edges stored in compressed sparse formats. By focusing computation on existing connections, this design achieves orders of magnitude efficiency gains while maintaining classification accuracy.

In the *Edge-based Transformation Approach*, community detection is reframed as an edge classification task using a convolutional neural network. To leverage CNNs, Cai et al. (2020) address this with their ComNet-R model, which proceeds in three stages. Each edge (u, v) in the orginal graph $\mathcal{G}$ is first transformed via an edge-2-image model into a small 2D colored image encoding local connectivity in its RGB (Red, Green, Blue) channels. Next, a supervised CNN binary classifier labels each image as either an intra-community or inter-community edge. After removing inter-community edges, the simplified graph $\mathcal{G}_2$ emerges, whose connected components correspond to candidate community fragments. A hierarchical merging module then applies a local modularity metric R to separate high-modularity components (preliminary communities) from temporary ones. Finally, these components are merged through an agglomerative procedure guided by the R metric to produce exactly K well-formed communities. This pipeline exploits CNN pattern recognition on locally encoded edge images while retaining the rigor of modularity-based community assembly.

ComNet was evaluated using NMI and F1 on both synthetic (LFR) and real-world networks of varying scales, demonstrating superior performance compared to baseline methods, including Louvain.

2.4.1.2 Graph Convolutional Network

Graph Convolutional Networks (GCNs), introduced earlier in Section 1.4.3.1, have become a cornerstone in graph-based learning. In the context of community detection, GCN-based methods can be broadly categorized into two main groups: (i) community classification approaches, which operate in a supervised or semi-supervised learning setting, leveraging labeled nodes to infer community membership; and (ii) unsupervised network representation methods, which aim to learn informative node embeddings that capture the structural and attribute-based patterns in the graph, typically followed by a separate clustering step to detect communities.

CayleyNets (Levie et al., 2019) introduce a specialized Cayley-polynomial spectral filter that, like its Chebyshev counterpart, achieves strong localization while retaining linear complexity in the number of edges. This enables the model to isolate narrow frequency bands relevant to tasks in the graph spectrum, crucial for distinguishing communities, before feeding these high-order spectral features into a semi-supervised softmax layer for classification.

The *Line Graph Neural Network* (*LGNN*) (Chen et al., 2019) enhances traditional Stochastic Block Models (SBMs) by improving community detection performance while reducing computational cost, particularly in directed networks. This is achieved by operating on the non-backtracking line graph, which treats edges as nodes and prevents path overcounting. LGNN applies belief-propagation inspired message-passing rules on this line graph to propagate and aggregate information, learning rich node representations via a supervised cross-entropy loss. The model seamlessly handling both binary and multiclass community detection.

In the unsupervised setting, Neural Overlapping Community Detection (NOCD) (Shchur and Günnemann, 2019) formulates community detection as a probabilistic inference task, integrating graph representation learning with statistical modeling through two GCNs layers. The model processes an adjacency matrix $\mathbf{A}$ and node features $\mathbf{X}$ to generate node embeddings $\mathbf{Z}$ via the transformation:

$$\mathbf{Z} = \mathrm{ReLU}\left(\hat{\mathbf{A}} \cdot \mathrm{ReLU}\left(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}_1\right)\mathbf{W}_2\right),$$

where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}$ denotes the normalized adjacency matrix with self-loops and $\tilde{\mathbf{D}}$ is the corresponding degree matrix. These embeddings are mapped to community affiliation probabilities $\mathbf{F} \in [0,1]^{n\times c}$ through a sigmoid activation $\mathbf{F} = \sigma(\mathbf{Z}\mathbf{W}_f)$ with weight matrix $\mathbf{W}_f \in \mathbb{R}^{k\times c}$ parameterizing c communities.

The edge generation process follows a Bernoulli-Poisson distribution where $\mathbf{A}_{uv} \sim \mathrm{Bernoulli}\left(1 - \exp\left(-\mathbf{f}_u^\top \mathbf{f}_v\right)\right)$, with optimization minimizing the regularized negative log-likelihood

$$\mathcal{L}(\mathbf{F}) = -\mathbb{E}_{(u,v)\sim P_E}\left[\log\left(1 - \exp\left(-\mathbf{f}_u^\top \mathbf{f}_v\right)\right)\right] + \mathbb{E}_{(u,v)\sim P_N}\left[\mathbf{f}_u^\top \mathbf{f}_v\right].$$

Here $\mathbf{f}_u \in \mathbb{R}^c$ denotes node u's community affiliation vector, while $\mathbb{E}_{(u,v)\sim P_E}$ and $\mathbb{E}_{(u,v)\sim P_N}$ represent expectations over positive edges sampled from the edge set $\mathcal{E}$ and negative edges from non-connections, respectively. The first term maximizes the likelihood of observed edges through the inner product $\mathbf{f}_u^\top \mathbf{f}_v$, while the second term penalizes spurious affiliations for non-adjacent node pairs, collectively enforcing a sparse community structure that naturally accommodates overlapping memberships.

The *Spectral Embedding Network* (*SENet*) (Zhang et al., 2021) begins by computing a shared-neighbor similarity matrix $\hat{\mathbf{S}}$ and fusing it with the original adjacency $\mathbf{A}$ to form an enhanced graph $\hat{\mathbf{A}}$, then instantiates a three-layer encoder (weights $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$) that propagates node features over $\hat{\mathbf{A}}$ to produce embeddings $\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, \mathbf{Z}^{(3)}$. A p-nearest-neighbor kernel $\mathbf{K}$ is derived from a learned kernel matrix and symmetrized, and the entire model is trained end-to-end by minimizing the smoothness objective equation:

$$\min_{\mathbf{W}_1,\mathbf{W}_2,\mathbf{W}_3} J = -\operatorname{tr}\Big((\mathbf{Z}^{(3)})^\top \mathbf{D}^{-\frac{1}{2}} \mathbf{K} \mathbf{D}^{-\frac{1}{2}} \mathbf{Z}^{(3)}\Big), \tag{2.16}$$

where tr is the trace and $\mathbf{D}$ is the degree matrix of $\mathbf{K}$. After optimization, the three layer-wise embeddings are concatenated into a final representation $\mathbf{Z}$, and k-means on $\mathbf{Z}$ yields the cluster partition C, effectively capturing both local and global structure through spectral regularization.

Community Deep Graph Infomax (CommDGI) (Zhang et al., 2020a) projects node embeddings $\mathbf{Z}$ via a GCN encoder into a soft affiliation matrix $\mathbf{F} = \sigma(\mathbf{Z}\,\mathbf{W}_f) \in [0,1]^{N\times k}$, where each row $\mathbf{f}_i$ encodes node–community affinities. It is trained with a composite objective function:

$$\mathcal{L} = \alpha\,\mathcal{L}_{\text{graph}} + \beta\,\mathcal{L}_{\text{comm}} + \mathcal{L}_{\text{mod}}, \tag{2.17}$$

where the graph mutual-information loss $\mathcal{L}_{\text{graph}}$ uses a noise-contrastive binary cross-entropy between samples from the joint distribution (positive examples) and samples from the product of marginals (negative examples). Specifically, letting N and M denote the numbers of positive and negative samples, respectively, and denoting corrupted features and adjacency by $\tilde{\mathbf{X}}$, $\tilde{\mathbf{A}}$, we define:

$$\mathcal{L}_{\text{graph}} = \frac{1}{N+M}\left[\sum_{i=1}^{N} \mathbb{E}_{\mathbf{X},\mathbf{A}}\big[\log D(\mathbf{z}_i, s)\big] + \sum_{j=1}^{M} \mathbb{E}_{\tilde{\mathbf{X}},\tilde{\mathbf{A}}}\big[\log\big(1 - D(\tilde{\mathbf{z}}_j, s)\big)\big]\right], \tag{2.18}$$

where D is the discriminator scoring agreement between a node embedding $\mathbf{z}_i$ (row of $\mathbf{Z}$) and the community summary s. The community cohesion loss $\mathcal{L}_{\text{comm}}$ encourages high affinity within each community and low affinity across communities, and $\mathcal{L}_{\text{mod}}$ is a modularity-based regularizer that promotes globally coherent partitions. After training, CommDGI applies soft k-means to the rows of $\mathbf{F}$ to obtain overlapping community assignments, thereby uniting contrastive representation learning with principled clustering.

Experimental evaluations reveal diverse testing paradigms across these GCN-based approaches. LGNN tested on synthetic and social network benchmarks, achieving high accuracy in supervised community assignment tasks. CayleyNets shows robust results across multiple experimental settings, including citation networks like Cora in semi-supervised learning scenarios. For overlapping community detection, NOCD specializes in social networks with inherent community overlaps. In the unsupervised domain, comparative analysis of SENet and CommDGI on standard citation benchmarks reveals complementary strengths: SENet achieves superior accuracy, suggesting precise node-level label assignment, while CommDGI attains higher NMI and F1 scores, indicating tighter community separability in the learned embedding space.

2.4.2 Graph Attention Networks

In Section 1.4.3.2, we introduced the Graph Attention Network (GAT) as a powerful extension of graph convolutional architectures that replaces fixed, uniform neighborhood aggregation with a learnable, node-wise attention mechanism. This adaptability is especially well suited to community detection, where the strength of associations between nodes can vary dramatically depending on local subgraph structure and attribute similarity. By learning to emphasize edges that are more indicative of shared community membership, rather than treating all neighbors equally, GAT naturally highlights intra-community connections while attenuating noisy inter-community links.

Several Graph Attention Network models have been proposed for community discovery in complex networks. One notable example is Deep Multiplex Graph Infomax (DMGI) (Park et al., 2020), which is tailored to attributed multiplex networks. Such networks are defined as $\mathcal{G} = \{V, E^{(1)}, \ldots, E^{(L)}, \mathbf{X}\}$, where V is a common node set, each $E^{(l)}$ captures a distinct relation layer, and $\mathbf{X}$ contains node attributes. DMGI learns embeddings in an unsupervised manner by combining node features with multi-relational structure. First, it applies contrastive learning across each layer by distinguishing real graphs from feature-shuffled corruptions with a discriminator. Then, it enforces a consensus among layer-specific embeddings through attention coefficients that dynamically weight each layer's contribution based on both its individual embedding and a cross-layer context vector. This attention-guided fusion not only resolves inter-layer heterogeneity but also ensures consistency across views, yielding unified node representations that capture both structural and attribute information.

High-order Deep Multiplex Infomax (HDMI) (Jing et al., 2021) extends DMGI by capturing both global contrastive signals and intrinsic attribute–structure relationships in multiplex networks. Although DMGI aligns layer-specific embeddings using a shared contrastive objective, it overlooks how node attributes shape higher-order connectivity. HDMI remedies this by augmenting each relation layer with attribute-conditioned edges that reflect similarities in node features. Embeddings from the original and enriched graphs are then combined through a semantic-attention fusion module that weights each layer according to its inferred community structure.

Another approach, GATFELPA (Graph Attention Network Fused with Enhanced Label Propagation Algorithm) (Tang et al., 2025), integrates graph attention networks and an enhanced label propagation algorithm to improve the community detection task. GATFELPA leverages Graph Attention Networks (GATs) within a Similarity Preservation Module to learn rich, context-aware node representations. This module specifically aims to ensure that the learned embeddings maintain important similarity relationships from the original graph structure and node attributes. By employing an attention mechanism, the GAT in this module dynamically weighs the importance of different neighbors during information aggregation, allowing GATFELPA to capture nuanced relationships and determine the relevance of neighboring nodes for a node's community membership. The objective of the Similarity Preservation Module is to generate embeddings where structurally or attibute-wise similar nodes are close in the embedding space, which is crucial for effective community detection. Subsequently, an enhanced Label Propagation Algorithm (LPA) is applied to these attention-aware and similarity-preserved node embeddings. The enhanced LPA iteratively propagates labels among nodes, with the high-quality, GAT-derived embeddings guiding the propagation process more effectively to achieve accurate and robust community assignments.

DMGI and its extension HDMI, evaluated on the same attributed multiplex datasets, exhibit comparable performance, with HDMI showing improved NMI, whereas GATFELPA, as a recent work, demonstrates high performance across ACC, NMI, ARI, and F1 on attributed networks.

2.4.3 Graph Autoencoders

Graph Autoencoders (GAEs) provide a natural extension of deep learning to graph-structured data by learning compact node representations that preserve the underlying topology. As detailed in Section 1.4.3.6, GAEs capture both local and global graph structure in their latent space, making them particularly well suited for community detection. Owing to their strong representational capacity and empirical success, GAEs have become a cornerstone in this thesis: we devote substantial attention to GAE-based architectures and demonstrate how their versatility underpins many community-detection methods.

In this section, we propose a taxonomy of GAE-based community detection methods that highlights two main families. Simple Encoder Approaches rely on a single graph encoder to produce embeddings. Dual Encoder Approaches employ two complementary encoders. In Section 3.3, we will present a comparative study of representative methods of each category, evaluating their performance on benchmark networks.

Simple encoder approaches employ a basic two-stage architecture. In what follows, we organize these methods according to their backbone architectures. First, we review GCN-based GAEs and highlight several notable contributions; then we turn to GAT-based GAEs approaches and discuss their key innovations.

Graph AutoEncoders (GAEs) (Thomas and Welling, 2016) are unsupervised models that learn fixed-point node embeddings by using a graph convolutional encoder to project node features and structure into a low-dimensional latent space, and then reconstruct the adjacency matrix via an inner-product decoder. *Variational Graph AutoEncoders* (VGAEs) (Thomas and Welling, 2016) augment this architecture by having the encoder output a mean and variance for each node's latent vector, sampling from that Gaussian to perform reconstruction, and regularizing the inferred posterior toward a Gaussian prior using the Kullback–Leibler (KL) divergence.

Adversarially Regularized Graph Autoencoder (ARGA) (Pan et al., 2018) regularizes the deterministic latent space of a GAE. It employs a GCN encoder and an inner product decoder for graph reconstruction, while a discriminator network is simultaneously trained to distinguish the encoder's latent embeddings from samples drawn from a predefined prior distribution. The encoder, acting as a generator, is then optimized to produce embeddings that fool the discriminator, thereby implicitly forcing the latent space to conform to the desired prior. ARGA's objective combines a graph reconstruction loss with this adversarial regularization:

$$\mathcal{L}_{\text{adv}} = -\tfrac{1}{2}\,\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}\big[\log D(\mathbf{Z})\big] - \tfrac{1}{2}\,\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\big[\log\big(1 - D\big(G(\mathbf{X},\mathbf{A})\big)\big)\big]. \tag{2.19}$$

$$\mathcal{L}_{\text{ARGA}} = \mathcal{L}_{\text{rec}} + \min_{\Theta_g}\max_{\Theta_d}\ \mathcal{L}_{\text{adv}}. \tag{2.20}$$

Here, $p_{\mathbf{z}}$ denotes the prior distribution over latent samples and p_{data} denotes the empirical data distribution. The discriminator $D(\mathbf{Z})$ attempts to distinguish latent samples drawn from the prior from those generated by the encoder, while the generator $G(\mathbf{X}, \mathbf{A})$ (the encoder) seeks to produce embeddings that can fool the discriminator.

The *Adversarially Regularized Variational Graph Autoencoder* (ARVGA) (Pan et al., 2018) is the variational extension of ARGA, which applies adversarial regularization to the probabilistic latent space of a VGAE. Its GCN encoder outputs parameters for a posterior distribution (mean and variance) over latent embeddings, from which samples are drawn via the reparameterization trick. A discriminator then ensures that the distribution of these sampled latent codes aligns with a specified prior, often replacing or complementing the traditional Kullback-Leibler (KL) divergence term found in VGAEs. ARVGA's objective combines reconstruction loss, potentially a KL divergence term, and the adversarial loss:

$$\mathcal{L}_{\text{ARVGA}} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\left[\log p(\mathbf{A}|\mathbf{Z})\right] - KL[q(\mathbf{Z}|\mathbf{X},\mathbf{A}) \,\|\, p(\mathbf{Z})] + \min_{\Theta_g} \max_{\Theta_d} \mathcal{L}_{\text{adv}}. \tag{2.21}$$

The *Marginalized Graph Autoencoder* (MGAE) (Wang et al., 2017a) for graph clustering functions as a specialized denoising GCN-based autoencoder, a crucial consideration for real-world attributed networks that often contain inherent noise. A key mechanism within MGAE involves explicitly injecting noise into the node attributes through η rounds of random feature removal, which systematically generates corrupted attribute matrices. The model then leverages the concept of marginalization by taking an expectation over these η distinct corruptions. This approach aims to enhance robustness to attribute noise, as the model is trained to reconstruct the clean input regardless of the specific noise instance encountered during a forward pass. The training of MGAE is thus formulated by an objective function that embodies both the denoising and marginalization principles:

$$L = \frac{1}{\eta}\sum_{i=1}^{\eta}\left\|\mathbf{X} - \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{X}}\mathbf{W}\right\|^2 + \lambda\|\mathbf{W}\|_F^2. \tag{2.22}$$

Here, the term $\tilde{\mathbf{X}}$ denotes the corrupted attribute matrix within the loss function, directly reflecting the denoising autoencoder aspect as the model aims to reconstruct the clean input from this noisy representation. $\mathbf{W}$ represents the learnable parameters of the attribute transformation. The outer sum $\frac{1}{\eta}\sum_{i=1}^{\eta}$ explicitly implements the marginalization concept by averaging the reconstruction loss over multiple noisy samples. Furthermore, λ is a hyperparameter controlling the strength of $\|\mathbf{W}\|_F^2$, which serves as a regularization term for $\mathbf{W}$. Following this rigorous denoising and embedding process, community memberships are ultimately derived by applying spectral clustering to the final node embeddings, leveraging their enhanced quality for improved graph partitioning. The MAGE model is benchmarked on attributed networks and demonstrates superior performance compared to ARGA and ARVGA across a comprehensive set of metrics, including NMI, ARI, F1-score, Precision, and Recall. The *Embedding Graph Auto-Encoder* (EGAE) (Zhang et al., 2023) builds on the observation that a relaxed k-means objective over inner-product similarities recovers the true partition whenever embeddings lie on a normalized hypersphere and clusters correspond to mutually orthogonal subspaces. To leverage this, EGAE employs a dual-decoder architecture: one decoder reconstructs the adjacency matrix via $\hat{\mathbf{Z}} = \mathbf{Z}\mathbf{Z}^\top$, where $\mathbf{Z}$ is the learned embedding matrix, while a second "clustering decoder" interprets the relaxed k-means loss $J_c = \mathrm{tr}(\mathbf{Z}\mathbf{Z}^\top) - \mathrm{tr}(P^\top\mathbf{Z}\mathbf{Z}^\top\mathbf{P})$ as an additional reconstruction objective that encourages embeddings to form tight, well-separated clusters. Here, P denotes the continuous indicator matrix. Rather than decoupling embedding and clustering into separate stages, the model minimizes a joint loss:

$$J = J_r + \alpha\, J_c, \tag{2.23}$$

where J_r is the standard reconstruction loss and α balances the two criteria. The encoder is a GCN that produces nonnegative, unit-norm embeddings, satisfying the orthogonality conditions of the theoretical analysis. Training alternates between gradient-based updates of the GCN weights and closed-form or relaxed updates of the soft-assignment matrix $\mathbf{P}$. By embedding the clustering objective directly into the autoencoder, EGAE yields node representations that are immediately amenable to inner-product k-means.

He et al. (2022) introduce the Semi-Supervised Graph Convolutional AutoEncoder (SSGCAE) to uncover overlapping communities in attributed graphs by combining three complementary objectives. First, a GCN encoder projects input data into a latent space from which an inner-product decoder reconstructs the adjacency matrix; second, a modularity-based loss steers embeddings toward partitions with strong community structure; third, a cross-entropy term incorporates labels from a small set of nodes to provide semi-supervised guidance. After training, each node's soft membership vector is binarized via a simple thresholding step to produce overlapping community membership assignments. Empirical results on noisy attributed networks show that SSGCAE can reliably recover meaningful overlapping communities, and the authors propose as future work the modeling of semantic communities to enrich attribute information and capture more nuanced structures. Instead of employing traditional GCNs, the *graph attention autoencoder* (Salehi and Davulcu, 2020) integrates self-attention mechanisms within its stacked encoder and decoder layers to jointly reconstruct node features and the graph structure. In this model, the encoder utilizes attention to intelligently aggregate information from a node's neighbors, producing embeddings that capture both local and global structural characteristics. Subsequently, the decoder reconstructs the original node features and promotes similarity between connected nodes within the embedding space. The training objective combines two essential components: a feature reconstruction loss, quantified as the squared difference between the original and reconstructed node features, and a structural loss designed to encourage embedding similarity among neighboring nodes. The complete loss function is defined as:

$$\mathcal{L} = \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 - \lambda \sum_{j \in \mathcal{N}_i} \log\left(1 + \exp(1 - \mathbf{z}_i^\top \mathbf{z}_j)\right), \tag{2.24}$$

where λ serves as a balancing hyperparameter for the influence of the two terms. This model has been rigorously evaluated on widely used citation network datasets, including Cora, CiteSeer, and PubMed, demonstrating superior accuracy compared to VGAE and GAE in its experimental evaluation (Salehi and Davulcu, 2020).

