PAKDD 2026 HONG KONG

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SCHOOL OF DATA SCIENCE
數據科學學院

Foundations and Recent Advances of Graph Learning

A Perspective from Distances and Representations

Jicong Fan Xudong Wang Qi Feng Chris Ding

School of Data Science, The Chinese University of Hong Kong, Shenzhen

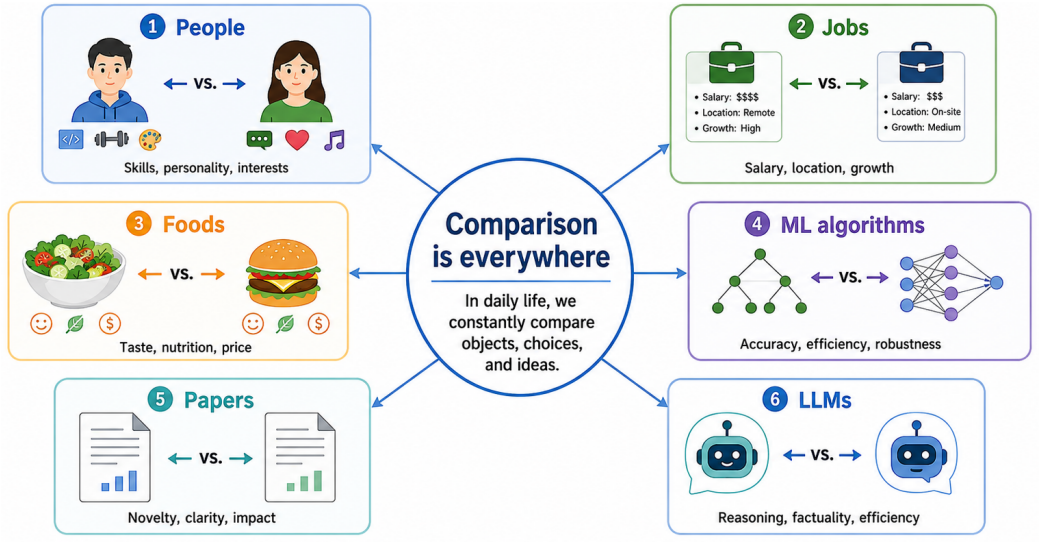
Tutorial 3 @ Tai Po I Room 3
Time: 09:00-12:30, June 9 (Tue)
PAKDD 2026, Hong Kong

Table of Contents

Outline and Schedule

Session	Topic	Time	Focus
Introduction	Background and motivation	9:00-9:15	Why distances and representations form one story
Part I	Graph distances	9:15-9:40	Edit, matching, transport, factorization
Part II	Graph kernels	9:40-10:10	Feature maps, the WL bridge, deep kernel
Part III-A	Graph embeddings	10:10-10:30	Spectral cuts, node2vec, etc.
Break	Tea Break	10:30-11:00	—
Part III-B	GNNs and graph transformers	11:00-11:30	Expressiveness and applications
Part IV	Self-supervised graph learning	11:30-12:00	Invariance, prediction, information max., graph entropy max.
Part V	Explainability and Generalizability, Discussion	12:00-12:30	Explainability, graph foundation models, open problems

Comparison is fundamental to how we understand the world



Two foundations of machine learning and data mining

Core message: representation defines the space; distance defines comparison.

Distance / Similarity

How close are two objects?

$d(x_i, x_j)$, $k(x_i, x_j)$

k-NN, k-means, kernel methods, DBSCAN
MDS, Isomap, LLE, t-SNE, UMAP, etc.

Representation

How should we describe data?

$z = f_\theta(x)$

PCA, ICA, LSH, autoencoder, neural networks
contrastive learning, etc

Why graph learning now?

Graphs are the native language of relational data

- Molecules, proteins, knowledge graphs, recommender systems, circuits, traffic, social networks, sensor networks, etc.
- The same object can be viewed at multiple scales: node, subgraph, whole graph, graph of graphs.
- Structure is not side information: it is part of the signal.

Why this is hard

- Graphs are irregular and variable-size.
- We must preserve both *combinatorics* and *attributes*.