Building upon the principles of deep graph autoencoders, *Deep Attentional Embedded Graph Clustering* (DAEGC) (Wang et al., 2019) utilizes attention mechanisms, specifically in the form of GATs, integrated within a graph autoencoder model. The DAEGC architecture leverages a GAT encoder with a self-attention mechanism to prioritize nodes within a neighborhood, thereby enhancing graph clustering by incorporating high-order neighbor information. This encoder is coupled with an inner product decoder, formally expressed as $\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^\top)$, where σ denotes the sigmoid activation function that maps the inner products to probabilities in $[0, 1]$, thereby reconstructing the adjacency matrix. This approach proves particularly effective for attributed graphs, embedding nodes into a low-dimensional vector space that adeptly captures both structural and feature information.

The model's parameters are optimized by minimizing a comprehensive loss function.

This function simultaneously incorporates two primary objectives: a reconstruction loss and a self-clustering loss based on Kullback-Leibler (KL) divergence:

$$\mathcal{L}_{total} = \mathcal{L}_{rec} + \beta \mathcal{L}_{clus} \tag{2.25}$$

Here, β is a hyperparameter. $\mathcal{L}_{rec}$ is the reconstruction loss, which measures the error between the original adjacency matrix $\mathbf{A}$ and its reconstruction $\hat{\mathbf{A}}$. $\mathcal{L}_{clus}$ is the self-clustering loss, which leverages a self-training mechanism fundamentally relying on Kullback-Leibler (KL) divergence. This loss quantifies the disparity between an auxiliary target distribution P (representing desired, sharpened cluster assignments) and the soft cluster assignment distribution Q (derived from the current node embeddings).

The soft assignment q_{ij} of node i to cluster j is commonly computed using a Student's t-distribution as a kernel, reflecting the similarity between a node's embedding $\mathbf{z}_i$ and a cluster centroid $\boldsymbol{\mu}_j$:

$$q_{ij} = \frac{\left(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_j\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{k=1}^{K}\left(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_k\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}, \tag{2.26}$$

where α represents the degrees of freedom (typically set to 1, effectively making it a Cauchy kernel), and K denotes the total number of clusters.

Subsequently, the target distribution p_{ij} is computed to refine these soft assignments. This step emphasizes high-confidence predictions and normalizes cluster frequencies, thus "sharpening" the cluster structure derived from Q and making the training process more robust to initial assignments:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{k=1}^{K} (q_{ik}^2 / \sum_i q_{ik})} \tag{2.27}$$

The self-clustering loss then rigorously minimizes the Kullback-Leibler divergence between this refined target distribution P and the soft assignment distribution Q:

$$\mathcal{L}_{clus} = D_{KL}(P||Q) = \sum_{i=1}^{N} \sum_{j=1}^{K} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2.28}$$

Empirically validated on attributed network datasets such as Cora and CiteSeer, DAEGC consistently demonstrates superior clustering accuracy, ARI, and NMI when compared against foundational models like Graph Autoencoders (GAE) and Variational Graph Autoencoders (VGAE). Notwithstanding these advancements, its sensitivity to crucial hyperparameters, such as embedding dimensionality, remains a notable challenge.

Building upon the advancements of deep graph autoencoders for community detection, the Community Detection based on Unsupervised Attributed Network Embedding (CDBNE) (Zhou et al., 2023) model is proposed to further enhance detection capabilities, especially in complex attributed networks. CDBNE notably extends DAEGC model by introducing two key components beyond the standard graph structure reconstruction: a modularity maximization module and an additional GAT decoder for attribute reconstruction. The modularity module explicitly guides the embedding process towards generating communities that adhere to the well-established graph modularity principle, a global quality measure for community partitions. Concurrently, the GAT-based decoder is specifically designed to reconstruct the original attributed information, ensuring that the learned embeddings capture both the structural and rich attribute semantics of the nodes. The objective function for CDBNE is a composite loss designed to optimize these multiple aspects simultaneously:

$$\mathcal{L}_{total} = \mathcal{L}_{rec} + \mathcal{L}_{att} + \gamma\mathcal{L}_{clus} - \beta\mathcal{L}_m \tag{2.29}$$

Here: $\mathcal{L}_{rec}$ represents the reconstruction loss for the graph structure, measuring the error between the original adjacency matrix $\mathbf{A}$ and its reconstruction $\hat{\mathbf{A}}$. $\mathcal{L}_{att}$ denotes the attribute reconstruction loss, which quantifies the discrepancy between the original node attributes $\mathbf{X}$ and their reconstruction $\hat{\mathbf{X}}$ generated by the dedicated GAT decoder (e.g., $\hat{\mathbf{X}} = f_{GAT}(\mathbf{Z})$, where f_{GAT} is the GAT decoder). $\mathcal{L}_{clus}$ is the self-clustering loss, which operates identically to that described for DAEGC, leveraging KL divergence to guide embeddings into distinct clusters based on their proximity to learned centroids. $\mathcal{L}_m$ is the modularity maximization loss, which aims to maximize the modularity of the detected communities. It is typically included as a negative term in the overall minimization objective to achieve maximization, and can be represented as:

$$\mathcal{L}_m = \frac{1}{4m}\mathrm{Tr}(\mathbf{H}^\top \mathbf{B}\mathbf{H}) \tag{2.30}$$

where m is the number of edges, $\mathbf{B} \in \mathbb{R}^{N\times N}$ is the modularity matrix ($\mathbf{B} = \mathbf{A} - \frac{dd^\top}{2m}$ where d is the degree vector), and $\mathbf{H}$ represents the community assignments (derived from node embeddings **Z**). Finally, β and γ are hyperparameters that balance the contributions of the various loss terms, allowing for flexible optimization based on specific network characteristics.

CDBNE demonstrates notably enhanced performance in terms of clustering accuracy, NMI, ARI compared to baseline models such as DAEGC, VAGER, and other foundational GAE variants. However, these improvements come at the cost of added architectural and objective function complexity, which can present challenges in practical application and further development.

Dual encoder approach involves employing two distinct encoder networks, each designed to process a different type of input data and generate its own embedding. As illustrated in Figure 2.7, this design enables the model to effectively capture and integrate multiple aspects of the input. In essence, dual encoders are well-suited for handling complex, multi-modal data by utilizing parallel encoding streams within a unified framework.

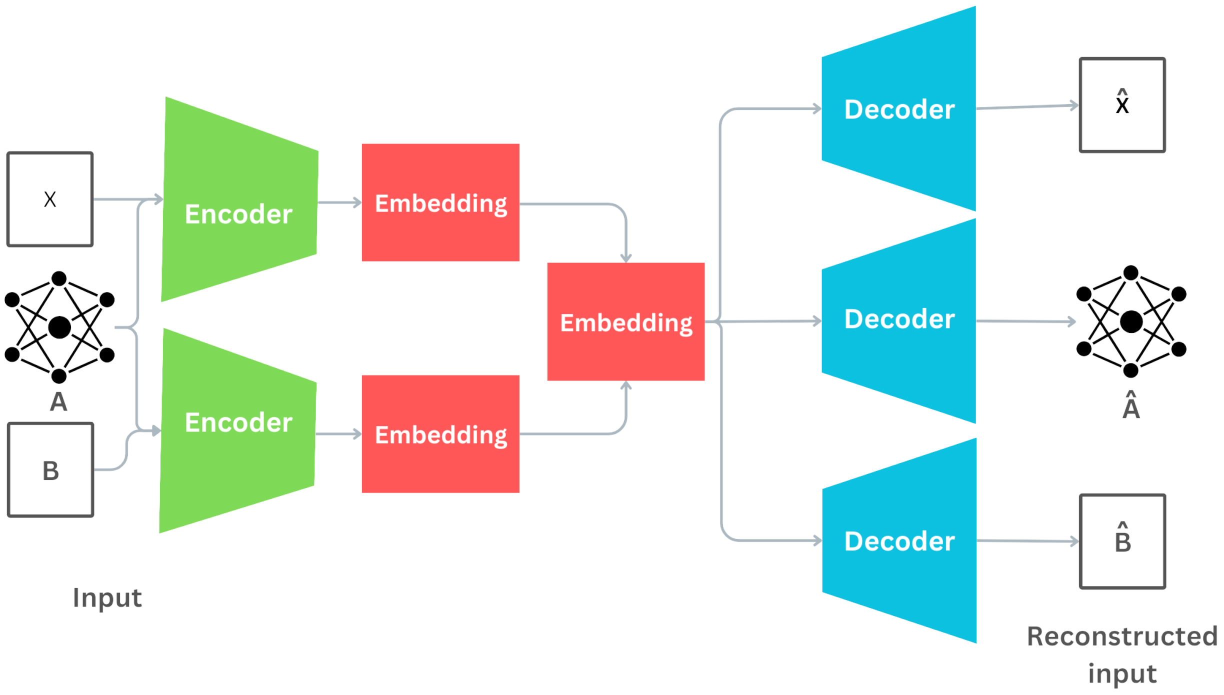


Figure 2.7—General architecture of the Dual Graph Attention Autoencoder, illustrating the dual-input design that processes (i) topological and attribute information, and (ii) topological and modularity information.

The Deep Dual Graph Attention Auto-Encoder (DDGAE) (Wu et al., 2024) represents an advanced methodological framework for community detection in attributed networks, integrating dual-view representation learning with self-optimizing clustering. The model processes attributed graphs through two complementary pathways: a high-order modularity view based on matrix **B** and a node attribute view based on feature matrix **X**. The modularity view employs a refined mathematical representation of community structure derived from the adjacency matrix, explicitly encoding mesoscopic patterns beyond direct neighborhood connections. Simultaneously, the attribute view processes node features through feature-aware attention mechanisms. Both pathways utilize dedicated graph attention encoders with multi-head attention layers that dynamically weight neighborhood influences during feature aggregation, enabling nodes to selectively emphasize connections indicative of latent community affiliations.

The embeddings generated from both views, $\mathbf{Z}_X$ (attribute view) and $\mathbf{Z}_B$ (modularity view) are fused through average pooling:

$$\mathbf{Z} = \frac{\mathbf{Z}_X + \mathbf{Z}_B}{2}.$$

This unified representation feeds into a triple-decoder architecture: a topological decoder regenerates adjacency relationships through inner product operations. An attribute decoder reconstructs original node features $\hat{\mathbf{X}}$ via feature-propagating layers and a specialized modularity decoder reconstructs community-informed structural patterns $\hat{\mathbf{B}}$. Each decoder contributes to a composite reconstruction loss that quantifies discrepancies between original and reconstructed graph properties.

The optimization objective is defined as:

$$\mathcal{L} = \mathcal{L}_f + \lambda \mathcal{L}_r + \beta \mathcal{L}_c$$

where λ and β are hyperparameters, $\mathcal{L}_r$ denotes the binary cross-entropy loss for adjacency reconstruction, $\mathcal{L}_f$ combines attribute and modularity reconstruction losses:

$$\mathcal{L}_f = \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 + \|\mathbf{B} - \hat{\mathbf{B}}\|_F^2,$$

and $\mathcal{L}_c$ is the self-clustering objective based on Kullback-Leibler divergence:

$$\mathcal{L}_c = \sum_{i=1}^{n} \sum_{k=1}^{K} p_{ik} \log \frac{p_{ik}}{q_{ik}}, \quad q_{ik} = \frac{(1 + \|\mathbf{z}_i - \boldsymbol{\mu}_k\|^2)^{-1}}{\sum_{k'=1}^{K} (1 + \|\mathbf{z}_i - \boldsymbol{\mu}_{k'}\|^2)^{-1}}$$

with $\boldsymbol{\mu}_k$ cluster centroids and $p_{ik} = q_{ik}^2/\sum_i q_{ik}$ as sharpened target distributions. Final community assignments derive from the optimized latent space as:

$$y_i = \underset{k}{\operatorname{argmax}} \ (q_{ik}^*)$$

where q_{ik}^* computes the converged soft assignment between node $\mathbf{z}_i^*$ and centroid $\boldsymbol{\mu}_k^*$.

Empirical validation across five benchmark networks demonstrates consistent performance advantages, particularly in detecting non-overlapping communities within attributed networks. Performance metrics enhanced in NMI and ARI compared to simple encoder models, attributable to the model's dual-pathway integration of structural and semantic signals. Attention analysis further confirms the framework's capability to resolve attribute-topology conflicts, evidenced by elevated attention weights assigned to nodes bridging multiple communities. These advantages, however, entail computational trade-offs: The requirement to compute and process the modularity matrix establishes quadratic memory complexity, while the dual attention mechanism increases training time relative to conventional graph autoencoders. These scalability constraints become pronounced in moderately-sized networks, suggesting optimal application in structurally complex graphs where precise attribute-topology alignment is paramount.

2.4.4 Deep Nonnegative Matrix Factorization

While standard NMF (Section 2.3.2) provides interpretable low-rank representations, its single-layer factorization exhibits fundamental limitations when modeling complex data structures. The shallow representation struggles to capture hierarchical features prevalent in real-world data, such as multi-scale community structures in networks. Furthermore, the linear factorization $\mathbf{A} \approx \mathbf{UP}$ inherently constrains expressive power, while the fixed low-rank constraint ($k \ll n$) risks oversimplifying heterogeneous data distributions. These limitations motivate *deep nonnegative matrix factorization* (*DNMF*), which learns hierarchical representations through sequential matrix factorizations.

DNMF extends NMF via L layered decompositions:

$$\begin{aligned} \mathbf{A} &\approx \mathbf{U}^{(1)}\mathbf{P}^{(1)}, \\ \mathbf{P}^{(1)} &\approx \mathbf{U}^{(2)}\mathbf{P}^{(2)}, \\ &\vdots \\ \mathbf{P}^{(L-1)} &\approx \mathbf{U}^{(L)}\mathbf{P}^{(L)}, \end{aligned}$$

yielding the deep architecture $\mathbf{A} \approx \mathbf{U}^{(1)}\mathbf{U}^{(2)}\cdots\mathbf{U}^{(L)}\mathbf{P}^{(L)}$, with all factors nonnegative. This structure captures features at multiple abstraction levels, where nested reconstructions introduce implicit nonlinearities that enhance modeling capacity. Layer-wise nonnegativity constraints further induce structured sparsity patterns beneficial for interpretation. Optimization typically extends NMF algorithms through coordinated layer-wise training. For network analysis, DNMF's multi-resolution capability (Song et al., 2015) proves particularly valuable in heterogeneous graphs with varying organizational scales.

Deep Autoencoder-like NMF (DANMF) (Ye et al., 2018) generalizes symmetric NMF ($\mathbf{A} \approx \mathbf{U}\mathbf{U}^\top$) into an encoder-decoder framework. The encoder compresses adjacency information through hierarchical layers $\mathbf{U}^{(l)}$ to a bottleneck embedding $\mathbf{V}^{(L)}$, while the decoder reconstructs $\mathbf{A}$ through the product $\mathbf{U}^{(1)}\cdots\mathbf{U}^{(L)}\mathbf{V}^{(L)}$. An auxiliary reconstruction $\mathbf{V}^{(L)} \approx (\mathbf{U}^{(L)})^\top\cdots(\mathbf{U}^{(1)})^\top\mathbf{A}$ ensures information preservation. The unified objective incorporates graph smoothness:

$$\mathcal{L} = \mathcal{L}_D + \mathcal{L}_E + \lambda\mathcal{L}_{\text{reg}} = \left\|\mathbf{A} - \prod_{l=1}^{L}\mathbf{U}^{(l)}\mathbf{V}^{(L)}\right\|_F^2 + \left\|\mathbf{V}^{(L)} - \prod_{l=L}^{1}(\mathbf{U}^{(l)})^\top\mathbf{A}\right\|_F^2 + \lambda\,\mathrm{tr}\Big((\mathbf{V}^{(L)})^\top\mathbf{L}\mathbf{V}^{(L)}\Big), \tag{2.31}$$

Here $\mathcal{L}_D$ denotes the decoder reconstruction loss and $\mathcal{L}_E$ the encoder reconstruction loss. The regularizer $\mathcal{L}_{\text{reg}}$ enforces graph smoothness based on the combinatorial graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The notation $\|\cdot\|_F$ denotes the Frobenius norm (cf. Equation 2.10). The scalar $\lambda > 0$ is a tunable hyperparameter that balances reconstruction fidelity against the smoothness prior. Nonnegativity constraints promote hierarchical latent attributes that bridge topological features with community structure.

Constrained Symmetric Deep Autoencoder NMF (CSDNMF) (Zhang et al., 2024) enhances multi-scale community detection through bidirectional consistency: The encoding phase projects $\mathbf{A}$ through successive nonnegative layers to high-level embeddings, while the decoding phase reconstructs $\mathbf{A}$ via mirrored factors. This approach incorporates two complementary regularizers: a graph-smoothness term $\mathrm{Tr}(\mathbf{Z}^\top\mathbf{L}\mathbf{Z})$ that preserves local connectivity patterns, and a symmetry penalty $\sum_{l=1}^{L} \|\mathbf{U}_{\text{enc}}^{(l)} - \mathbf{U}_{\text{dec}}^{(l)}\|_F^2$ that aligns encoder/decoder factors to enhance robustness in undirected networks.

Modularized Robust Adaptive Semi-Supervised NMF (MRASNMF) (Ghadirian and Bigdeli, 2024) reformulates modularity maximization as a symmetric NMF, enabling integration of partial supervision via a knowledge-priority matrix. Its architecture combines complementary community perspectives through multi-view clustering while employing adaptive weighting to dynamically balance the influence of potentially noisy labels. The optimization features an iterative update scheme with proven convergence that tunes the trade-off between modularity maximization and pairwise constraint satisfaction.

Both DANMF and CSDNMF achieve moderate results across citation networks such as Cora, CiteSeer, and PubMed. Specifically, CSDNMF consistently outperforms DANMF in terms of NMI, ARI, F-score, and ACC, indicating improved community separability and clustering quality. These models also surpass traditional embedding-based methods such as Node2Vec (Grover and Leskovec, 2016) and LINE (Tang et al., 2015). In contrast, MRASNMF is evaluated on a different category of relatively small, non-attributed networks and incorporates prior knowledge into the factorization process with varying percentages.

2.4.5 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) frame representation learning as a two-player minimax game between a generator G and a discriminator D. The generator attempts to produce samples (here, candidate node-embeddings or neighbor distributions) that mimic the true data distribution, while the discriminator seeks to distinguish real samples from those synthesized by the generator G. Formally, the basic GAN objective is the following:

$$\min_G \max_D \; V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] \; + \; \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]. \tag{2.32}$$

In the graph-embedding setting (as in GraphGAN (Wang et al., 2018a)), one treats each node v as "real" data and models the generator's goal as sampling plausible neighbors u for v. Concretely, let $p_{\text{true}}(u \mid v)$ be the empirical neighbor distribution and let $G(u \mid v)$ be the generator's learned conditional distribution. The adversarial objective becomes as follows:

$$\min_{G}\max_{D}\sum_{v\in V}\Big[\mathbb{E}_{u\sim p_{\text{true}}(\cdot|v)}\log D(v,u)\ +\ \mathbb{E}_{u\sim G(\cdot|v)}\log\big(1-D(v,u)\big)\Big]. \tag{2.33}$$

Here $D(v,u)$ outputs the probability that (v,u) is an actual edge. By alternating gradient updates on D and G, the method learns embeddings for each node (through the parameters of G and D) that capture both local connectivity and higher-order structure in the graph.

Adversarial robustness emerges as the generator G continuously adapts to the discriminator's feedback. In social-network analysis this adaptation improves link prediction and community detection performance. Extensions that integrate node attributes or hierarchical community priors into the adversarial framework, yield richer, domain-aware embeddings.

CommunityGAN (Jia et al., 2019) casts overlapping community detection as a motif-aware adversarial game, fundamentally extending the Affiliation Graph Model (AGM) (Yang and Leskovec, 2012). While the original AGM is a generative model that defines the probability of an edge existing between nodes based on their community affiliations (Yang and Leskovec, 2012). CommunityGAN embeds this probabilistic community-affiliation structure into a Generative Adversarial Network (GAN) framework and elevates the task from edge generation to motif (particularly clique) generation (Jia et al., 2019). In CommunityGAN, each node v is represented by a nonnegative membership vector $\mathbf{F}_v \in \mathbb{R}^C_{\geq 0}$, where $F_{v,c}$ measures v's affiliation strength to the community c. For any candidate set of r vertices $s = \{v_1,\ldots,v_r\}$, CommunityGAN first computes, for each community c, $p_c(s) = 1 - \exp\Big(-\prod_{i=1}^{r} F_{v_i,c}\Big)$, and then aggregates over all communities to obtain:

$$p(s) = 1 - \exp\Big(-\sum_{c=1}^{C}\prod_{i=1}^{r} F_{v_i,c}\Big) = 1 - \exp\big(-\odot(\mathbf{F}_{v_1},\ldots,\mathbf{F}_{v_r})\big), \tag{2.34}$$

with $\odot$ denoting the sum of entry-wise products. The generator G, parameterized by $\{\mathbf{F}_v\}$, builds motifs one vertex at a time, selecting each next node in proportion to its clique probability with the already chosen set. This can be implemented efficiently via a relevance-weighted random walk on a virtual "product" node. The discriminator D, with its own embeddings $\{\mathbf{d}_v\}$, scores any candidate motif s as:

$$D(s) = 1 - \exp\big(- \odot (\mathbf{d}_{v_1}, \dots, \mathbf{d}_{v_r})\big). \tag{2.35}$$

These two networks then compete under the minimax objective:

$$\min_{\theta_G} \max_{\theta_D} \sum_{v \in V} \Big[\mathbb{E}_{s \sim p_{\text{true}}(\cdot|v)} \log D(s) + \mathbb{E}_{s \sim G(\cdot|v)} \log\big(1 - D(s)\big)\Big]. \tag{2.36}$$

with D trained to distinguish real from generated motifs and G updated (through policy gradients) to fool D. At convergence, the learned $\{\mathbf{F}_v\}$ preserve high-order clique structure and directly reveal overlapping communities, eliminating the need for any post-hoc clustering.

SEAL (Zhang et al., 2020b) reframes local community detection as a seed-aware adversarial growth process. Its architecture comprises four key components: a seed selector, implemented as a semi-supervised node regression model, which predicts the suitability of nodes as seeds for community growth; a specialized generator based on an Incremental Graph Pointer Network (iGPN) that expands a seed into a full community through sequential neighbor node attachments; a Graph Isomorphism Network (GIN) discriminator that evaluates candidate subgraphs against ground-truth communities to provide structural feedback; and a locator that identifies regions of high intra-connectivity to suppress free-rider nodes and provide auxiliary regularization.

There are several GAN-based embedding approaches, such as GraphGAN (Wang et al., 2018a) and GANE (Generative Adversarial Network Embedding) (Hong et al., 2020), that learn node representations adversarially, which can subsequently be fed into a clustering algorithm (e.g. k-means) for community detection. GANSE (Semi-Supervised Generative Adversarial Network) (Liu et al., 2024) is an early semi-supervised community detection method based on GANs. First, it reinforces the input topology using vertex-similarity-based link prediction to address missing edges. A generator is then trained to generate candidate partitions, countered by a discriminator trained on limited ground-truth samples. Internal cohesion is enforced by injecting a clustering-coefficient reward into the training objective. Finally, isolated nodes are reallocated to their strongest affinity communities. The experiments evaluated GANSE on four large real-world networks with simulated edge incompleteness, GANSE achieves superior performance compared with CommunityGAN and SEAL.

2.5 Conclusion

This chapter provided a comprehensive survey of community detection methods, establishing problem definition, outlining evaluation measures, summarizing benchmark datasets, and presenting key application areas. To navigate the diverse landscape of approaches, we introduced a taxonomy categorizing methods as classical community detection algorithms, traditional machine learning-based methods, and contemporary GNN-based techniques. Our analysis of each category detailed their underlying principles and representative performance. Building on this survey, the next chapter advances the discussion by presenting two comprehensive comparative studies that rigorously evaluate the empirical performance of leading GNN methods, focusing on two prominent architectures.

Second part

Contributions

« لَيسَ الجَمَالُ بِأَثوابٍ تُزَيِّنُنا ... إِنَّ الجَمالَ جَمالُ العَقلِ وَالأَدَبِ »

Ali ibn Abi Talib

CHAPTER 3

Community Detection Methods: Comparative Studies

Community detection in complex networks seeks to uncover groups of nodes that are more densely connected internally than with the rest of the network. Over the past decade, a wide variety of deep-learning methods have been proposed to leverage both structural and attribute information for the community detection task. Among these, convolutional neural networks (CNNs), adapted to graph domains, and graph auto-encoders (GAEs) represent two prominent paradigms. CNN-based approaches exploit localized filter operations to capture motif-level patterns, while GAE-based methods learn low-dimensional embeddings via reconstruction and clustering objectives. Our research (Bekkair et al., 2023, 2024) address these comparative studies. The specific models used in our first and second comparative studies are detailed in Section 2.4.1.2 and Section 2.4.3, respectively.

Despite their shared goal, these two families differ substantially in their fundamental approach. A comparison through separated experiments is therefore essential to understand their relative strengths, weaknesses, and suitability for different network types.

3.1 Experimental Setup

This section details the experimental setup for the two comparative studies, including computational environments, key software libraries, datasets, and evaluation metrics.

3.1.1 CNN-Based Community Detection

All CNN-based CD experiments were conducted on Google Colab's, which provides 12.7 GB of system RAM and approximately 15 GB of GPU memory. The model development and evaluation were implemented in Python 3.8, using NetworkX[1] for graph construction, manipulation, and preprocessing, and Keras[2] for defining convolutional layers, building model architectures, and orchestrating training.

3.1.2 GAE-Based Community Detection

The GAE-based CD experiments, which focus on attributed networks, were executed on a local workstation equipped with a 512 GB SSD, 32 GB of RAM, and an Intel® Xeon® Silver 4112 quad-core processor (2.60 GHz, 8.25 MB cache, 8 threads). All modeling and analysis were performed in Python 3.8 Jupyter notebooks, leveraging PyTorch Geometric (PyG)[3] for graph neural network layers and utilities, and TensorFlow[4] for implementing deep-learning modules and operations.

3.1.3 Datasets

For the CNN-based study, we used four ground-truth benchmark networks: Zachary's karate club, the Dolphins social network, the American college football network, and the Email-Eu-core communication network (Zachary, 1977; Lusseau et al., 2003; Girvan and Newman, 2002). For the GAE-based study, we employed three citation networks Cora, CiteSeer, and PubMed (McCallum et al., 2000; Giles et al., 1998; Sen et al., 2008) (see Section 2.1.2), each providing node feature matrices and ground-truth community labels.

3.1.4 Evaluation Metrics

For the CNN-based community detection experiments, we employ the normalized mutual information (NMI) as the sole metric to assess how well detected clusters align with ground truth communities. In contrast, the GAE-based experiments are evaluated using

[1]https://networkx.org/
[2]https://keras.io/
[3]https://pytorch-geometric.readthedocs.io/
[4]https://www.tensorflow.org/

three complementary metrics NMI, adjusted Rand index (ARI), and F1-score to capture the overall agreement, the pairwise clustering accuracy, and the balance between precision and recall, respectively. All metrics are defined in Section 2.1.1.

3.2 A Comparative Study of CNN-based CD

In this section, we present the detailed implementations of two CNN-based transformation approaches for community detection and analyze their empirical performance across four benchmark networks.

3.2.1 Implementation

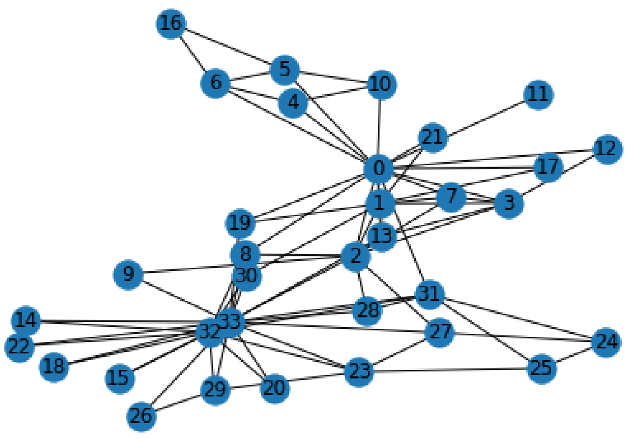
In this stage, we describe the two transformation-based approaches studied in (Xin et al., 2017) and (Cai et al., 2020), namely the node-based and edge-based methods.

The node-based transformation method first extracts the adjacency matrix **A** of the target network and augments it with local neighbor information via the function defined in Equation 2.12, using a hop count $S_0 = 2$ and an attenuation factor $\sigma = 0.60$. These parameter values are chosen based on experimental validation and are consistent with state-of-the-art practices. This enriched adjacency matrix encodes, for each node, the strength of connections up to two hops away and yields an $N \times N$ matrix whose i-th row represents the relationship of node i with every other node. Rather than reshape each row into a two-dimensional submatrix as in the original paper, we preserve the one-dimensional structure and apply 1D convolutions directly to the N-length vector.

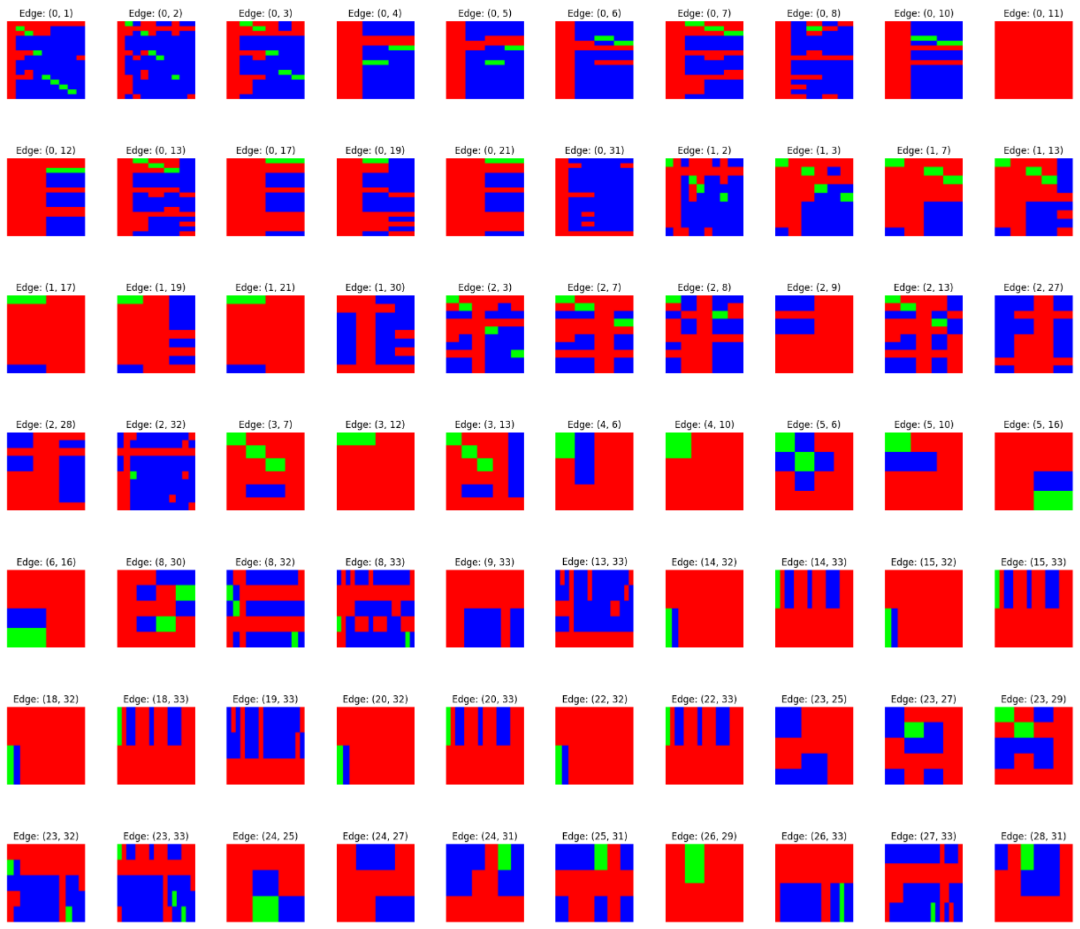
Our specific CNN implementation comprises two successive 1D convolutional layers, each with ten filters of kernel size five, followed by 1D max-pooling of size two and a final fully connected layer with an output activation function whose number of outputs matches the number of communities in the dataset. A batch normalization layer precedes the flattening step to improve training stability.

The edge-based transformation approach, by contrast, begins by converting each edge (i, j) into a color image via the E2I model (Cai et al., 2020). Let set_i and set_j denote the neighbor sets of nodes i and j, respectively; we construct an $x \times y \times 3$ matrix whose entries encode whether a neighbor pair $(u, v) \in \text{set}_i \times \text{set}_j$ is directly connected (red), reflects a two-hop relationship (green), or is disconnected (blue). Since the degrees of nodes in complex networks are highly variable, the generated edge images differ in spatial size. To ensure compatibility with the CNN input, an image normalization process is applied. Specifically, the area interpolation cropping method provided by TensorFlow (Abadi et al., 2016) is used to resize all images to a uniform $M \times M \times 3$ format, achieving consistent input dimensions across the dataset. These normalized images are then fed into the CNN (see Figure 3.1b for examples on the Zachary dataset).

In the second stage, the resulting images are classified by the comNet architecture into ”inner” (intra-community) or ”inter” (inter-community) edges; comNet employs multiple 2D convolutional layers with varied kernel sizes, 2D max-pooling of size two, two fully connected layers (512 neurons and 2 neurons, respectively) and batch normalization layers of size 40, strictly following the specifications of the original work. Finally, the algorithm identifies the top K disconnected components (out of M) that maximize network modularity R and treats these as preliminary communities. The remaining $M - K$ components are iteratively merged into the preliminary set in a manner that further increases R, yielding the final community assignments.



(a) Zachary karate network.



(b) The converted edges of Zachary karate network colored 2D images via E2I.

Figure 3.1—Zachary karate network and its corresponding edge-to-image (E2I) representation.

3.2.2 Results and Discussion

We evaluate the CNN-based community detection performance on four benchmark networks using the normalized mutual information (NMI) score. Table 3.1 presents NMI results for the node-based and edge-based transformation approaches when 50% of the nodes are labeled in the node-based method. Both methods achieve perfect alignment with ground truth on the Zachary karate, Dolphins, and Football networks, demonstrating their ability to recover known community structure in small graphs. On the larger Email-Eu-core network, the node-based approach attains a respectable NMI of 0.74, whereas the edge-based method only reaches 0.26, indicating that its classification and modularity merging pipeline struggles with increased scale and complexity.

Figure 3.1b illustrates sample E2I transformations for the Zachary network, where each edge is represented as a color image encoding neighbor relationships. Although the edge-based pipeline can produce high accuracy on small graphs, it incurs substantial implementation and computational overhead: it requires color-image generation, a deep convolutional classifier, and a modularity-based component that duplicates and merges subgraphs. In contrast, the node-based transformation is straightforward to implement augmenting the adjacency matrix with locality information and applying 1D CNNs directly to node-wise vectors and scales more gracefully with network size.

Overall, while both methods perform equally well on small networks, the node-based approach offers a simpler, more efficient solution that sustains high accuracy on larger datasets.

Table 3.1—NMI scores for the node-based and edge-based transformation approaches using 50% labeled nodes across benchmark networks

Dataset	Node-based Transformation Approach	Edge-based Transformation Approach
Zachary karate	1.0	1.0
dolphins	1.0	1.0
football	0.92	0.93
Email-Eu-core	0.74	0.26

3.3 A Comparative Study of Graph Autoencoder-based CD Methods in Attributed Networks

In this section, we present the detailed implementations and empirical evaluation of two representative graph autoencoder (GAE) models for community detection in attributed networks. Both methods learn low-dimensional embeddings by reconstructing network structure and/or node attributes, followed by clustering in a latent space.

3.3.1 Implementation

All models considered in this study fall into two categories: simple encoders and dual encoders and follow a common pipeline of unsupervised embedding extraction followed by K-means clustering (with the number of communities K specified a priori). For reproducibility, every experiment was implemented in Python within Jupyter notebooks, random seeds were fixed at 42, and results were averaged over five independent runs (mean $\pm$ standard deviation). The core graph-neural-network components and utilities were built using PyTorch Geometric (PyG)[5], TensorFlow[6], and scikit-learn for clustering and metric calculations.

The simple-encoder group begins with Embedding Graph AutoEncoder (EGAE) (Zhang et al., 2023), which uses two graph-convolutional layers: the first layer projects raw node features into a 32-dimensional space with ReLU activation, and the second compresses to 16 dimensions; an inner-product decoder reconstructs the adjacency matrix, and L1 regularization promotes sparsity.

Building on this design, ARGA and its variational counterpart ARVGA (Pan et al., 2018) incorporate adversarial training via a discriminator, ARGA retains the two-layer GCN encoder and inner-product decoder while enforcing a predefined prior on the embeddings, whereas ARVGA adds a third layer that models the mean and standard deviation of a Gaussian latent distribution and samples embeddings via the reparameterization trick.

DAEGC (Wang et al., 2019) replaces the GCN encoder with a two-layer graph-attention network (GAT), mapping features first into a 256-dimensional hidden space and then reducing to 16 dimensions. Its inner-product decoder reconstructs the adjacency

[5] https://pytorch-geometric.readthedocs.io/
[6] https://www.tensorflow.org/

matrix, and a self-training clustering module iteratively refines soft assignments by minimizing the Kullback–Leibler divergence between inferred labels and an ideal target distribution.

In contrast, the dual-encoder DDGAE model (Wu et al., 2024) employs two parallel GAT encoders to capture complementary views of the graph, one emphasizing topological modularity and the other focusing on node attributes. Each encoder projects inputs into a 256-dimensional space before reducing to 16 dimensions. The architecture then reconstructs both the adjacency and attribute matrices through inner-product decoders, ensuring comprehensive preservation of structural and attribute information. A self-training module alternates between updating community centers and refining node assignments by optimizing a combined reconstruction and clustering loss, with final community membership determined by an argmax over the resulting soft assignment scores.

3.3.2 Results and Discussion

The community detection performance of the five models grouped into simple-encoder methods (DAEGC, EGAE, ARGA, ARVGA) and the dual-encoder DDGAE is summarized in Table 3.2 for the Cora, CiteSeer and PubMed datasets.

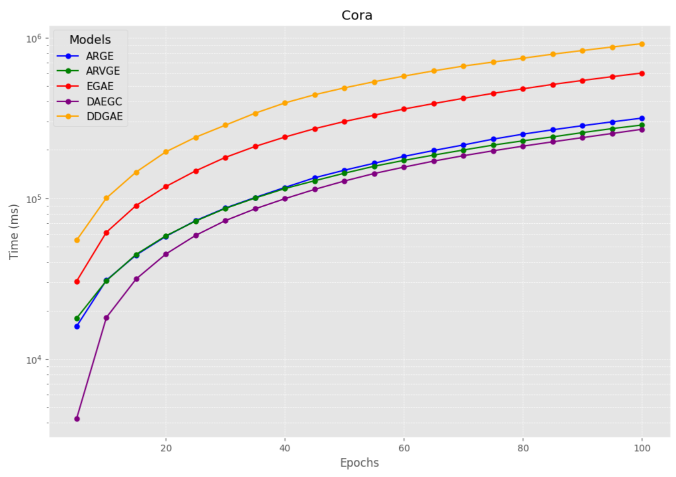
On Cora, EGAE achieves the highest NMI (0.5518) while DDGAE attains the best ARI (0.5243) and F1-score (0.7106). For CiteSeer, DAEGC leads in both NMI (0.3665) and ARI (0.3423), whereas ARGA records the top F1-score (0.5888). On the larger and more complex PubMed corpus, DDGAE outperforms all simple encoders across every metric (NMI = 0.3004, ARI = 0.3151, F1 = 0.6814), reflecting its ability to integrate adjacency, attribute and modularity information. These results suggest that the additional modularity matrix **B** exploited by the dual-encoder architecture provides a richer representation, particularly valuable in denser graphs like PubMed. Simple-encoder models benefit from advanced decoder designs EGAE's L1-regularized inner-product decoder and DAEGC's attention-based reconstruction both yield competitive performance, but lack the multi-view fusion that distinguishes DDGAE. Notably, ARVGA's variational formulation offers robustness to uncertainty, yet its performance lags behind ARGA on most datasets, indicating that adversarial regularization may not uniformly translate into clustering gains.