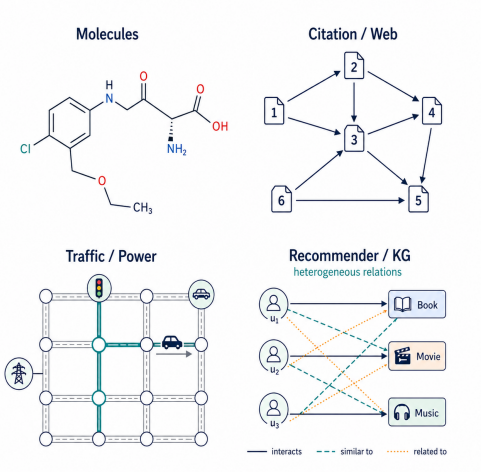


Figure. Representative graph-structured domains

Two primitive questions behind almost all graph methods

Question A: comparison

Beyond the isomorphism test, how similar are two graphs, nodes, or substructures?

$$G \longleftrightarrow G', \quad v \longleftrightarrow v', \quad S \longleftrightarrow S'.$$

Core tools:

- graph edit distance,
- matching and alignment,
- optimal transport,
- kernels and statistical distances.

Question B: representation

How should we encode a graph into a vector or a latent object?

$$f_\theta : \mathcal{G} \to \mathbb{R}^d.$$

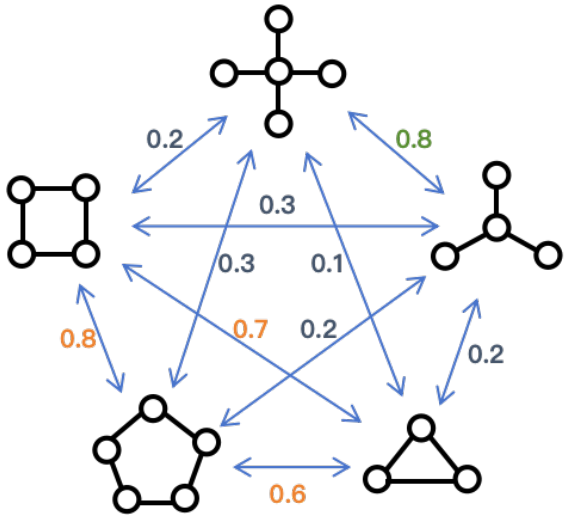
Core tools:

- graph kernels,
- message passing GNNs,
- graph transformers,
- self-supervised pretraining.

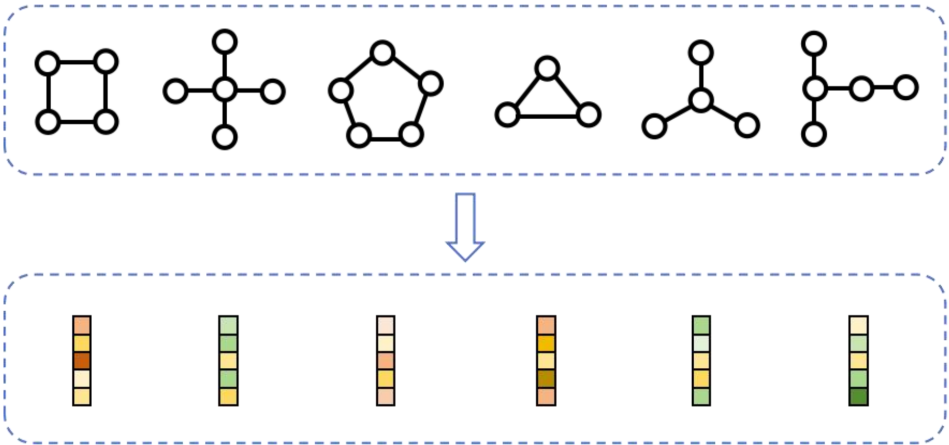
Main thesis of this tutorial

Distances and representations are not competing viewpoints. Modern graph learning moves back and forth between them.

A toy example of graph comparison



A toy example of graph representation



A unifying lens: explicit comparison versus learned encoding

Explicit comparison

objects $\rightarrow d(\cdot,\cdot) \rightarrow$ retrieval/clustering/...

- Good when interpretability matters.
- Good when data are scarce but domain priors are strong.
- Often expensive, but conceptually transparent.

Learned encoding

object $\rightarrow f_\theta(\cdot) \rightarrow$task loss
or similarity in latent

- Good when scale and transfer matter.
- Learns task-adapted invariances.
- May sacrifice interpretability and exactness.

Takeaway: Modern graph learning often defines a comparison principle first, then amortizes it into a learned representation.

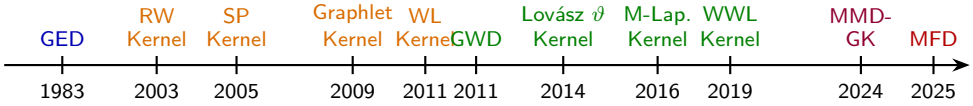
Timeline / Evolution of Graph Comparison and Representation

Graph comparison asks: how should two graphs be aligned, matched, or metrized?

edit / matching → kernelized similarity → OT / learned graph metrics

Timeline / Evolution of Graph Comparison and Representation

Graph comparison asks: how should two graphs be aligned, matched, or metrized?



edit / matching → kernelized similarity → OT / learned graph metrics

Graph representation asks: how should a graph be embedded so that comparison and downstream prediction become easy?

spectral relaxation → walk-based embedding → message passing → global attention / pretraining

- The modern field increasingly *couples* the two: comparison induces representation, and representation defines comparison.

Tutorial learning objectives

By the end of the tutorial, participants should be able to:

1. Understand the definition, strength, and weakness of each graph distance/kernel.
2. Understand the key ideas or principles of major graph representation learning methods.
3. Compare self-supervised graph objectives: contrastive, predictive, generative, and information-theoretic.
4. Form a critical view of emerging directions like explainability and graph foundation models.

Meta-goal

Rather than listing models, we emphasize principles: invariance, expressiveness, scalability, explainability, and transferability.

What makes a good graph distance?

Desiderata

A useful graph distance should ideally be:

- **discriminative:** different structures receive different scores,
- **stable:** small perturbations induce controlled changes,
- **invariant:** unaffected by node relabeling,
- **interpretable:** decomposable into meaningful discrepancies,
- **computable:** exact or well-approximated at useful scales.

A taxonomy of graph comparison

Family	Primitive	Strength	Main limitation
Edit / matching	node-edge operations, correspondences	interpretable, label-aware	NP-hard or combinatorial
Kernel	shared substructures	PSD similarity, classical learning theory	feature design bias
Transport	alignment of distributions / metrics	flexible for size mismatch and attributes	cubic or near-cubic cost
Embedding distance	learned latent vectors	scalable, task-aware	depends on encoder quality
Factorization	matrix / spectral summaries	global structure, clustering-friendly	may lose local semantics

Takeaway: There is no universal best graph distance.

Why explicit graph comparison is still useful in 2026

When distances are preferable

- small or medium-size datasets,
- retrieval and nearest-neighbor search,
- clustering with limited labels,
- domains needing interpretable mismatch decomposition,
- evaluation of generative or self-supervised models.

Why explicit graph comparison is still useful in 2026

When distances are preferable

- small or medium-size datasets,
- retrieval and nearest-neighbor search,
- clustering with limited labels,
- domains needing interpretable mismatch decomposition,
- evaluation of generative or self-supervised models.

Canonical applications

- molecular similarity and scaffold retrieval,
- graph clustering and graph database search,
- graph matching in vision and pattern recognition,
- cross-network alignment and anomaly detection.

Takeaway: Distances never disappeared. They became the reference geometry against which learned representations are judged.

Graph edit distance (GED): the classical starting point

Given edit operations $\mathcal{O}$ (node/edge insertion, deletion, substitution) with costs $c(o)$, the **graph edit distance** [Sanfeliu & Fu, 1983] is

$$\mathrm{GED}(G, G') = \min_{\pi \in \Pi(G, G')} \sum_{o \in \pi} c(o),$$

where $\Pi(G, G')$ denotes all valid edit paths transforming G to G'.

Strengths

- Natural, human-interpretable decomposition into atomic operations.
- Flexible: can incorporate labels, attributes, and semantic substitution costs.
- Well suited to retrieval, matching, and structured pattern recognition.