Moreover, Figure 3.2 compares execution times for each model across datasets. The DDGAE model consistently incurs the greatest runtime due to its dual-encoder structure and simultaneous reconstruction of multiple input matrices. EGAE, while more efficient than DDGAE, still requires longer training than the other simple-encoder methods, reflecting the overhead of its advanced decoder. DAEGC and ARGA strike a balance between computational cost and accuracy, with ARVGA demonstrating the shortest runtime but also the lowest clustering metrics overall.

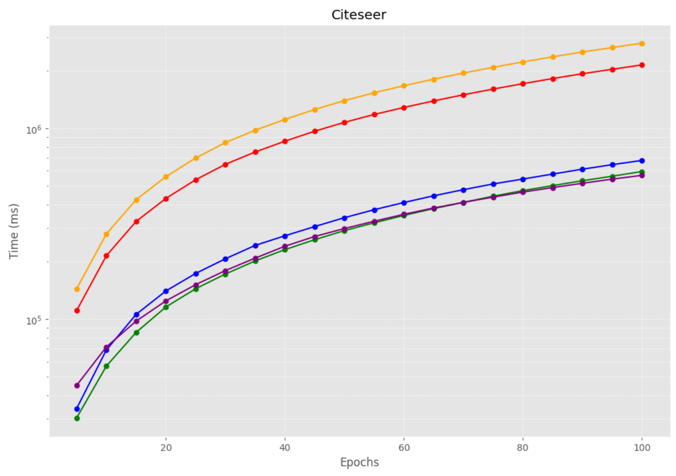
In summary, DDGAE emerges as the most effective approach for attributed community detection, especially on large, complex graphs, by virtue of its multi-view encoding and sophisticated decoders, at the expense of increased computational resource demands. Simple-encoder methods such as EGAE and DAEGC remain attractive when resources are limited or when rapid prototyping is required, offering respectable performance with lower runtime overhead.

Table 3.2—Comparative performance of GAE models on Cora, CiteSeer, and PubMed datasets in terms of NMI, ARI, and F1-score. All results are reported as the average over five independent runs.

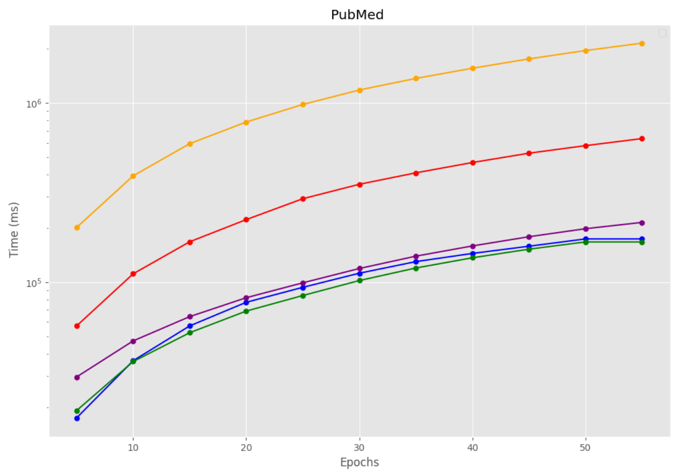
Category	**Model**	Cora			CiteSeer			PubMed		
		NMI	ARI	F1-Score	NMI	ARI	F1-Score	NMI	ARI	F1-Score
Simple Encoder	DAEGC	0.5223	0.4844	0.6929	**0.3665**	**0.3423**	0.5121	0.1079	0.0828	0.4639
	EGAE	**0.5518**	0.5102	0.5050	0.2806	0.2720	0.1911	0.2566	0.2555	0.5833
	ARGA	0.4892	0.4224	0.6891	0.3070	0.2936	**0.5888**	0.2185	0.2021	0.6173
	ARVGA	0.3938	0.2850	0.5032	0.3504	0.3291	0.5842	0.0443	0.0110	0.3637
Dual Encoder	DDGAE	0.5423	**0.5243**	**0.7106**	0.3032	0.2665	0.5038	**0.3004**	**0.3151**	**0.6814**



(a) Cora



(b) CiteSeer.



(c) PubMed.

Figure 3.2—Execution time comparison across three datasets, measured over training epochs.

3.4 Conclusion

In this chapter, we have presented two complementary comparative studies of deep-learning approaches of community detection: one based on convolutional neural networks (CNNs) applied to structural transformations of graph data, and the other on graph autoencoders (GAEs) in attributed networks.

Across both studies, our experiments employed multiple benchmark datasets and rigorous evaluation metrics, NMI for CNN methods, and NMI, ARI, and F1-score for GAE methods, ensuring that comparisons reflect real differences in model design rather than idiosyncrasies of data or implementation.

The CNN-based study demonstrated that both node and edge-transformation pipelines can recover ground-truth communities with high accuracy on small networks. The node-based 1D-CNN approach, however, stands out for its simplicity, lower computational overhead, and good scaling to larger graphs (e.g. Email-Eu-core), whereas the edge-based image-classification pipeline, despite matching performance on toy networks, incurs significant preprocessing and runtime costs and degrades on larger inputs.

Similarly, in the attributed-network setting, GAE models reliably uncover community structure by embedding nodes into low-dimensional spaces and clustering. The simple EGAE achieves respectable performance with minimal complexity, but the dual-encoder DDGAE consistently delivers superior accuracy, particularly on large, richly attributed graphs like PubMed, by integrating both topological and modularity information. This multi-view fusion comes at the expense of longer training times and higher memory usage, highlighting an inherent trade-off between representational power and computational efficiency.

Taken together, these results underline a recurring theme: more expressive architectures (edge-based CNNs, dual-encoder GAEs) can yield incremental accuracy gains but demand greater resources, whereas streamlined, single-view models (node-based CNNs, simple GAEs) offer a pragmatic balance of performance, simplicity, and scalability. For practitioners working with real-world networks, the choice between these paradigms should be guided by data size, attribute richness, and available computational budget.

In light of the comparative study's findings, the upcoming chapter describes our original models and associated algorithms, which constitute the primary contributions of this thesis.

Chapter 4

Proposed Methods: Classical Refinement and GAE

Detecting cohesive groups of nodes, known as communities, lies at the heart of network science, offering insights into everything from functional modules in biological systems to interest clusters on social media and topic relationships in information networks. In this chapter, we introduce two contributions that advance community detection in attributed graphs and leverage deep learning on graphs. First, we describe Adamic–Adar Label Propagation (AA-LPA), a deterministic refinement of the classic Label Propagation Algorithm that uses shared-neighbor centrality to stabilize and sharpen community assignments. Next, we present a novel unsupervised model based on graph autoencoders tailored for attributed networks, which jointly learns node embeddings and community structure without relying on ground-truth labels. Together, these models bridge efficient, interpretable methods with powerful representation learning to deliver robust solutions to real-world community detection challenges.

4.1 Enhanced Label Propagation Algorithm (LPA)

Label Propagation Algorithm (LPA) (Detailed in Section 2.2.6) stands out for its conceptual simplicity and its near-linear time complexity, making it especially attractive for large-scale graphs. In LPA, each node iteratively adopts the most frequent label among its neighbors until convergence, producing a partition of the network into communities. However, despite its efficiency, LPA's dependence on the arbitrary order in which nodes are updated and on uniform weighting of neighbor labels frequently yields unstable, fragmented communities, which undermines reproducibility and interpretability in critical applications such as disease module detection or personalized recommendation systems.

To overcome these shortcomings, researchers have proposed a variety of stabilizing enhancements. Centrality-based node ordering schemes employ measures such as degree, betweenness or k-core values to impose a deterministic update sequence (Wang et al., 2017b); edge-weighting methods leverage metrics from random-walk theory to bias label influence toward more structurally significant connections (Pons and Latapy, 2005). While these strategies improve consistency, they often fail to incorporate the nuanced information encoded in shared-neighbor relationships or address only one source of randomness (ordering or tie-breaking) at a time.

In this part, we introduce *Adamic–Adar Label Propagation* (AA-LPA) (Bekkair et al., 2025b), a fully deterministic variant of LPA that uses a unified structural-centrality framework to address update ordering, label-influence weighting, and tie resolution simultaneously. The Adamic–Adar index itself is defined in Section 4.1.1 of this chapter. Our key innovations are:

1. **AA-based node prioritization.** We compute each node's Adamic–Adar centrality, which emphasizes the importance of shared neighbors, to determine a fixed, high-resolution propagation order.

2. **AA-weighted label influence.** When aggregating neighbor labels, we multiply each label's frequency by the sum of its Adamic–Adar scores and further amplify this weight by the current size of the candidate community, thereby suppressing noisy fluctuations.

3. **Deterministic tie-breaking.** In the event of equal maximum influence, we resolve ties by comparing the respective centrality measures of the competing label communities, ensuring that every propagation step is reproducible.

We evaluate AA-LPA on several benchmark networks of varying size and topology, including Zachary's Karate Club, the Dolphins social network and the Football network, and compare against both stochastic LPAs and leading modularity-optimization methods (Louvain, Leiden). Across all datasets, AA-LPA achieves near-perfect agreement with ground-truth community labels (NMI and F1 scores approaching 1.0). By integrating local (shared-neighbor) and mesoscopic (community size) signals within a single deterministic framework, AA-LPA delivers stable, interpretable partitions without sacrificing efficiency.

4.1.1 Adamic–Adar Index

The Adamic–Adar Index (AAI), denoted $S_{\mathrm{AA}}(x, y)$ and introduced by Adamic and Adar (Adamic and Adar, 2003), quantifies the similarity between two nodes based on their shared neighbors, weighting rarer connections more heavily. For nodes x and y, it is defined as

$$S_{AA}(x, y) = \sum_{u \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(u)|}, \tag{4.1}$$

where $\Gamma(x)$ denotes the neighbor set of node x, $\Gamma(y)$ denotes the neighbor set of node y, and $|\Gamma(u)|$ is the degree of the common neighbor u. By down-weighting highly connected "hub" nodes via the logarithm, AAI emphasizes shared neighbors that provide stronger evidence of a meaningful link.

Computing $S_{\mathrm{AA}}(x, y)$ requires examining the intersection of $\Gamma(x)$ and $\Gamma(y)$, yielding a time complexity of $O(d_x + d_y)$, where d_x and d_y are the degrees of x and y respectively. Although AAI may underperform in networks where hubs do not dilute link significance, it remains a popular choice for link prediction in social, biological, and recommendation-system applications due to its ability to capture nuanced structural dependencies.

4.1.2 AA-Weighted LPA with Centrality-Guided Cohesion

AA-Weighted Label Propagation with Centrality-Guided Cohesion is our proposed algorithm as a deterministic community detection algorithm that enhances the classic Label Propagation Algorithm (LPA) by incorporating the Adamic–Adar (AA) index for edge weighting and a centrality-guided tie-breaking strategy. While LPA is valued for its simplicity and efficiency, its random update order and arbitrary tie resolution often produce nonreproducible and fragmented communities. Our method addresses these issues through three key mechanisms: Deterministic Stabilization, AA-weighted Label Influence, and Deterministic Tie Resolution. Each of these is detailed subsequently.

First, for **Deterministic Stabilization**, we propose a novel node centrality metric derived from aggregated Adamic-Adar (AA) scores to prioritize the order in which nodes are updated. This metric, $s(u)$, quantifies the influence of a node u within the network:

$$s(u) = \sum_{v \in \mathcal{N}(u)} S_{AA}(u, v), \tag{4.2}$$

where $\mathcal{N}(u)$ denotes the set of neighbors of node u, and $S_{AA}(u, v)$ (Equation 4.1) represents the Adamic-Adar similarity between node u and its neighbor v. By sorting nodes in descending order of their $s(u)$ values, we ensure that high-centrality nodes (i.e., those with strong connections to well-connected common neighbors) are updated first. This deterministic update order stabilizes the label propagation process and actively encourages the formation of cohesive communities. The rationale is that influential nodes, being well-connected, rapidly disseminate consistent labels throughout the network, accelerating convergence and enhancing the robustness of the detected communities. This mechanism is illustrated in Figure 4.1 using examples such as the Karate Club network and the Dolphins network, highlighting the node with maximal centrality computed via Equation 4.2.

Second, for **AA-Weighted Label Influence**, to further enhance the significance of structurally important connections during label propagation, we define influence weights for edges. This weight, $w(k)$, considers both the local Adamic-Adar similarity and the global community size:

$$w(k) = \sum_{\substack{v \in \mathcal{N}(u) \\ \ell(v) = k}} S_{AA}(u, v) \times |C_k| \tag{4.3}$$

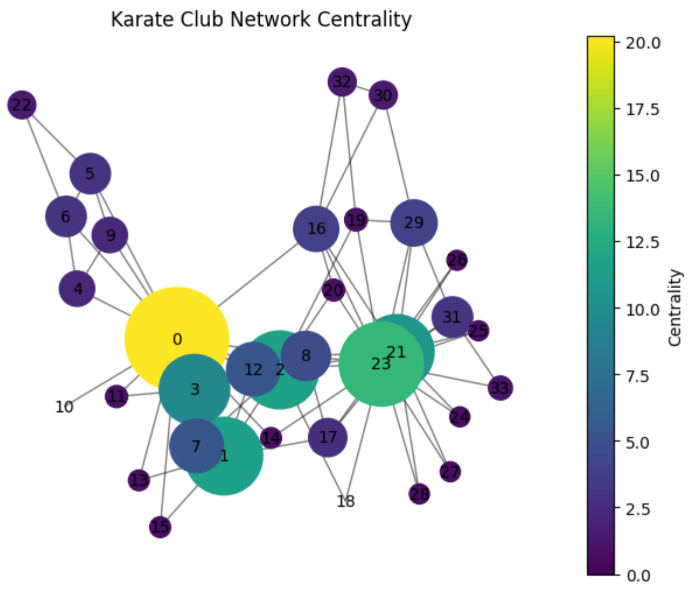
Where, $S_{AA}(u, v)$ quantifies the local structural influence by measuring the Adamic-Adar similarity between node u and its neighbor v, thereby prioritizing edges with discriminative common neighbors. The term $|C_k|$ represents the size of the community associated with label k and acts as a measure of global community attraction. This amplification of labels from larger communities helps to mitigate the inherent fragmentation bias often observed in standard LPA, while simultaneously minimizing noise introduced by incidental or weak edges.

Third, for **Deterministic Tie Resolution**, in instances where a node's neighbors exhibit multiple labels with equal maximal influence (i.e., an equal-weight label conflict), we employ a deterministic tie-breaking rule. Instead of random selection, which introduces instability, we resolve ties by selecting the community whose highest-ranked node (based on our proposed centrality metric) has the greatest centrality:

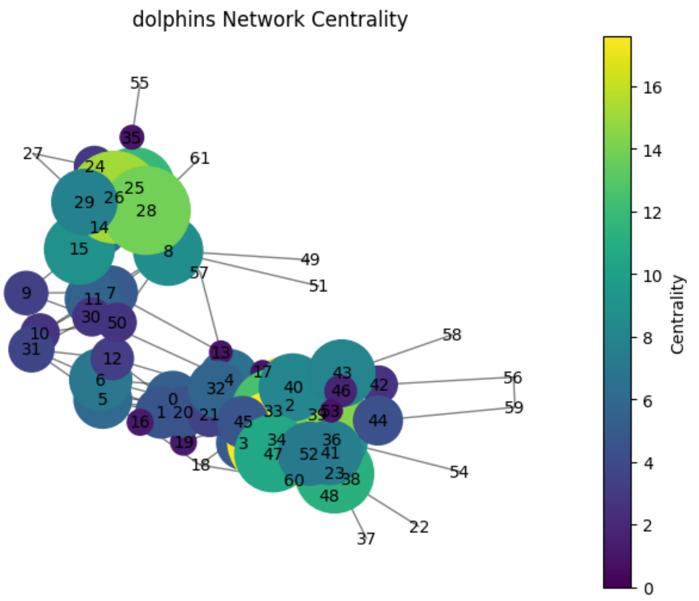
$$\ell_{\text{new}} = \arg \max_{k \in \mathcal{K}_{\max}} s(k) \tag{4.4}$$

where $\mathcal{K}_{\max}$ is the set of labels with the maximum influence, and $s(k)$ represents the centrality of the highest-ranked node within community k. This approach eliminates stochastic decisions, thereby improving the reproducibility and stability of the community detection results, while preserving structural consistency within the network.

In summary, node centrality (Equation 4.2) quantifies structural influence through Adamic-Adar-weighted connections. During label updates, Equation 4.4 resolves ties by selecting the community containing the most central node.



(a) Karate Club: nodes sized and colored by AA centrality; color bar indicates score scale.



(b) Dolphins network: nodes sized and colored by AA centrality; color bar indicates score scale.

Figure 4.1—Visualization of AA centrality on two benchmark networks.

Algorithm 3 gives the full pseudocode for AA-LPA. By unifying local shared-neighbor information with global community size signals and enforcing a fixed update order.

Algorithm 3 Adamic-Adar LPA (AA-LPA)

```
1: Input: Graph $\mathcal{G} = (V, E)$, maximum iterations $T$
2: Output: Node labels $\ell$
3: Compute AA scores for all edges via Equation 4.1.    ▷ for all $u \in V$
4: Calculate node centralities $s(u)$ via Equation 4.2
5: Initialize $\ell(u) \leftarrow u \ \forall u \in V$    ▷ Each node starts with its own unique label
6: for $t = 1$ to $T$ do
7:     Sort descending $V$ by $s(u)$,    ▷ Process most central nodes first
8:     Compute $|C_k|$ for all current communities    ▷ Find the size of each community
9:     for each $u$ in sorted order do
10:        Initialize $w_k = 0 \ \forall k$    ▷ $w_k$: weight accumulator for community $k$
11:        for each $v \in \mathcal{N}(u)$ do    ▷ For each neighbor of $u$
12:            $w_{\ell(v)} \leftarrow w_{\ell(v)} + \text{AA}(u, v) \times |C_{\ell(v)}|$    ▷ Accumulate weighted vote
13:        end for
14:        $\mathcal{K}_{\text{max}} \leftarrow \{k \mid w_k = \max(w)\}$    ▷ Find communities with max weight
15:        if $|\mathcal{K}_{\text{max}}| > 1$ then    ▷ If there is a tie
16:            $\ell_{\text{new}} \leftarrow \arg\max_{k \in \mathcal{K}_{\text{max}}} s(k)$ ▷ Break it by choosing the community with the
    most central node
17:        else
18:            $\ell_{\text{new}} \leftarrow \mathcal{K}_{\text{max}}[0]$    ▷ Otherwise, choose the winner
19:        end if
20:        if $\ell_{\text{new}} \neq \ell(u)$ then
21:            Update $\ell(u) \leftarrow \ell_{\text{new}}$    ▷ Update the node's label if it changed
22:        end if
23:    end for
24:    if Label convergence achieved then
25:        break
26:    end if
27: end for
28: return $\ell$    ▷ Return final community assignments
```

The time complexity of the algorithm decomposes as follows. First, for **AA preprocessing**, computing Adamic-Adar scores for every edge takes $O(|E|\ \bar{d}^2)$, where $\bar{d}$ is the average degree of the graph. Second, **computation of centrality**, which involves summing the precomputed AA scores to obtain the centrality of each node, according to equation 4.2, requires $O(|E|)$. Third, the **iteration phase** over T rounds. Each update step within a round incurs a complexity of $O(|V| \log |V| + |E|)$, primarily dominated by sorting the nodes according to their centrality and traversing their neighbors for label aggregation. Overall, the algorithm runs in

$$O\Big(|E|\,\bar{d}^2 \;+\; T\big(|V|\log|V| + |E|\big)\Big).$$

To illustrate the interpretable benefits and ensure a strong understanding of AA-LPA's deterministic nature, a step-by-step example is presented below. The algorithm primarily consists of two main phases:

Precomputing Phase: This initial phase involves two key calculations:

1. **AA Calculation:** The Adamic-Adar (AA) index is computed for all edges in the network. This index quantifies the strength of connections by considering the number and informativeness of shared neighbors. Figure 4.2a presents an example graph with AA scores computed for each edge (Equation 4.1).