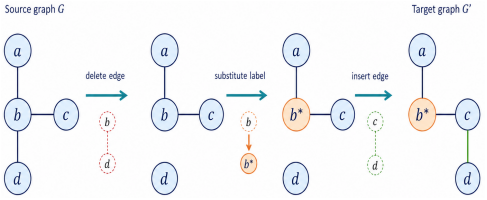
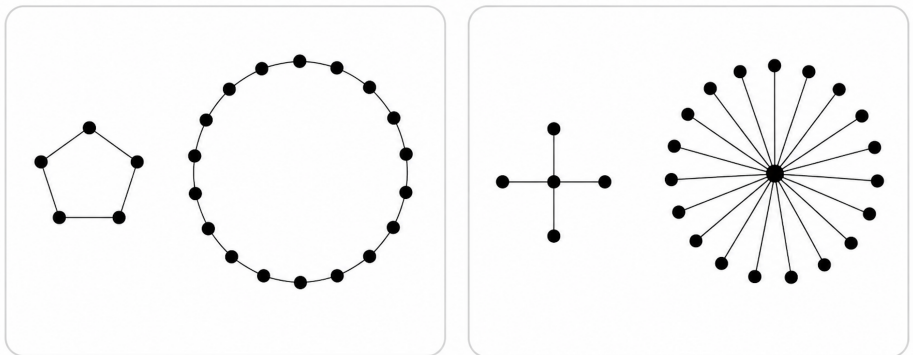


Figure. A valid edit path from G to G'

Limitations of GED

- Exact GED computation is NP-hard.
  - Approximation algorithms [Riesen & Bunke, 2009; Neuhaus & Bunke, 2007; Abu-Aisheh et al., 2015; Chang et al., 2022].
- GED is too sensitive to the size of the graph.



From edits to measures: optimal transport (OT)

Instead of transforming one graph into another via edit paths, transport compares two measures by moving mass [G. Monge, 1781]:

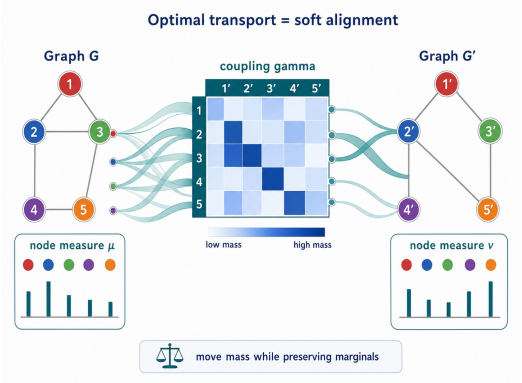
$$\mathcal{W}_c(\mu,\nu) = \min_{\gamma\in\Pi(\mu,\nu)} \sum_{i,j} c_{ij}\,\gamma_{ij} = \min_{\gamma\in\Pi(\mu,\nu)} \langle C,\gamma\rangle.$$

E.g.: $c_{ij} = \|x_i - x'_j\|^2$

Why OT is attractive

- handles varying graph sizes,
- supports soft alignment,
- naturally fuses features and geometry,
- differentiable versions exist.

Complexity: $O(n^3 \log n)$



Sinkhorn regularization: making OT practical

Add entropic regularization to the transport problem [Cuturi 2013]:

$$\mathcal{W}_c^{\varepsilon}(\mu,\nu) = \min_{\gamma\in\Pi(\mu,\nu)} \langle C,\gamma\rangle + \varepsilon\,\mathrm{KL}(\gamma\|\mu\otimes\nu).$$

Benefits

- faster iterative scaling: $O(n^2 T)$
- smooth and differentiable objective,
- useful inside neural training loops.

Trade-off

Large ε improves speed and smoothness but can oversoften alignments.

Takeaway: Sinkhorn turns OT from a beautiful idea into a usable primitive for machine learning.

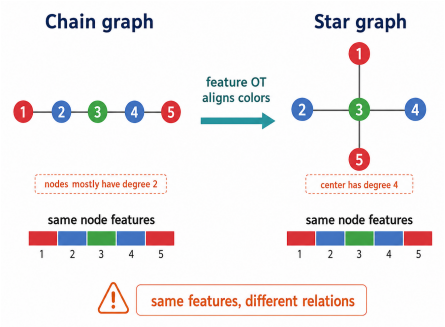
Why vanilla OT is not enough for graph comparison

Vanilla OT compares node features under a ground cost c_{ij}, but graphs are not just bags of nodes.

Missing ingredient

We also need to compare *relations between nodes*, not only nodes themselves.

- If node features are weak, feature-space OT can ignore topology.
- Two graphs with different internal geometry may still have similar node marginals.



Gromov-Wasserstein (GW): transport over relational structure

GW [Memoli, 2011] compares two metric measure spaces by aligning *intra-graph distances*:

$$\mathrm{GW}(G, G') = \min_{\gamma \in \Pi(\mu,\nu)} \sum_{i,i',j,j'} \left| d_G(i,i') - d_{G'}(j,j') \right|^2 \gamma_{ij} \gamma_{i'j'}.$$

d_G and $d_{G'}$ could be the adjacency matrices or the shortest-path distance matrices.

What changed?

The objective now matches *pairwise relations* rather than raw node identities.

- naturally handles graphs of different sizes,
- useful when there is no obvious node correspondence,
- conceptually close to structure matching up to soft isometry.

Fused Gromov-Wasserstein (FGW): combining attributes and structure

FGW [Vayer et al., 2019] interpolates between feature matching and relational matching:

$$\mathrm{FGW}_\alpha = \min_{\gamma \in \Pi(\mu,\nu)} (1-\alpha)\langle C, \gamma \rangle + \alpha \sum_{i,i',j,j'} L_{ii'jj'}\, \gamma_{ij}\gamma_{i'j'}.$$

Use cases

- attributed graphs,
- molecules with atom types,
- multimodal graphs with structure and side features.

Hyperparameter meaning

α determines whether topology or attribute similarity dominates.

Takeaway: FGW is one of the clearest examples of the distance and representation views meeting in one objective.

OT, GW, and GED: when should we prefer which?

Method	Best when	Main benefit	Main drawback
GED	semantic edit operations are meaningful	high interpretability	combinatorial cost
OT	node features are reliable and soft alignment is enough	differentiable, flexible size handling	topology may be underused
GW	internal geometry matters more than labels	topology-aware soft matching	expensive quartic-like objective before optimization tricks
FGW	both topology and features matter	balanced attributed comparison	tuning and computational burden

Takeaway: Choosing a graph distance is really choosing a structural inductive bias.

Beyond classical OT: factorization and learned graph metrics

Factorization view

Compare graphs through low-rank or structured matrix decompositions of adjacency, Laplacian, or feature-coupled operators.

Example direction

Graph minimum factorization distance compares graphs through optimal low-rank factor structures, offering a new route beyond edit and transport formulations.

MMFD [Fan, ICML 2025]

$$\mathrm{MMFD}(G_1, G_2) = \min_{\mathbf{R}_{12} \in \mathcal{R}} \left\| \frac{1}{n_1} \sum_{j=1}^{n_1} \phi\left(\mathbf{z}_j^{(1)}\right) - \frac{1}{n_2} \sum_{j=1}^{n_2} \mathbf{R}_{12} \phi\left(\mathbf{z}_j^{(2)}\right) \right\| = \left| \frac{1}{n_1} \sqrt{\sum_{uv} [\boldsymbol{\mathcal{A}}_1^{\phi}]_{uv}} - \frac{1}{n_2} \sqrt{\sum_{uv} [\boldsymbol{\mathcal{A}}_2^{\phi}]_{uv}} \right|$$

$\boldsymbol{\mathcal{A}}_i^{\phi} = \Phi_i^{\top} \Phi_i$ is a PSD proxy of $\mathbf{A}_i$ and Φ_i is from the SVD of $\mathbf{A}_i$.