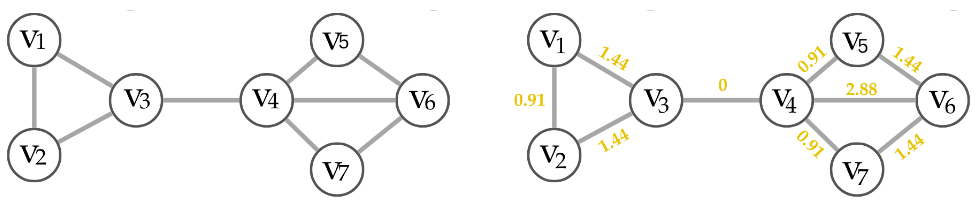
2. **Node Centrality Calculation:** For each node, its centrality score is determined as the sum of the Adamic-Adar scores across all its incident edges. This implies that nodes connected to many "informative" neighbors (i.e., neighbors contributing to high AA scores) will naturally possess higher centrality scores (Equation 4.2). Figure 4.2b further illustrates this calculation.

Enhanced Label Propagation Phase: This second phase incorporates several deterministic mechanisms to guide label propagation:

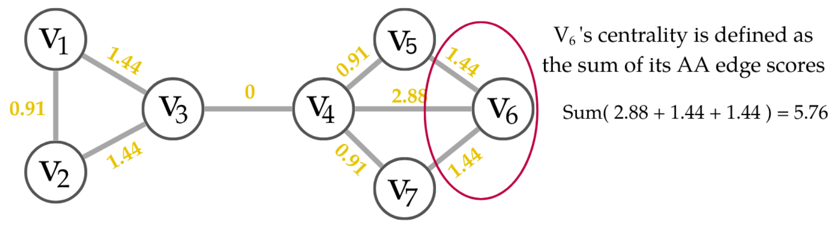
1. **Node Prioritization:** All nodes are sorted in descending order based on their precomputed AA-derived centrality scores. These top-scoring nodes are considered influential, and processing them first guides the label propagation process more effectively and efficiently, fostering quicker convergence.

2. **Label Update and Deterministic Tie Resolution:** Labels are iteratively updated based on the Adamic-Adar scores. When a node adopts a label from a neighbor, the AA score dictates the influence, giving preference to connections with stronger shared-neighbor contexts. For example, consider node V_6 with three neighbors V_4, V_5, V_7. If V_4's connection to V_6 yields the highest Adamic-Adar score among V_6's neighbors, then V_6 will adopt V_4's label (Figure 4.2c). Crucially, in scenarios where multiple neighbors present identical scores (ties), AA-LPA employs a deterministic tie-breaking mechanism. The node with the highest precomputed

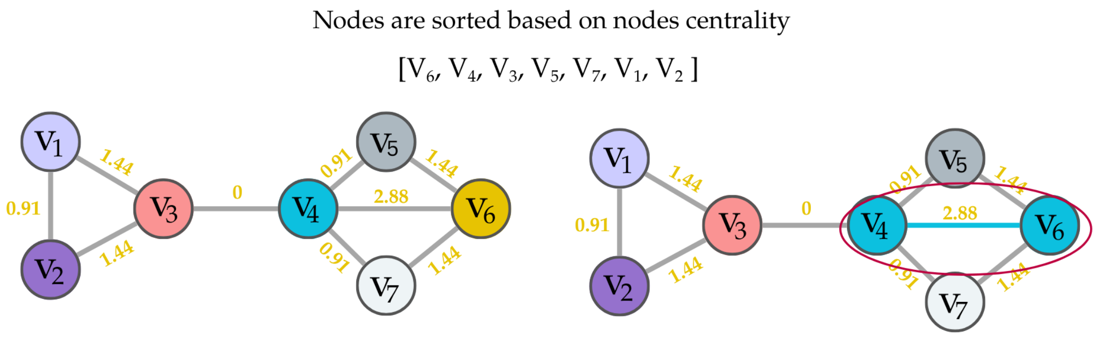
centrality score from the globally sorted list, among the tied neighbors, is chosen. This ensures that the label choice is always clear and consistently based on the inherent network structure, effectively eliminating any element of randomness. An illustrative example of this propagation and tie-breaking process is provided in Figure 4.2d.



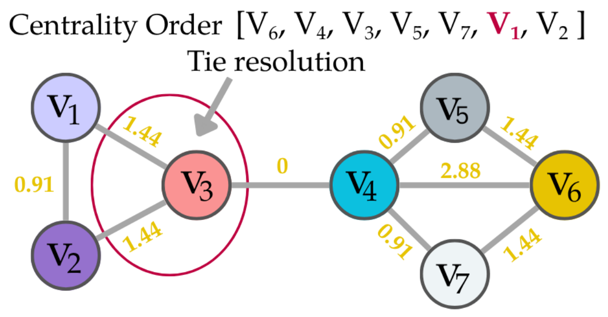
(a) Adamic–Adar edge-weight calculation: (left) original simple graph; (right) same graph with AA-weighted edges computed using AA formula.


V₆'s centrality is defined as the sum of its AA edge scores
Sum(2.88 + 1.44 + 1.44) = 5.76

(b) Node centrality computation: each node's centrality score $s(u)$ is computed as the sum of its incident AA-weighted edges.


Nodes are sorted based on nodes centrality
[V6, V4, V3, V5, V7, V1, V2]

(c) Propagation order and label updates:(left) nodes are initialized with unique labels and sorted by descending centrality; (right) nodes then update sequentially in that order, each choosing the neighbor with the highest AA-weighted edge as its new label.



(d) Deterministic tie resolution: when two neighbors have equal AA influence, the tie is broken by selecting the label of the node that appears first in the centrality ranking.

Figure 4.2—Step-by-step illustrative example of AA-LPA.

4.1.3 Experimental Analysis

In this section, we present a comprehensive experimental evaluation, detailing the setup, dataset characteristics, and implementation environment, followed by a thorough presentation of results and their interpretations.

4.1.3.1 Experimental setup

The free version of Google Colab serves as the computational environment for this study, offering specifications that include approximately 100 GB of disk space and 12.7 GB of RAM. These resources provide sufficient capacity to support the execution and analysis of the source code, which has been made publicly available via QR[1] in Figure 4.3.

Figure 4.3—QR code linking to the AA-LPA GitHub repository.

Three widely used ground-truth network datasets Zachary's Karate Club, Dolphins, and American Football are employed in this study. A detailed description of each dataset is provided in Section 2.1.2.

4.1.3.2 Results and Discussion

The results in Table 4.1 show that AA-LPA delivers competitive or superior performance relative to state-of-the-art community detection methods when measured against ground truth labels. On the Karate Club network, AA-LPA achieves an F1 score of 0.970 and an NMI of 0.837, demonstrating its capacity to disambiguate small-scale social structures by propagating labels from the most influential nodes as determined by AA centrality. Although its modularity (Q) is on par with LPA-S and Leiden, the higher NMI and F1 score underscore its strength in recovering the known partition.

On the Dolphins network, AA-LPA attains perfect alignment (NMI = 1.0, F1 = 1.0), outperforming all baselines. This result highlights how the combination of AA-weighted edges and community-size amplification overcomes LPA's tendency toward fragmented

[1]https://github.com/abdelfateh10/AA-LPA

clusters, steering label updates toward stable, cohesive communities. The marginally lower modularity (0.373 versus Leiden's 0.527) reflects a deliberate trade-off: AA-LPA favors fidelity to the ground truth over maximizing density-based modularity, which can conflate structural tightness with functional groupings.

For the Football network, AA-LPA again leads on NMI (0.928) and F1 (0.931). The deterministic tie-breaking rule resolving equal-weight conflicts by comparing node centralities ensures consistent label flows even in finely balanced structures. While modularity remains similar to Louvain and Leiden, the marked gains in NMI and F1 confirm that optimizing modularity alone does not guarantee alignment with semantic communities.

These findings validate our core hypothesis: standard LPA's random update order and uniform label weighting often yield unstable or fragmented partitions. By integrating AA-derived centrality for node ordering and edge weighting within a deterministic method, AA-LPA injects structural insight into propagation and guarantees reproducibility. Although AA preprocessing incurs extra overhead potentially limiting scalability to very large graphs, future work on approximate scoring or parallel implementations could mitigate this. Moreover, while AA-LPA excels in networks where shared neighbors indicate functional ties (e.g., social or biological systems), its efficacy on highly degree-heterogeneous or scale-free graphs merits further study. Overall, the consistent improvements in NMI and F1 across diverse benchmarks underscore AA-LPA's value for interpretable, ground truth aligned community detection.

Table 4.1—Comparison of Modularity (Q), NMI, and F1 scores against ground-truth communities

Notes: Walktrap (Pons and Latapy, 2005), Infomap (Rosvall and Bergstrom, 2008), CNM (Clauset et al., 2004), Louvain (Blondel et al., 2008), Leiden (Traag et al., 2019), SimCMR (Tunali, 2021), LPAc (Zhang et al., 2014), FLPA (Traag and Šubelj, 2023), LPA (Raghavan et al., 2007), LPA-S (Douadi et al., 2024).

Network	Measures	Walktrap	Infomap	CNM	Louvain	Leiden	SimCMR	LPAc	FLPA	LPA	LPA-S	our
Karate	Q	0.353	0.402	0.381	0.419	0.420	0.395	0.383	0.267	0.353	0.371	**0.371**
	NMI	0.504	0.699	0.692	0.659	0.677	0.696	0.666	0.571	0.545	1.00	0.837
	F1	0.609	0.811	0.789	0.721	0.725	0.816	0.727	0.813	0.717	**1.00**	0.970
Dolphins	Q	0.489	0.523	0.495	0.519	**0.527**	0.526	0.488	0.497	0.466	0.511	0.373
	NMI	0.582	0.547	0.557	0.479	0.577	0.581	0.538	0.577	0.586	0.588	**1.0**
	F1	0.609	0.811	0.789	0.721	0.725	0816	0.727	0.813	0.717	**1.0**	**1.0**
Football	Q	0.603	0.594	0.550	0.604	0.605	0.603	0.587	0.597	0.584	0.581	0.584
	NMI	0.887	0.858	0.698	0.882	0.990	0.897	0.899	0.893	0.994	0.910	**0.928**
	F1	0.868	0.824	0.655	0.861	0.870	0.877	0.851	0.862	0.848	0.853	**0.931**

4.1.4 Summary of AA-LPA Algorithm

The classic Label Propagation Algorithm suffers from unstable and fragmented outputs due to its inherent randomness, undermining its reliability for applications demanding consistent community assignments. In this work, we introduced AA-LPA, a fully deterministic variant that leverages Adamic–Adar centrality to order node updates, weight label votes, and resolve ties. Experiments on benchmark networks show that AA-LPA achieves stronger alignment with ground truth communities reflected in higher NMI and F1 scores by seeding propagation from structurally influential nodes and amplifying community aware label influence. The resulting partitions are both stable and interpretable, making AA-LPA well suited for domains such as social network analysis and biological module detection, where reproducibility and semantic coherence are essential.

Future work will explore scalable approximations for AA scoring to alleviate the $O(|E|\bar{d}^2)$ preprocessing cost on large graphs, as well as extensions to heterogeneous and attributed networks via multimodal similarity measures. We also plan to develop dynamic variants that update centrality scores incrementally for streaming data, and to conduct a rigorous theoretical analysis of convergence under AA-weighted rules. Finally, integrating AA-LPA with modularity based methods and benchmarking on very large real world graphs will help bridge the gap between structural accuracy and practical scalability.

4.2 Graph-Attention Autoencoder in Attributed Networks

In recent years, community detection has benefited from advances in deep learning, most notably through Graph Neural Networks (GNNs) (Su et al., 2022), which are capable of learning rich, nonlinear representations of complex, high-dimensional network data. Among these, Graph Autoencoders (GAEs) (Thomas and Welling, 2016; Veličković et al., 2018) have emerged as a flexible framework: they compress graph structure and node attributes into low-dimensional embeddings and then reconstruct the original graph via a decoder, preserving important topological and feature information.

A key enhancement to GNNs is the incorporation of attention mechanisms. Graph Attention Networks (GATs) (Vaswani et al., 2017; Veličković et al., 2018) extend convolutional GNNs by computing attention coefficients that dynamically weight each neighbor's contribution during message passing. Multi-head attention further allows the model to capture diverse semantic relationships by attending to different subspaces in parallel.

Graph Autoencoders (GAEs) extend traditional autoencoders to graph-structured data and have been adapted for attributed network problems related to community detection. Several GAE variants leverage the Graph Attention Network (GAT) architecture to improve feature representation and incorporate an additional clustering objective, often using the Kullback–Leibler divergence as a loss term (Wang et al., 2019; Zhou et al., 2023). Despite these advances, significant challenges remain (Liu et al., 2021): classical methods still struggle to jointly capture both structural and attribute information, scalability issues arise in large networks, performance degrades in the presence of noise or overlapping communities, and many models do not generalize well across diverse network types and configurations.

To address these gaps, we propose *Graph Attention Autoencoder Clustering-Oriented* (G2ACO) (Bekkair et al., 2025a), an unsupervised model that combines a multi-head graph attention encoder with an inner-product decoder and an explicit clustering objective. Our contributions are threefold:

1. **G2ACO Architecture.** We design a graph autoencoder whose encoder uses multi-head attention to produce node embeddings that capture both structural and attribute information; the decoder reconstructs edges via an inner-product layer.

2. **Clustering Objective.** We introduce a K-means loss term to the training objective, which encourages embeddings to form tight, well-separated clusters. This directs the autoencoder toward community-aware representations by minimizing intra-cluster distances and maximizing inter-cluster separation.

3. **Empirical Validation.** We evaluate G2ACO on four benchmark attributed networks Cora, CiteSeer, and PubMed as a citation network and BlogCatalog as a social network and demonstrate that it outperforms competitive GAE- and GAT-based clustering methods in terms of modularity, NMI, and F1 scores.

4.2.1 G2ACO

This section presents an overview of our G2ACO architecture, which learns meaningful graph representations for unsupervised community detection. G2ACO consists of a multi-head graph attention encoder and an inner-product decoder (Figure 4.4). Given the adjacency matrix $\mathbf{A}$ and the attribute matrix $\mathbf{X}$, the autoencoder efficiently projects the graph into a low-dimensional latent space. Our model comprises three main components, an encoder, a decoder, and a composite loss function that combines reconstruction and clustering objectives. In the following, we describe each component in detail.

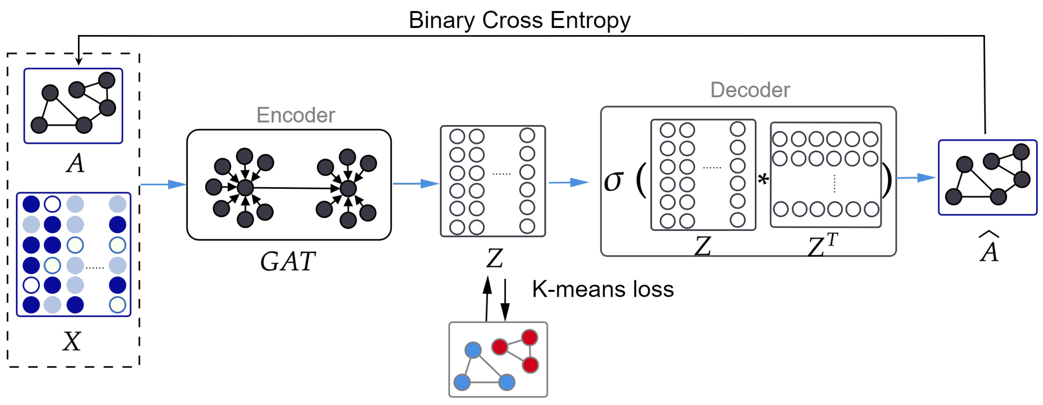


Figure 4.4—G2ACO architecture for unsupervised community detection. The multi-head attention encoder fuses structural ($\mathbf{A}$) and attribute ($\mathbf{X}$) inputs into latent embeddings, while the inner-product decoder reconstructs the graph. A dual-objective loss combines reconstruction loss with a K-means clustering term to encourage cohesive community representations.

4.2.1.1 Encoder

Our model's encoder component is built upon the principles of Graph Autoencoders (GAE) for robust representation generation.Specifically, we employ a Graph Attention Network (GAT) with a multi-head attention mechanism. This choice is motivated by GAT's proven ability to adaptively capture semantic information across multiple attention heads through trainable weights, enabling a more robust analysis of complex graph-structured data compared to other architectures like Graph Convolutional Networks (GCNs) (Vaswani et al., 2017).

The GAT encoder processes input graph data to learn expressive node representations. The updated representation of node i in layer l, denoted as $\mathbf{z}_i^{(l)}$, is determined by computing a weighted sum of the hidden states $\mathbf{z}_j^{(l-1)}$ of its neighboring nodes. The attention coefficients α_{ij}^h serve as trainable weights for the h-th attention head, allowing the network to selectively prioritize significant neighbors during the aggregation process Veličković et al. (2018). The formal definition of the node update rule in a GAT layer is given by Equation 4.5.

$$\mathbf{z}_i^{(l)} = \big\Vert_{h=1}^{H} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{h,l}\mathbf{W}^{h,l}\mathbf{z}_j^{(l-1)}\right). \tag{4.5}$$

Here, $\Vert$ denotes concatenation across the H attention heads, σ is a non-linear activation function, $\mathcal{N}_i$ represents the set of neighbors of node i, and $\mathbf{W}^{h,l} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ is the learnable weight matrix for head h in layer l.

The core of the GAT encoder lies in its attention mechanism, which effectively mediates the interaction between attribute and structural information by dynamically reconciling node feature transformations and topological constraints. In the attribute space, node features $\mathbf{z}_j^{(l-1)}$ are projected into latent compatibility subspaces through learnable weight matrices $\mathbf{W}^{h,l}$. Simultaneously, the structural space restricts aggregation to only topologically connected neighbors $\mathcal{N}_i$. The attention coefficients $\alpha_{ij}^{h,l}$ resolve this interaction through a dual constraint system: structural connectivity enforces hard neighborhood filtering ($j \in \mathcal{N}_i$), while attribute compatibility scores $e_{ij}^{h,l}$ determine soft influence weights. The use of parallel attention heads ($h = 1, \ldots, H$) allows independent transformations $\mathbf{W}^{h,l}$ to jointly capture various interaction patterns and explore multiple latent subspaces.

The attention mechanism begins by computing the unnormalized coefficients $e_{ij}^{h,l}$ for the pairs of nodes (i, j) in layer l, as defined in Equation 4.6.

$$e_{ij}^{h,l} = \text{LeakyReLU}\left(a^{h,l}\left(\mathbf{W}^{h,l}\mathbf{z}_i^{(l-1)}, \mathbf{W}^{h,l}\mathbf{z}_j^{(l-1)}\right)\right) \tag{4.6}$$

where $\mathbf{z}_i^{(l-1)}, \mathbf{z}_j^{(l-1)} \in \mathbb{R}^{d^{(l-1)}}$ are the input embeddings from layer $l-1$. The function $a^{h,l} : \mathbb{R}^{d^{(l)}} \times \mathbb{R}^{d^{(l)}} \to \mathbb{R}$ is a learnable function that maps projected features to a scalar compatibility score. The LeakyReLU activation function, with a small positive slope $\alpha > 0$, introduces non-linearity and allows for the modeling of asymmetric interactions between nodes.

These unnormalized coefficients are then normalized via a softmax function over the neighborhood $\mathcal{N}_i$ to produce interpretable attention weights, as shown in Equation 4.7.

$$\alpha_{ij}^{h,l} = \frac{\exp(e_{ij}^{h,l})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik}^{h,l})} \tag{4.7}$$

Equation 4.7 guarantees that $\alpha_{ij}^{h,l} \in [0, 1]$ and $\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{h,l} = 1$. This normalization ensures that the aggregated features in Equation 4.5 accurately reflect the proportional influence of each neighbor on the central node's representation.

The objective of the GAT encoder is to train node representations across two layers, effectively embedding comprehensive information from both the initial attribute data ($\mathbf{X}$) and the network's topological structure ($\mathbf{A}$). This process transforms the high-dimensional input data into a lower-dimensional latent space, denoted by $\mathbf{Z}$, where $\mathbf{Z} \in \mathbb{R}^{N \times d}$. Here, N represents the total number of nodes in the network, and d refers to the dimension of the obtained latent space. The transformation is formally expressed by the function $\phi_{enc}(\mathbf{A}, \mathbf{X})$, resulting in a compact and informative representation $\mathbf{Z}$ in this low-dimensional space.

To enhance the model's generalization capabilities and mitigate overfitting, the dropout technique is employed at both layers of the GAT encoder, with varying dropout rates applied to each layer.

4.2.1.2 Decoder

The decoder serves as the inverse of the encoder, taking the compressed, latent representation generated by the encoder and transforming it back into a reconstructed version of the original input data. Its primary role is to effectively "uncompress" the encoded information, aiming to restore the data's original form as accurately as possible.

Specifically, the decoder applies a function, denoted as $\phi_{dec}(\mathbf{Z})$, to project the encoded information, $\mathbf{Z}$, from the latent space back into the original data space. This projection yields a reconstructed representation, referred to as $\hat{\mathbf{A}}$, which in our context signifies the reconstructed topological structure of the input graph.

For the G2ACO model, we have chosen a simple inner product decoder, also commonly known as a dot product decoder. This choice is motivated by its interpretability, computational efficiency and its effectiveness in reconstructing graph structures, particularly in scenarios where node similarities can be directly inferred from the dot product of their latent representations. The formulation of our decoder is presented in Equation 4.8:

$$\hat{\mathbf{A}} = \sigma(\mathbf{Z} \cdot \mathbf{Z}^T) \tag{4.8}$$

where $\sigma(\cdot)$ represents the sigmoid activation function. The sigmoid function ensures that the output values of $\hat{\mathbf{A}}$ are scaled between 0 and 1, which is crucial for interpreting $\hat{\mathbf{A}}$ as a probability distribution representing the likelihood of edges in the reconstructed graph. The matrix multiplication $\mathbf{Z} \cdot \mathbf{Z}^T$ computes the inner product between all pairs of node embeddings in the latent space $\mathbf{Z}$, effectively measuring their similarity, which is then transformed into edge probabilities by the Sigmoid function. This simple yet powerful decoder design allows for an efficient and reconstruction of the graph's adjacency matrix.

4.2.1.3 Optimization

Optimizing a deep learning model for community discovery is essential to effectively extract meaningful representations from graph data. The optimization strategy of our model uniquely combines reconstruction optimization with clustering-oriented information within a unified objective function. This objective is refined iteratively during each training epoch. The overall optimization goal is to minimize the composite loss function, $\mathcal{L}$, defined as:

$$\mathcal{L} = \lambda\,\mathcal{L}_r \;+\; \beta\,\mathcal{L}_c. \tag{4.9}$$

Here, $\mathcal{L}_r$ represents the reconstruction loss, and $\mathcal{L}_c$ denotes the clustering loss. Both λ and β are positive hyperparameters. λ controls the significance of the reconstruction component, while β adjusts the importance of the clustering optimization.

The primary objective of the G2ACO model is to minimize the discrepancy between the original network structure, denoted as $\mathbf{A}$, and the reconstructed adjacency matrix, $\hat{\mathbf{A}}$, generated by the autoencoder. This is achieved using the binary cross-entropy loss function, which quantifies the dissimilarity between these two probability distributions:

$$\mathcal{L}_r = -\frac{1}{N}\sum_{i=1}^{N}\left[\mathbf{A}_i \cdot \log(\hat{\mathbf{A}}_i) + (1-\mathbf{A}_i)\cdot \log(1-\hat{\mathbf{A}}_i)\right]$$

Through this minimization, the model refines its parameters to accurately capture and reproduce the underlying network structure.

In addition to minimizing reconstruction error, our model incorporates an objective of forming the latent representation to be highly conducive to community detection tasks. This is achieved by guiding the autoencoder to generate representations suitable for clustering through a dedicated clustering loss. This loss function helps map the data in the latent space in a manner that promotes clear cluster formation.

By integrating the K-means loss into the G2ACO model, the learning process is directed toward capturing the intrinsic structure of the data while simultaneously promoting the formation of cohesive node groups (communities). This dual optimization allows the model to achieve improved performance in community detection by simultaneously reconstructing the input data and optimizing for the community structure, resulting in a robust solution. The K-means loss functions serve as a guiding mechanism, directing the learning process towards revealing significant community patterns within the encoded representations.

The K-means clustering loss, $\mathcal{L}_c$, is calculated by summing the distances between each data point in the latent space, **Z**, and its nearest cluster centroid:

$$\mathcal{L}_c = \frac{1}{N}\sum_{i=1}^{N} \min_{j=1,\ldots,k} \|\mathbf{z}_i - \mu_j\|,$$

where, $\|\cdot\|$ represents the Euclidean distance, $\mathbf{z}_i$ denotes the i-th data point in **Z**, μ_j represents the centroid of the j-th cluster, N signifies the total number of data points, and k reflects the number of clusters.

The model parameters θ (for the encoder) are updated using the Adam optimizer to minimize the composite loss $\mathcal{L}$. The gradients of $\mathcal{L}_c$ with respect to θ are computed as:

$$\nabla_\theta \mathcal{L}_c = \frac{2\beta}{N}\sum_{i=1}^{N}(\mathbf{z}_i - \mu_{j^*}) \cdot \nabla_\theta \mathbf{z}_i,$$

where μ_{j^*} corresponds to the nearest centroid for $\mathbf{z}_i$. Crucially, a detached copy of **Z** is used to decouple gradient computation from cluster assignments. The centroids $\{\mu_j\}$ are treated as constants during backpropagation and are updated (every epoch) using the latest embeddings through a non-differentiable K-means step. The Adam optimizer adaptively adjusts the learning rate for θ using estimates of the first and second moments of the gradients, ensuring stable convergence despite this alternating optimization strategy.

4.2.2 Experiments

This section details the experimental evaluation of G2ACO. We assess its performance by applying K-means clustering to the learned node embeddings **Z** for community detection. The subsequent subsections delineate the experimental setup, present the results and their discussion, provide a comparative analysis against competing models, and describe the ablation studies conducted.

The G2ACO implementation is publicly available online[2] and can also be accessed directly by scanning the QR code in Figure 4.5.

[2]https://github.com/abdelfateh10/G2ACO

Figure 4.5—QR code linking to the G2ACO GitHub repository.

4.2.2.1 Experimental Settings

This subsection outlines the experimental environment, evaluation metrics, datasets utilized, and the methodology for parameter selection that configured our experiments.

All experiments were executed within the Google Colab environment, provisioned with 12.7 GB of RAM and a GPU offering 15 GB of memory. The model implementation relied on the PyTorch Geometric library, a widely recognized toolkit for graph-based deep learning.

The quality of community detection was quantitatively assessed using two standard metrics, NMI and ARI.

Our method was evaluated in three distinct citation networks: *Cora*, *CiteSeer*, and *PubMed*, where nodes represent documents and edges denote citations. Furthermore, the *BlogCatalog* social network was included, characterized by nodes representing blogs, edges signifying social interactions, and node attributes comprising tag vectors. Detailed specifications for these datasets are provided in Section 2.1.2.

All hyperparameters were systematically selected through preliminary empirical experimentation. Our GAT encoder architecture comprises a hidden layer with 256 units and produces 64-dimensional embeddings. For the citation networks $($Cora, CiteSeer, PubMed$)$, 8 attention heads were employed, while the BlogCatalog network utilized 2 attention heads.

Model optimization was performed using the Adam optimizer. The clustering weight β was set to 0.1 for Cora and CiteSeer, 10 for PubMed, and 1 for BlogCatalog. Conversely, the reconstruction weight λ was configured to 1 for all three citation datasets and 0.3 for BlogCatalog.

To regularize the encoder, dropout was applied at each GAT layer, with dataset-specific rates determined empirically.

4.2.2.2 Competitive Models

To evaluate G2ACO, we compare it with a diverse set of established community detection methods that operate on the same benchmark datasets. These baselines include classical algorithms, traditional machine learning techniques, and deep learning-based graph autoencoders:

- **K-means** (Krishna and Murty, 1999): Clusters nodes solely on their attribute vectors by iteratively minimizing intra-cluster variance and maximizing inter-cluster separation.
- **DeepWalk** (Perozzi et al., 2014): Learns node embeddings via truncated random walks and Word2Vec, capturing network structure for downstream tasks such as multi-label classification.
- **TADW** (Yang et al., 2015): Extends DeepWalk by integrating textual node attributes with topology via matrix factorization.
- **GAE & VGAE** (Thomas and Welling, 2016): Utilize GCN-based encoders and inner-product decoders for unsupervised graph embedding; VGAE adds a variational inference objective.
- **MGAE** (Wang et al., 2017a): Applies marginalized graph convolution with noise-corrupted node features, followed by spectral clustering on the learned embeddings.
- **ARGA & ARVGA** (Pan et al., 2018): Introduce adversarial regularization into GAE and VGAE to improve embedding robustness.
- **DAEGC** (Wang et al., 2019): Combines GAT-based embedding with self-clustering objectives, jointly optimizing reconstruction and clustering losses.
- **GATE** (Salehi and Davulcu, 2020): Incorporates self-attention layers in both encoder and decoder to jointly reconstruct features and graph structure.
- **SENet** (Zhang et al., 2021): Merges spectral clustering loss with higher-order convolutions on a kernel matrix derived from shared-neighbor similarities.
- **VGAER** (Qiu et al., 2022): Enhances VGAE by including modularity information alongside adjacency in its variational framework.

- **CDBNE** (Zhou et al., 2023): Utilizes a GAT-based autoencoder with modularity maximization and self-training via KL divergence for community discovery.

4.2.2.3 Ablation Analysis

To quantitatively ascertain the individual contributions of each component within G2ACO, we conduct a series of ablation studies. Specifically, we isolate the impact of the clustering loss, examine the effect of key hyperparameters, assess the multi-head attention mechanism, and evaluate the significance of incorporating node attributes.

Clustering Loss Ablation

Figure 4.6 presents a comparison of NMI and ARI for the full G2ACO model against a variant devoid of the clustering objective (denoted $\text{G2ACO}_{\text{NoClust}}$) across all benchmark networks. The inclusion of the clustering loss demonstrably improves both metrics by over 10%, underscoring its critical role in shaping informative latent embeddings.

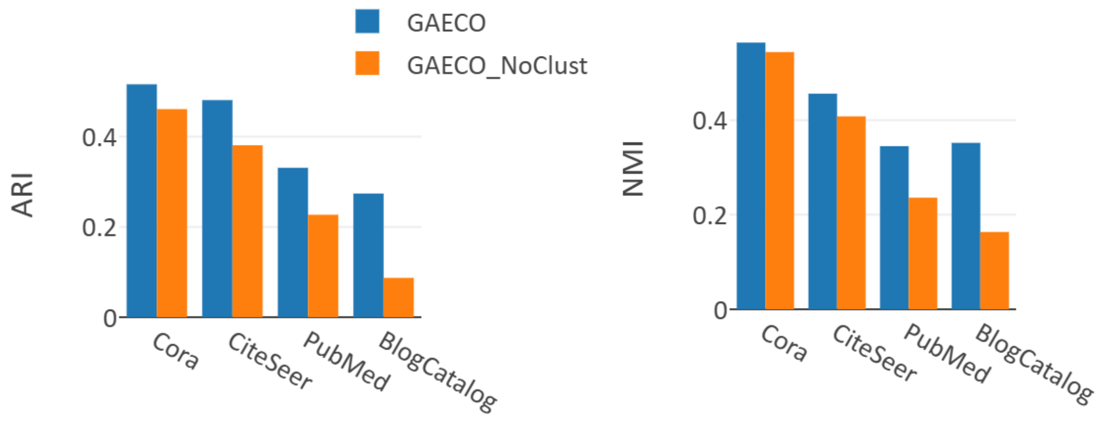


Figure 4.6—Effect of the clustering loss on ARI and NMI. $\text{G2ACO}_{\text{NoClust}}$ omits the $\mathcal{L}_c$ term.

Hyperparameter Sensitivity

We next systematically vary the reconstruction weight λ and clustering weight β to gauge their influence on model performance. Figure 4.7 illustrates the NMI and ARI obtained for the PubMed and BlogCatalog networks under different parameter settings. Optimal results were observed at $(\lambda, \beta) = (1, 10)$ for PubMed and $(0.3, 1)$ for BlogCatalog, indicating that dataset-specific tuning of these hyperparameters is beneficial.

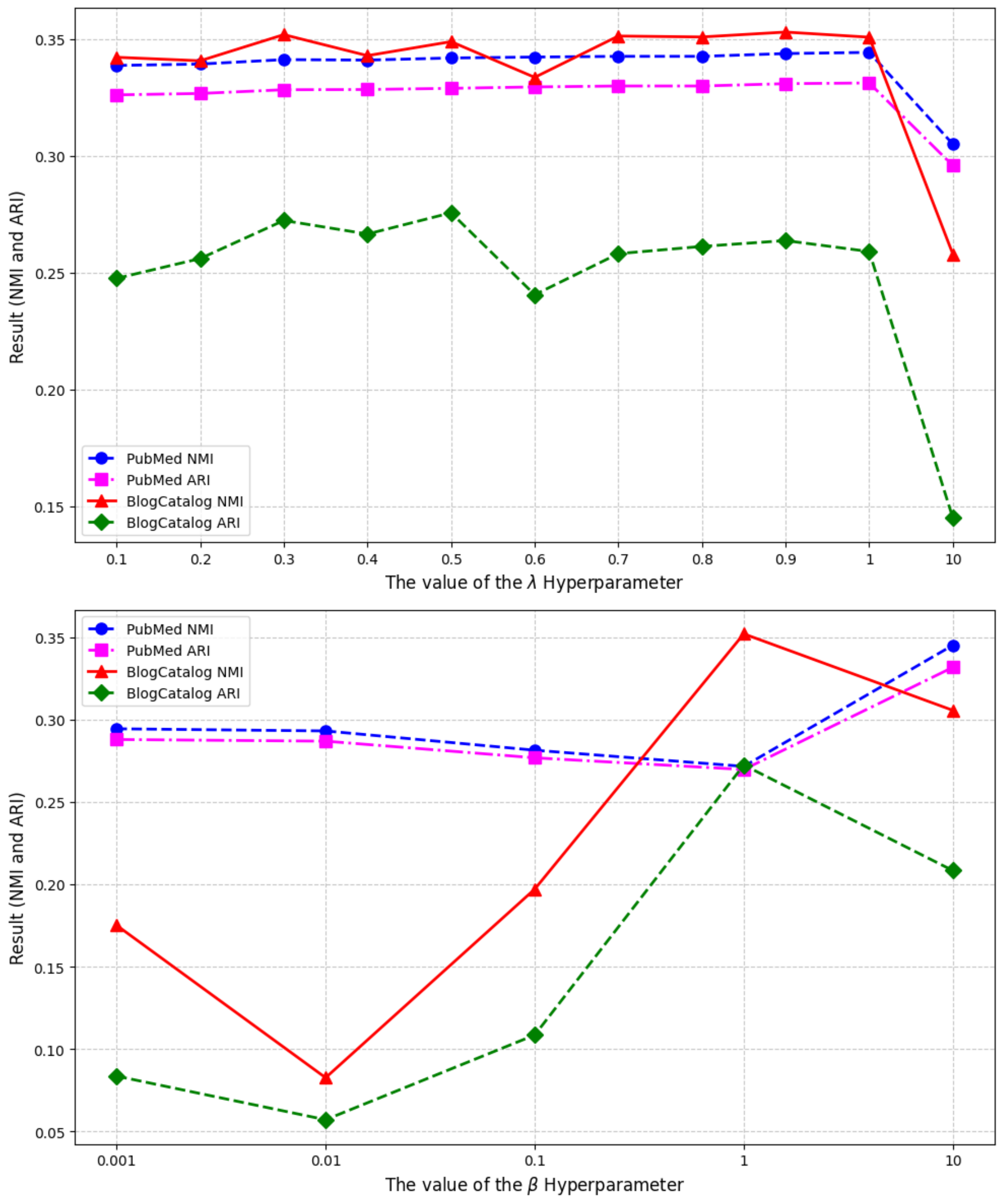


Figure 4.7—NMI and ARI across different (λ, β) settings on PubMed and BlogCatalog.

Attention Head Ablation

To thoroughly assess the multi-head attention mechanism, we compare the performance of a single-head configuration against multiple heads $(2, 4, 8, 16)$ while maintaining a fixed hidden size of 256 units and an embedding size of 64 dimensions. As depicted in Figure 4.8, citation networks generally benefit most from 8 attention heads, whereas the social network (BlogCatalog) achieves peak performance with 2 heads, highlighting that the optimal number of attention heads is contingent upon graph characteristics.

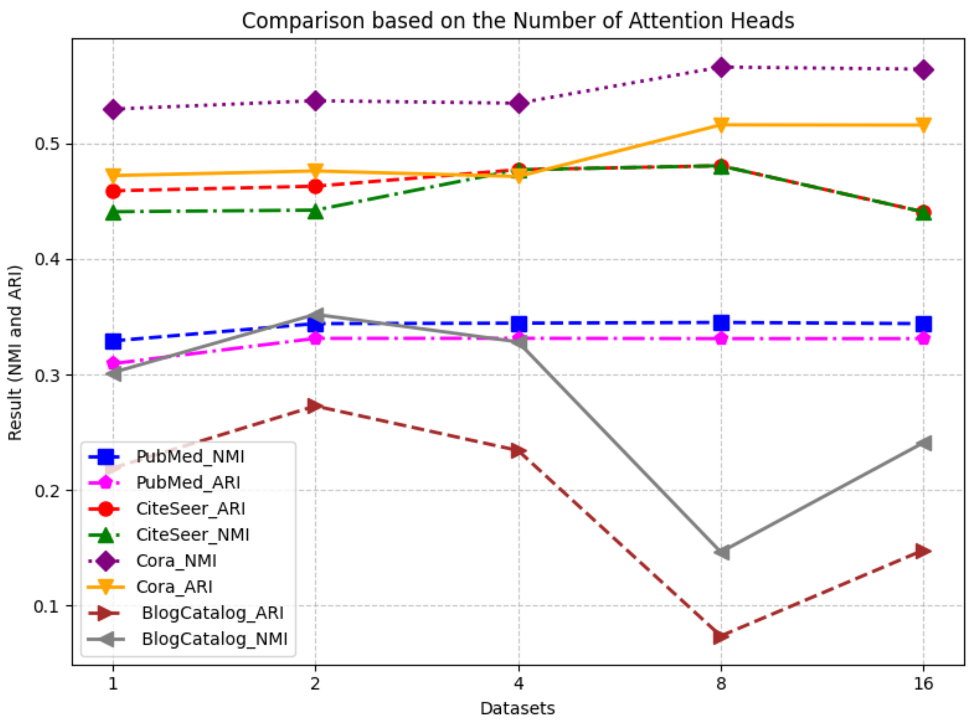


Figure 4.8—Impact of number of attention heads on NMI and ARI. Hidden layer = 256, embedding = 64.

Role of Node Attributes

Finally, we evaluate G2ACO's performance when utilizing only the adjacency matrix (denoted G2ACO$_{\text{NoAtt}}$, effectively setting $\mathbf{X} = \mathbf{I}$) versus employing both adjacency and node attributes. Table 4.2 presents the NMI and ARI scores across four datasets for these two configurations. The results unequivocally demonstrate that including node attributes substantially boosts both metrics, thereby confirming the critical importance of attribute information for effective community detection.

Table 4.2—NMI and ARI for G2ACO with and without node attributes.

Dataset	Metric	G2ACO$_{\text{NoAtt}}$	G2ACO
		$\mathbf{A}$	$\mathbf{A}, \mathbf{X}$
Cora	NMI	0.223	**0.564**
	ARI	0.162	**0.516**
CiteSeer	NMI	0.077	**0.456**
	ARI	0.065	**0.481**
PubMed	NMI	0.003	**0.345**
	ARI	0.009	**0.331**
BlogCatalog	NMI	0.164	**0.352**
	ARI	0.087	**0.273**

4.2.3 Results and Discussion

We evaluate community detection performance on Cora, CiteSeer, PubMed, and BlogCatalog datasets using NMI and ARI. The detailed results are reported in Tables 4.3, 4.4, 4.5, and 4.6, respectively. Each table also specifies the input data utilized: topology ($\mathbf{A}$), attributes ($\mathbf{X}$), and modularity matrix ($\mathbf{B}$), and highlights the best-performing methods.

Across all datasets, G2ACO consistently outperforms the strongest baseline, CDBNE. In terms of NMI, G2ACO achieves at least a 3% lead on Cora and BlogCatalog, a 2% lead on CiteSeer, and a 1% lead on PubMed. For ARI, G2ACO's performance is superior by at least 0.3% on Cora, 3% on CiteSeer, and 0.1% on BlogCatalog. On PubMed, however, G2ACO's ARI is marginally lower than CDBNE, with a difference of 0.06%.

These observed performance disparities reflect intrinsic differences among the datasets and underscore the critical role of attribute information. Cora and CiteSeer are networks of intermediate scale characterized by rich node features, which facilitate the effective joint utilization of both $\mathbf{A}$ and $\mathbf{X}$. Conversely, PubMed, being a relatively larger network than Cora and CiteSeer, presents greater challenges due to its limited feature richness (with attribute vectors of 500 dimensions) and more complex topology. On the other hand, BlogCatalog, a social network whose relational properties differ from those of citation networks (e.g., often featuring reciprocal connections rather than unidirectional citations), while possessing a moderate node count and numerous attributes, exhibits particularly complex clustering patterns typical of dense social graphs due to its high edge density.

Additionally, Table 4.2 further reinforces the importance of attribute information:

models relying solely on features (e.g., K-means using only attributes) achieve moderate NMI and ARI scores. This demonstrates that while attribute data alone can provide a reasonable basis for community detection, its rich combination with topological information is essential for achieving superior performance.

Finally, among attention-based autoencoder methods, including DAEGC, GATE, CDBNE, and G2ACO, all utilize **A** and **X** as inputs. However, G2ACO's multi-head attention encoder demonstrates a superior capacity to capture more nuanced relationships than the single-head or simpler self-attention mechanisms employed by the others. This advanced attention strategy consistently delivers superior community detection performance across all four benchmarks, thereby demonstrating the effectiveness of GAT-based representation learning for uncovering nonlinear structures in attributed networks.

Table 4.3—Evaluation Results on the Cora.

Model	**Input**	**NMI**	**ARI**
K-means	$\mathbf{X}$	0.547	0.501
DeepWalk	$\mathbf{A}, \mathbf{X}$	0.384	0.291
TADW	$\mathbf{A}, \mathbf{X}$	0.366	0.240
GAE	$\mathbf{A}, \mathbf{X}$	0.397	0.293
VGAE	$\mathbf{A}, \mathbf{X}$	0.408	0.347
MGAE	$\mathbf{A}, \mathbf{X}$	0.511	0.448
ARGA	$\mathbf{A}, \mathbf{X}$	0.449	0.352
ARVGA	$\mathbf{A}, \mathbf{X}$	0.450	0.374
DAEGC	$\mathbf{A}, \mathbf{X}$	0.528	0.496
GATE	$\mathbf{A}, \mathbf{X}$	0.527	0.451
ARVGA-AX	$\mathbf{A}, \mathbf{X}$	0.526	0.495
SENet	$\mathbf{A}, \mathbf{X}$	0.550	0.490
VGAER	$\mathbf{A}, \mathbf{B}$	0.433	0.347
CDBNE	$\mathbf{A}, \mathbf{X}$	0.537	0.513
G2ACO	$\mathbf{A}, \mathbf{X}$	**0.564**	**0.516**

Table 4.4—Evaluation Results on the CiteSeer.

Model	**Input**	**NMI**	**ARI**
K-means	$\mathbf{X}$	0.312	0.285
DeepWalk	$\mathbf{A}, \mathbf{X}$	0.131	0.137
TADW	$\mathbf{A}, \mathbf{X}$	0.320	0.286
GAE	$\mathbf{A}, \mathbf{X}$	0.174	0.141
VGAE	$\mathbf{A}, \mathbf{X}$	0.163	0.101
MGAE	$\mathbf{A}, \mathbf{X}$	0.412	0.414
ARGA	$\mathbf{A}, \mathbf{X}$	0.350	0.341
ARVGA	$\mathbf{A}, \mathbf{X}$	0.261	0.245
DAEGC	$\mathbf{A}, \mathbf{X}$	0.397	0.410
GATE	$\mathbf{A}, \mathbf{X}$	0.401	0.381
ARVGA-AX	$\mathbf{A}, \mathbf{X}$	0.338	0.301
SENet	$\mathbf{A}, \mathbf{X}$	0.417	0.424
VGAER	$\mathbf{A}, \mathbf{B}$	0.216	-
CDBNE	$\mathbf{A}, \mathbf{X}$	0.438	0.455
G2ACO	$\mathbf{A}, \mathbf{X}$	**0.456**	**0.481**

Table 4.5—Evaluation Results on the PubMed.

Model	**Input**	**NMI**	**ARI**
K-means	$\mathbf{X}$	0.278	0.246
DeepWalk	$\mathbf{A}, \mathbf{X}$	0.238	0.255
TADW	$\mathbf{A}, \mathbf{X}$	0.224	0.177
GAE	$\mathbf{A}, \mathbf{X}$	0.249	0.246
VGAE	$\mathbf{A}, \mathbf{X}$	0.216	0.201
MGAE	$\mathbf{A}, \mathbf{X}$	0.282	0.248
ARGA	$\mathbf{A}, \mathbf{X}$	0.276	0.291
ARVGA	$\mathbf{A}, \mathbf{X}$	0.117	0.078
DAEGC	$\mathbf{A}, \mathbf{X}$	0.266	0.278
GATE	$\mathbf{A}, \mathbf{X}$	0.322	0.299
ARVGA-AX	$\mathbf{A}, \mathbf{X}$	0.239	0.226
SENet	$\mathbf{A}, \mathbf{X}$	0.306	0.297
VGAER	$\mathbf{A}, \mathbf{B}$	0.212	-
CDBNE	$\mathbf{A}, \mathbf{X}$	0.336	**0.337**
G2ACO	$\mathbf{A}, \mathbf{X}$	**0.345**	0.331

Table 4.6—Evaluation Results on the BlogCatalog.

Model	Input	NMI	ARI
K-means	$\mathbf{X}$	0.0662	0.0015
DeepWalk	$\mathbf{A}, \mathbf{X}$	0.193	0.113
TADW	$\mathbf{A}, \mathbf{X}$	0.067	0.010
GAE	$\mathbf{A}, \mathbf{X}$	0.098	0.03
VGAE	$\mathbf{A}, \mathbf{X}$	0.072	0.043
MGAE	$\mathbf{A}, \mathbf{X}$	0.168	0.086
ARGA	$\mathbf{A}, \mathbf{X}$	0.189	0.106
ARVGA	$\mathbf{A}, \mathbf{X}$	0.224	0.145
DAEGC	$\mathbf{A}, \mathbf{X}$	0.190	0.126
GATE	$\mathbf{A}, \mathbf{X}$	0.248	0.159
ARVGA-AX	$\mathbf{A}, \mathbf{X}$	0.261	0.170
SENet	$\mathbf{A}, \mathbf{X}$	0.241	0.161
VGAER	$\mathbf{A}, \mathbf{B}$	0.115	0.085
CDBNE	$\mathbf{A}, \mathbf{X}$	0.325	0.272
G2ACO	$\mathbf{A}, \mathbf{X}$	**0.352**	**0.273**

4.2.3.1 Runtime Efficiency

To comprehensively assess G2ACO's computational efficiency, we measured both its training time and the number of epochs required for convergence on the Cora, CiteSeer, PubMed, and BlogCatalog datasets, utilizing a consistent hardware setup. For comparison, we benchmarked G2ACO against CDBNE, employing its publicly available implementation. As illustrated in Table 4.7, G2ACO consistently demonstrates superior runtime performance and lower memory consumption. While CDBNE performed adequately on Cora, CiteSeer, and BlogCatalog, it encountered an out-of-memory error on the PubMed dataset, leading to its termination after only one epoch.

Table 4.7—Execution Time Comparison of G2ACO and CDBNE.

	G2ACO		CDBNE	
	Time	Epochs	Time	Epochs
Cora	**57 s**	100	149 s	100
CiteSeer	**59 s**	100	267 s	100
PubMed	**77 s**	25	Stopped after 300s (out of memory)	1
BlogCatalog	**58s**	25	143 s	25

4.2.3.2 Visualization of Learned Communities

To provide a visual, two-dimensional illustration of the communities discovered by G2ACO, we project the final node embeddings, learned from both the adjacency matrix (**A**) and attribute matrix (**X**), into two dimensions using t-SNE (Van der Maaten and Hinton, 2008). In these plots (Figure 4.9), each point represents a node, color-coded by its ground-truth community label. The resulting clusters visually convey the model's ability to separate communities and indicate the degree of overlap present.

Figure 4.9(a) shows G2ACO's embeddings on the Cora dataset, where communities appear as compact, well-separated clusters. By contrast, Figure 4.9(b) displays CDBNE's t-SNE projection on the same data, exhibiting more diffuse clusters and greater inter-community overlap, which underscores G2ACO's superior ability to capture the true community structure.

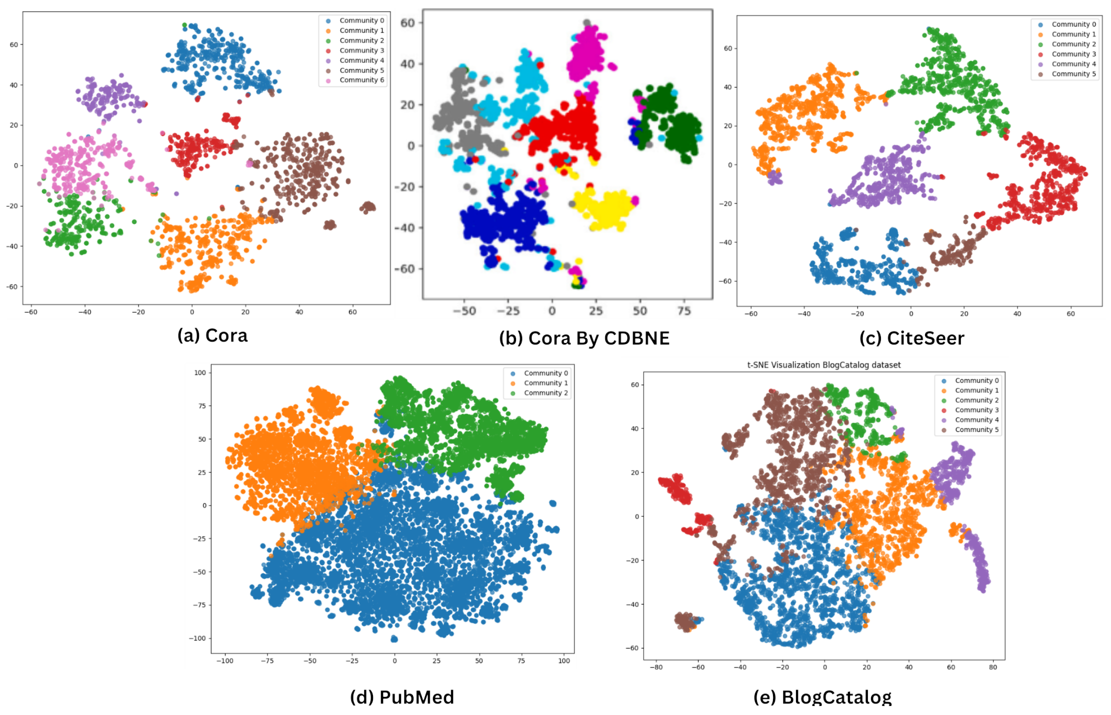


Figure 4.9—t-SNE visualization of node embeddings on Cora, CiteSeer, PubMed, and BlogCatalog. Colors indicate ground-truth communities; panels compare (a) G2ACO and (b) CDBNE on Cora.

4.2.4 Summary of the G2ACO Model

We proposed G2ACO as an unsupervised, multi-head attention autoencoder designed to learn rich, low-dimensional representations from attributed networks with the explicit goal of enhancing community detection. By combining a reconstruction loss with a novel K-means clustering objective, G2ACO simultaneously preserves structural and attribute information while steering embeddings toward well-separated communities. Empirical evaluation in three benchmark citation networks (Cora, CiteSeer, PubMed) and the BlogCatalog social graph consistently demonstrates that G2ACO outperforms state-of-the-art baselines. Future work will focus on scaling G2ACO to large-scale and dynamic networks, as well as extending its capabilities to handle overlapping community structures in evolving graph data.

4.3 Conclusion

This chapter presented two complementary proposed approaches to community detection that reflect the evolution of network analysis from classical heuristics to deep representation learning. The first model, AA-LPA (Bekkair et al., 2025b), demonstrates that deterministic enhancements to classical algorithms can yield interpretable and stable community assignments by incorporating structural centrality cues. In contrast, the second model, G2ACO Bekkair et al. (2025a), showcases the power of graph autoencoders to learn latent representations that unify topological and attribute information for more accurate, unsupervised clustering. While AA-LPA offers simplicity and reproducibility for un-attributed networks, G2ACO introduces a multi-head attention autoencoder coupled with a K-means clustering objective, jointly learning structural and attribute representations that outperform existing baselines on diverse citation and social networks. G2ACO captures the complexity of attributed graphs through deep neural architectures. Together, these methods illustrate the trade-offs and complementarities between interpretability and expressive power in modern community detection, paving the way for future hybrid models that integrate structural priors with learned representations in scalable and adaptive models.

Conclusion and perspectives

This thesis embarked on a comprehensive investigation into the intricate problem of community detection in graph-structured data, a fundamental challenge in network analysis. Part One laid the foundational groundwork, providing a thorough background in graph theory, machine learning, and deep learning paradigms, particularly as they apply to graph-based problems. This foundational understanding culminated in a review of the state of the art in community detection methodologies, encompassing classical methods, traditional machine learning approaches, and recent deep learning innovations. This exhaustive taxonomy highlighted existing capabilities and, critically, inherent limitations within the current landscape.

Building upon this extensive preparatory work, Part Two presented the core contributions of this research. It started with rigorous comparative studies that critically assessed the performance of established methods. These evaluations were conducted in two distinct contexts: first, an analysis of CNN-based approaches tailored for non-attributed networks, and second, an examination of Graph Autoencoder (GAE)-based models for attributed networks. These comparative analyses served to identify specific areas where existing methods face significant challenges, thereby establishing performance benchmarks and setting the stage for the novel contributions that followed.

The first model, AA-LPA, transforms the classic Label Propagation Algorithm into a fully deterministic procedure by ordering updates via Adamic–Adar centrality, weighting neighbor labels by shared-neighbor importance, and resolving ties without randomness. AA-LPA achieves stable and interpretable partitions that align closely with ground truth.

The principal contribution of this thesis, however, lies in the development of Graph Attention Autoencoder Clustering-Oriented (G2ACO), a novel deep learning model specifically designed for attributed networks. G2ACO is an unsupervised multi-head graph attention autoencoder with a dual-objective loss. By combining a reconstruction term with a novel K-means clustering loss applied to detached embeddings, G2ACO learns low-dimensional node representations that are both faithful to the original graph and intrinsically organized into cohesive communities. The proposed optimization strategy decouples centroid updates from backpropagation, effectively handling non-differentiability and ensuring stable convergence. Empirical results on citation and social networks demonstrate that G2ACO outperforms competitive baselines in terms of NMI and ARI, while also converging faster than existing models

Collectively, the contributions of this thesis contribute the field of community detection by offering both refined classical algorithms and innovative deep learning solutions. AA-LPA provides an interpretable tool for non-attributed graphs, while G2ACO emerges as a powerful and highly expressive deep representation learning framework for attributed networks, effectively unifying topological and attribute information. These models not only achieve superior performance against existing baselines, but also offer valuable insights into the trade-offs between interpretability and expressive power in graph-based learning.

Looking ahead, one promising direction is the development of scalable centrality approximations using sampling techniques or parallel computation to extend AA-LPA to billion-edge graphs. Adapting G2ACO for online and dynamic clustering by integrating incremental centroid updates and streaming friendly attention mechanisms could enable community detection in evolving networks. Extending both frameworks with soft-assignment or hierarchical clustering objectives would allow the discovery of overlapping and nested communities. Investigating hybrid models that combine AA-LPA's interpretability with G2ACO's expressive power, such as initializing deep embeddings with centrality-based seeds, offers a path toward more versatile algorithms. Finally, applying these methods to real-world dynamic systems social media, biological interaction networks, and infrastructure graphs, will test their robustness in diverse, noisy environments and drive further refinements.

By integrating deterministic structural insights with deep, clustering-aware embeddings, this work lays the foundation for future community detection that are scalable, adaptive, and reliable, capable of uncovering complex, multi-scale structures in modern networked data.

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA, 2016. USENIX Association. ISBN 9781931971331.

Emmanuel Abbe. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18, 2017. ISSN 1533-7928.

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630 (8016):493–500, 2024.

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25 (3):211–230, 2003.

Alessia Amelio and Clara Pizzuti. Is normalized mutual information a fair measure for comparing community detection methods? In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1584–1585, 2015. doi: 10.1145/2808797.2809344.

Arash A. Amini, Aiyou Chen, Peter J. Bickel, and Elizaveta Levina. Pseudo-likelihood methods for community detection in large sparse networks. *Annals of Statistics*, 41, 2013. ISSN 00905364. doi: 10.1214/13-AOS1138.

Imelda Atastina, Benhard Sitohang, GA Putri Saptawati, and Veronica S Moertini. A review of big graph mining research. In *IOP Conference Series: Materials Science and Engineering*, volume 180, page 012065. IOP Publishing, 2017.

AL Barabasi and M Pósfai. *Network Science*. Cambridge University Press, 2016. ISBN 978-1-107-07626-6.

Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

A Attea Bara'a, Amenah D Abbood, Ammar A Hasan, Clara Pizzuti, Mayyadah Al-Ani, Suat Özdemir, and Rawaa Dawoud Al-Dabbagh. A review of heuristics and metaheuristics for community detection in complex networks: Current usage, emerging development and future directions. *Swarm and Evolutionary Computation*, 63:100885, 2021.

Earl R Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic Discrete Methods*, 3(4):541–550, 1982.

Abdelfateh Bekkair, Slimane Oulad-Naoui, Slimane Bellaouar, Rababe Roumaissa Krimat, and Alla Haddaoui. Cnn-based community detection: A comparison study. In *2023 5th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–6. IEEE, 2023.

Abdelfateh Bekkair, Slimane Bellaouar, Slimane Oulad-Naoui, and Kaoutar Zita. A comparative study of graph autoencoder-based community detection in attributed networks. In *2024 International Conference on Telecommunications and Intelligent Systems (ICTIS)*, pages 1–6. IEEE, 2024.

Abdelfateh Bekkair, Slimane Bellaouar, and Slimane Oulad-Naoui. Unsupervised graph attention autoencoder clustering-oriented for community detection in attributed networks. *International Journal of Data Science and Analytics*, pages 1–14, 2025a.

Abdelfateh Bekkair, Slimane Bellaouar, and Slimane Oulad-Naoui. Community detection via enhanced label propagation with adamic-adar similarity and community size influence. In *2025 7th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–6, 2025b. doi: 10.1109/PAIS66004.2025.11126491.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Vincent D. Blondel, Jean Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008, 2008. ISSN 17425468. doi: 10.1088/1742-5468/2008/10/P10008.

Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2007.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Biao Cai, Yanpeng Wang, Lina Zeng, Yanmei Hu, and Hongjun Li. Edge classification based on convolutional neural networks for community detection in complex network. *Physica A: Statistical Mechanics and its Applications*, 556:124826, 10 2020. ISSN 0378-4371. doi: 10.1016/J.PHYSA.2020.124826.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, 30(9):1616–1637, 2018.

Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386, 2017.

Youngchul Cha and Junghoo Cho. Social-network analysis using topic models. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, page 565–574, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314725. doi: 10.1145/2348283.2348360.

Naiyue Chen, Yun Liu, Haiqiang Chen, and Junjun Cheng. Detecting communities in social networks using label propagation with information entropy. *Physica A: Statistical Mechanics and its Applications*, 471:788–798, 2017.

Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014a. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014b.

Mahfuzur Rahman Chowdhury, Intesur Ahmed, Farig Sadeque, and Muhammad Yanhaona. Topic modeling using community detection on a word association graph. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 908–917, 2023.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 70(6):066111, 2004.

J-J Daudin, Franck Picard, and Stéphane Robin. A mixture model for random graphs. *Statistics and computing*, 18(2):173–183, 2008.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th Inter-*

national Conference on Neural Information Processing Systems, NIPS'16, page 3844–3852, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Chong Deng, Xin-Jian Xu, and Shihui Ying. Strong consistency of spectral clustering for the sparse degree-corrected hypergraph stochastic block model. *IEEE Transactions on Information Theory*, 70(3):1962–1977, 2024. doi: 10.1109/TIT.2023.3302283.

Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 606–610. SIAM, 2005.

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.

Asma Douadi, Nadjet Kamel, and Lakhdar Sais. Label propagation algorithm for community discovery based on centrality and common neighbours. *The Journal of Supercomputing*, 80(8):11816–11842, 2024.

Jeffrey L. Elman. Finding structure in time. In *Cognitive Science*, volume 14, pages 179–211, 1990.

Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell rna-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):390, 2019.

K Erciyes. *Algebraic Graph Algorithms*. Springer, 2021.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.

Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, pages 128–140, 1741.

Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. A survey of community search over big graphs. *The VLDB Journal*, 29: 353–392, 2020.

Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Commun. ACM*, 59(7):96–104, June 2016. ISSN 0001-0782. doi: 10.1145/2818717.

Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2 2010. ISSN 0370-1573. doi: 10.1016/J.PHYSREP.2009.11.002.

Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40, 1977. ISSN 00380431. doi: 10.2307/3033543.

Mohammad Ghadirian and Nooshin Bigdeli. A new adaptive robust modularized semi-supervised community detection method based on non-negative matrix factorization. *Neural Processing Letters*, 56(2):134–152, April 2024. doi: 10.1007/s11063-024-11588-y.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: an automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, DL '98, page 89–98, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919653. doi: 10.1145/276675.276685.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for earning in raph domains. *Proceedings of the International Joint Conference on Neural Networks*, 2:729–734, 2005. doi: 10.1109/IJCNN.2005.1555942.

Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. volume 13-17-August-2016, pages 855–864. Association for Computing Machinery, 8 2016. ISBN 9781450342322. doi: 10.1145/2939672.2939754.

Roger Guimerà and Luís A. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005. doi: 10.1038/nature03288.

Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing*, pages 373–382, Cham, 2017. Springer International Publishing. ISBN 978-3-319-70096-0.

William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. volume 2017-December, 2017.

Pengyi Hao, Zhaojie Qian, Shuang Wang, and Cong Bai. Community aware graph embedding learning for item recommendation. *World Wide Web*, 26(6):4093–4108, 2023.

Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009. ISBN 9780387848570. doi: 10.1007/978-0-387-84858-7.

Chaobo He, Yulong Zheng, Junwei Cheng, Yong Tang, Guohua Chen, and Hai Liu. Semi-supervised overlapping community detection in attributed graph with graph convolutional autoencoder. *Information Sciences*, 608:1464–1479, 2022. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2022.07.036.

Yafeng He, Zhiqiang Chen, and Alan C. Evans. Structural insights into aberrant topological patterns of large-scale cortical networks in alzheimer's disease. *Journal of Neuroscience*, 28(18):4756–4766, 2008. doi: 10.1523/JNEUROSCI.4623-07.2008.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 289–296. Morgan Kaufmann, 1999.

John H Holand. Adaptation in natural and artificial systems. *Ann Arbor: The University of Michigan Press*, 32, 1975.

Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5:109–137, 1983. ISSN 03788733. doi: 10.1016/0378-8733(83)90021-7.

Huiting Hong, Xin Li, and Mingzhong Wang. Gane: A generative adversarial network embedding. *IEEE Transactions on Neural Networks and Learning Systems*, 31(7): 2325–2335, 2020. doi: 10.1109/TNNLS.2019.2921841.

Zhihao Huang, Xiaoxiong Zhong, Qiang Wang, Maoguo Gong, and Xiaoke Ma. Detecting community in attributed networks by dynamically exploring node attributes and topological structure. *Knowledge-Based Systems*, 196:105760, 2020. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2020.105760.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2: 193–218, 1985.

Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. Communitygan: Community detection with generative adversarial nets. In *The World Wide Web Conference*, WWW '19, page 784–794, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313564.

Rushi Jiao, Yichi Zhang, Le Ding, Bingsen Xue, Jicong Zhang, Rong Cai, and Cheng Jin. Learning with limited annotations: A survey on deep semi-supervised learning for medical image segmentation. *Computers in Biology and Medicine*, 169:107840, 2024. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2023.107840.

Di Jin, Jing He, Bianfang Chai, and Dongxiao He. Semi-supervised community detection on attributed networks using non-negative matrix tri-factorization with node popularity. *Frontiers of Computer Science*, 15:1–11, 2021.

Baoyu Jing, Chanyoung Park, and Hanghang Tong. Hdmi: High-order deep multiplex infomax. In *Proceedings of the Web Conference 2021*, WWW '21, page 2414–2424, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449971.

I.T. Jolliffe and J. Cadima. *Principal Component Analysis*. Springer, 2016.

Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 83(1): 016107, 2011.

George Karypis. Metis: Unstructured graph partitioning and sparse matrix ordering system. *Technical report*, 1997.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. ieee, 1995.

B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49, 1970. ISSN 15387305. doi: 10.1002/j.1538-7305.1970.tb01770.x.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2017.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Monika I. Konaklieva and Balbina J. Plotkin. β-lactams and ureas as cross inhibitors of prokaryotic systems. *Applied Microbiology*, 3(3):605–628, 2023. ISSN 2673-8007. doi: 10.3390/applmicrobiol3030043.

Konstantinos Konstantinidis, Symeon Papadopoulos, and Yiannis Kompatsiaris. Community structure and evolution analysis of osn interactions around real-world social phenomena. In *Proceedings of the 17th Panhellenic Conference on Informatics*, PCI '13, page 9–16, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319690. doi: 10.1145/2491845.2491849.

Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

K. Krishna and M. Narasimha Murty. Genetic k-means algorithm. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 29:433–439, 1999. ISSN 1083-4419. doi: 10.1109/3477.764879.

Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4, pages 950–957, 1992.

Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 611–617, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150476.

Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008. doi: 10.1103/PhysRevE.78.046110.

Peter Langfelder and Steve Horvath. WGCNA: an r package for weighted correlation network analysis. *BMC Bioinformatics*, 9:559, 2008. doi: 10.1186/1471-2105-9-559.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998. doi: 10.1109/5.726791.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2, 2007. doi: 10.1145/1217299.1217301.

Ian X. Y. Leung, Pan Hui, Pietro Liò, and Jon Crowcroft. Towards real-time community detection in large networks. *Phys. Rev. E*, 79:066107, Jun 2009. doi: 10.1103/PhysRevE.79.066107.

Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2019. doi: 10.1109/TSP.2018.2879624.

Jundong Li, Xia Hu, Jiliang Tang, and Huan Liu. Unsupervised streaming feature selection in social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1041–1050, 2015.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Jian Yang, and Philip S. Yu. Deep learning for community detection: progress, challenges and opportunities. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165.

Xiaoyang Liu, Mengyao Zhang, Yanfei Liu, Chao Liu, Chaorong Li, Wei Wang, Xiaoqin Zhang, and Asgarali Bouyer. Semi-supervised community detection method based on generative adversarial networks. *Journal of King Saud University - Computer and Information Sciences*, 36(3):102008, 2024. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2024.102008.

Zhigang Liu, Guangxiao Yuan, and Xin Luo. Symmetry and nonnegativity-constrained matrix factorization for community detection. *IEEE/CAA Journal of Automatica Sinica*, 9(9):1691–1693, 2022. doi: 10.1109/JAS.2022.105794.

David Lusseau et al. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4): 396–405, 2003.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

Trevor Martin. community2vec: Vector representations of online communities encode semantic relationships. In *Proceedings of the Second Workshop on NLP and Computational Social Science*, pages 27–31, 2017.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

Julian Mcauley and Jure Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):1–28, 2014.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

Aaron F McDaid, Thomas Brendan Murphy, Nial Friel, and Neil J Hurley. Improved bayesian inference for the stochastic block model with application to large networks. *Computational Statistics & Data Analysis*, 60:12–31, 2013.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013b. Curran Associates Inc.

Edward F Moore. The shortest path through a maze. In *Proc. of the International Symposium on the Theory of Switching*, pages 285–292. Harvard University Press, 1959.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

M. E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103, 2006. ISSN 00278424. doi: 10.1073/pnas.0601602103.

Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 69(6):066133, 2004.

Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2609–2615. AAAI Press, 2018. ISBN 9780999241127.

Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. Unsupervised attributed multiplex network embedding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5371–5378. AAAI Press, 2020. doi: 10.1609/AAAI.V34I04.5985.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20*, pages 284–293. Springer, 2005.

Alex Pothen. Graph partitioning algorithms with applications to scientific computing. In *Parallel Numerical Algorithms*, pages 323–368. Springer, 1997.

Jonathan D. Power, Alexander L. Cohen, Steven M. Nelson, Gagan S. Wig, Kelly A. Barnes, Jessica A. Church, Alecia C. Vogel, Timothy O. Laumann, Fran M. Miezin, Bradley L. Schlaggar, and Steven E. Petersen. Functional network organization of the human brain. *Neuron*, 72(4):665–678, 2011. doi: 10.1016/j.neuron.2011.09.006.

Chenyang Qiu, Zhaoci Huang, Wenzhe Xu, and Huijia Li. Vgaer: Graph neural network reconstruction based community detection. In *AAAI: DLG-AAAI'22*, 2022.

Sandeep Kumar Rachamadugu and T.P. Pushphavathi. Effective community detection with topic modeling in article recommender systems using ls-slm and pcc-lda. *J. Intell. Fuzzy Syst.*, 48(6):793–809, March 2025. ISSN 1064-1246. doi: 10.3233/JIFS-233851.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 76, 2007. ISSN 15393755. doi: 10.1103/PhysRevE.76.036106.

Carl Rasmussen. The infinite gaussian mixture model. *Advances in neural information processing systems*, 12, 1999.

Jörg Reichardt and Stefan Bornholdt. Clustering of sparse data via network communities—a prototype study of a large onlinemarket. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(06):P06016, 2007.

Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. Struc2vec: Learning node representations from structural identity. volume Part F129685, 2017. doi: 10.1145/3097983.3098061.

Stuart A Rice. The identification of blocs in small political bodies. *American Political Science Review*, 21(3):619–627, 1927.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105, 2008. ISSN 10916490. doi: 10.1073/pnas.0706851105.

Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Hadiseh Safdari and Caterina De Bacco. Anomaly detection and community detection in networks. *Journal of Big Data*, 9(1):122, 2022. ISSN 2196-1115. doi: 10.1186/s40537-022-00669-1.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860, 2010.

Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

Amin Salehi and Hasan Davulcu. Graph attention auto-encoders. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 989–996, 2020. doi: 10.1109/ICTAI50040.2020.00154.

Aniello De Santo, Antonio Galli, Vincenzo Moscato, and Giancarlo Sperlì. A deep learning approach for semi-supervised community detection in online social networks. *Knowledge-Based Systems*, 229, 2021. ISSN 09507051. doi: 10.1016/j.knosys.2021.107345.

Bilal Saoud and Abdelouahab Moussaoui. A new hierarchical method to find community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 495: 418–426, 2018.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20: 61–80, 1 2009. ISSN 10459227. doi: 10.1109/TNN.2008.2005605.

Robert Sedgewick and Kevin Wayne. *Algorithms, 4th Edition.* Addison-Wesley, 2011. ISBN 978-0-321-57351-3.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Breno Serrano and Thibaut Vidal. Community detection in the stochastic block model by mixed integer programming. *Pattern Recognition*, 152:110487, 2024.

Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *Deep Learning on Graphs Workshop, KDD*, 2019.

Xiaohua Shi, Hongtao Lu, Yangchen He, and Shan He. Community detection in social network with pairwisely constrained symmetric non-negative matrix factorization. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 541–546, 2015. doi: 10.1145/2808797.2809383.

Hyun Ah Song, Bo-Kyeong Kim, Thanh Luong Xuan, and Soo-Young Lee. Hierarchical feature extraction by multi-layer non-negative matrix factorization network for classification task. *Neurocomputing*, 165:63–74, 2015. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2014.08.095.

Giancarlo Sperlí. A deep learning based community detection approach. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, page 1107–1110, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359337. doi: 10.1145/3297280.3297574.

Victor Spirin and Leonid A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003. doi: 10.1073/pnas.2034524100.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, Quan Z. Sheng, and Philip S. Yu. A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. ISSN 21622388. doi: 10.1109/TNNLS.2021.3137396.

Feng Tang, Jian Li, Xiao Liu, Cheng Chang, and Lei Teng. Gatfelpa integrates graph attention networks and enhanced label propagation for robust community detection. *Scientific Reports*, 15(1):3952, 2025. doi: 10.1038/s41598-024-84962-4.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

Robert Tarjan. Depth-first search and linear graph algorithms. In *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, pages 114–121, 1971. doi: 10.1109/SWAT.1971.10.

Yee W. Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

N Kipf Thomas and Max Welling. Variational graph auto-encoders.(2016). In *Neural Information Processing Systems Workshop on Bayesian Deep Learning*, 2016.

Vincent A Traag and Lovro Šubelj. Large network community detection by fast label propagation. *Scientific Reports*, 13(1):2701, 2023.

Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

Volkan Tunali. Large-scale network community detection using similarity-guided merge and refinement. *IEEE Access*, 9:78538–78552, 2021.

A. M. TURING. I.—computing machinery and intelligence. *Mind*, LIX(236):433–460, 10 1950. ISSN 0026-4423. doi: 10.1093/mind/LIX.236.433.

Saif Ur Rehman, Kexing Liu, Tariq Ali, Asif Nawaz, and Simon James Fong. A graph mining approach for ranking and discovering the interesting frequent subgraph patterns. *International Journal of Computational Intelligence Systems*, 14(1):152, 2021.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. volume 2017-December, 2017.

Petar Veličković, Arantxa Casanova, Pietro Liò, Guillem Cucurull, Adriana Romero, and Yoshua Bengio. Graph attention networks. 2018. doi: 10.1007/978-3-031-01587-8_7.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 889–898, New York, NY, USA, 2017a. Association for Computing Machinery. ISBN 9781450349185. doi: 10.1145/3132847.3132967.

Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: a deep attentional embedding approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 3670–3676. AAAI Press, 2019. ISBN 9780999241141.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018a.

Meng Wang, Xiaodong Cai, Yan Zeng, and Xiaoxi Liang. A community detection algorithm based on jaccard similarity label propagation. In *Intelligent Data Engineering and Automated Learning–IDEAL 2017: 18th International Conference, Guilin, China, October 30–November 1, 2017, Proceedings 18*, pages 45–52. Springer, 2017b.

Wenjun Wang, Xiao Liu, Pengfei Jiao, Xue Chen, and Di Jin. A unified weakly supervised framework for community detection and semantic matching. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22*, pages 218–230. Springer, 2018b.

Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang. Semantic community identification in large attribute networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Feb. 2016. doi: 10.1609/aaai.v30i1.9977.

Xiao Wang, Fang Dai, Wenyan Guo, and Junfeng Wang. Community detection in attributed networks using stochastic block models. *Physica A: Statistical Mechanics and its*

Applications, 666:130432, 2025. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2025.130432.

Todd Waskiewicz. Friend of a friend influence in terrorist social networks. In *Proceedings on the international conference on artificial intelligence (ICAI)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer ..., 2012.

Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore, 2022.

Xunlian Wu, Wanying Lu, Yining Quan, Qiguang Miao, and Peng Gang Sun. Deep dual graph attention auto-encoder for community detection. *Expert Systems with Applications*, 238:122182, 2024. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2023.122182.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32, 2021. ISSN 21622388. doi: 10.1109/TNNLS.2020.2978386.

Xin Xin, Chaokun Wang, Xiang Ying, and Boyang Wang. Deep community detection in topologically incomplete networks. *Physica A: Statistical Mechanics and its Applications*, 469:342–352, 2017. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2016.11.029.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A.J. Schweiger. Scan: A structural clustering algorithm for networks. 2007. doi: 10.1145/1281192.1281280.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2111–2117. AAAI Press, 2015. ISBN 9781577357384.

Jaewon Yang and Jure Leskovec. Community-affiliation graph model for overlapping network community detection. In *2012 IEEE 12th international conference on data mining*, pages 1170–1175. IEEE, 2012.

Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th international conference on data mining*, pages 1151–1156. IEEE, 2013.

Fanghua Ye, Chuan Chen, and Zibin Zheng. Deep autoencoder-like nonnegative matrix factorization for community detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1393–1402, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3271697.

Zhijun Yin, Liangliang Cao, Quanquan Gu, and Jiawei Han. Latent community topic analysis: Integration of community discovery with topic modeling. *ACM Trans. Intell. Syst. Technol.*, 3(4), September 2012. ISSN 2157-6904. doi: 10.1145/2337542.2337548.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

Tao You, Hui Min Cheng, Yi Zi Ning, Ben Chang Shia, and Zhong Yuan Zhang. Community detection in complex networks using density-based clustering algorithm and manifold learning. *Physica A: Statistical Mechanics and its Applications*, 464, 2016. ISSN 03784371. doi: 10.1016/j.physa.2016.07.025.

Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

Fataneh Dabaghi Zarandi and Marjan Kuchaki Rafsanjani. Community detection in complex networks using structural similarity. *Physica A: Statistical Mechanics and its Applications*, 503:882–891, 8 2018. ISSN 03784371. doi: 10.1016/j.physa.2018.02.212.

Haizheng Zhang, Baojun Qiu, C. Lee Giles, Henry C. Foley, and John Yen. An lda-based community structure discovery approach for large-scale social networks. In *2007 IEEE Intelligence and Security Informatics*, pages 200–207, 2007. doi: 10.1109/ISI.2007.379553.

Hongyuan Zhang, Pei Li, Rui Zhang, and Xuelong Li. Embedding graph auto-encoder for graph clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11): 9352–9362, 2023. doi: 10.1109/TNNLS.2022.3158654.

Tianqi Zhang, Yun Xiong, Jiawei Zhang, Yao Zhang, Yizhu Jiao, and Yangyong Zhu. Commdgi: Community detection oriented deep graph infomax. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 1843–1852, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412042.

Wei Zhang, Shanshan Yu, Ling Wang, Wei Guo, and Man-Fai Leung. Constrained symmetric non-negative matrix factorization with deep autoencoders for community detection. *Mathematics*, 12(10), 2024. ISSN 2227-7390. doi: 10.3390/math12101554.

Xian-Kun Zhang, Xue Tian, Ya-Nan Li, and Chen Song. Label propagation algorithm based on edge clustering coefficient for community detection in complex networks. *International Journal of Modern Physics B*, 28(30):1450216, 2014.

Xiaotong Zhang, Han Liu, Xiao-Ming Wu, Xianchao Zhang, and Xinyue Liu. Spectral embedding network for attributed graph clustering. *Neural Networks*, 142:388–396, 2021. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2021.05.026.

Yan Zhang, A.J. Friend, Amanda L. Traud, Mason A. Porter, James H. Fowler, and Peter J. Mucha. Community structure in congressional cosponsorship networks. *Physica A: Statistical Mechanics and its Applications*, 387(7):1705–1712, None 2008. doi: 10.1016/j.physa.2007.11.004.

Yao Zhang, Yun Xiong, Yun Ye, Tengfei Liu, Weiqiang Wang, Yangyong Zhu, and Philip S. Yu. Seal: Learning heuristics for community detection with generative adversarial networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1103–1113, New York, NY, USA, 2020b. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403154.

Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. Anrl: attributed network representation learning via deep neural networks. In *Ijcai*, volume 18, pages 3155–3161, 2018.

Xinchuang Zhou, Lingtao Su, Xiangju Li, Zhongying Zhao, and Chao Li. Community detection based on unsupervised attributed network embedding. *Expert Systems with Applications*, 213:118937, 3 2023. ISSN 0957-4174. doi: 10.1016/J.ESWA.2022.118937.

Jia Zhu, Zetao Zheng, Min Yang, Gabriel Pui Cheong Fung, and Changqin Huang. Protein complexes detection based on semi-supervised network embedding model. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(2):797–803, 2021. doi: 10.1109/TCBB.2019.2944809.

Xiaojin Zhu and Andrew Goldberg. *Introduction to semi-supervised learning*. Morgan & Claypool Publishers, 2009.

Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.