Beyond classical OT: factorization and learned graph metrics

MFD [Fan, ICML 2025]

$$
\begin{aligned}
&\mathrm{MFD}(G_1, G_2) \\
&= \min_{\mathbf{R}_{12} \in \mathcal{R}} \left\| \frac{1}{n_1} \sum_{j=1}^{n_1} \zeta(\phi_j^{(1)}) - \frac{1}{n_2} \sum_{j=1}^{n_2} \zeta(\mathbf{R}_{12} \phi_j^{(2)}) \right\| \\
&= \min_{\mathbf{R}_{12} \in \mathcal{R}} \left(\frac{1}{n_1^2} \sum_{u,v} k(\phi_u^{(1)}, \phi_v^{(1)}) + \frac{1}{n_2^2} \sum_{u,v} k(\mathbf{R}_{12} \phi_u^{(2)}, \mathbf{R}_{12} \phi_v^{(2)}) - \frac{2}{n_1 n_2} \sum_{u,v} k(\phi_u^{(1)}, \mathbf{R}_{12} \phi_v^{(2)}) \right)^{1/2}
\end{aligned}
$$

where k denotes a nonlinear kernel with feature map ζ.

Example: MMFD for graph comparison

$G_1, G_2, \ldots, G_7$ from left to right:

					 G_5	 G_6	 G_7
	–	**0.0914**	0.1589	0.2097	0.2528	0.2505	0.2505
	0.0914	–	**0.0675**	0.1182	0.1614	0.1590	0.1591
	0.1589	0.0675	–	**0.0507**	0.0939	0.0915	0.0916
	0.2097	0.1182	0.0507	–	0.0432	**0.0408**	0.0409
G_5 	0.2528	0.1614	0.0939	0.0432	–	0.0024	**0.0023**
G_6 	0.2505	0.1590	0.0915	0.0408	0.0024	–	**0.0001**
G_7 	0.2505	0.1591	0.0916	0.0409	0.0023	**0.0001**	–

For instance, the difference between G_6 and G_7 is less than the difference between G_5 and G_6.

Example: MMFD and MFD for graph clustering

Method	AIDS ($N = 2000$)			PROTEINS ($N = 1113$)			ENZYMES ($N = 600$)			REDDIT-MULTI-5K ($N = 4999$)		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
SP kernel	79.49±0.84	0.39±0.62	-0.71±1.13	64.42±0.00	6.03±0.00	5.87±0.00	22.00±0.00	2.57±0.00	1.69±0.00	20.02±0.00	0.05±0.00	0.00±0.00
GK kernel	79.95±0.00	0.04±0.00	-0.07±0.00	59.61±0.22	0.24±0.18	0.10±0.19	17.07±0.13	0.80±0.25	0.00±0.00	–	–	–
RW kernel	79.90±0.00	0.09±0.00	-0.15±0.00	–	–	–	17.00±0.00	0.66±0.00	0.25±0.00	–	–	–
WL kernel	78.50±0.00	1.17±0.00	-2.09±0.00	60.38±0.00	1.55±0.00	0.81±0.00	21.00±0.00	3.09±0.00	1.48±0.00	20.00±0.00	0.00±0.00	0.00±0.00
LT kernel	79.95±0.00	0.04±0.00	-0.07±0.00	–	–	–	17.00±0.09	0.42±0.11	0.00±0.00	–	–	–
WL-OA kernel	80.40±0.00	2.46±0.00	2.38±0.00	60.38±0.00	1.55±0.00	0.81±0.00	20.00±0.00	1.35±0.00	0.32±0.00	–	–	–
InfoGraph+KM	92.21±0.81	54.49±3.53	63.78±3.84	59.22±0.21	3.22±1.94	0.00±0.00	22.06±0.98	2.40±0.45	1.25±0.52	20.16±0.02	0.30±0.05	0.00±0.00
InfoGraph+SC	95.65±1.55	72.21±9.20	80.17±7.19	64.02±2.31	5.17±1.87	7.06±2.65	23.75±0.50	4.64±0.65	2.23±0.41	20.00±0.00	0.00±0.00	0.00±0.00
GraphCL+KM	90.40±1.06	46.56±4.31	55.29±5.28	59.47±0.01	0.37±0.31	0.00±0.00	21.50±0.22	1.55±0.12	0.90±0.09	20.32±0.00	0.56±0.00	0.00±0.00
GraphCL+SC	96.08±1.96	72.97±10.86	81.65±8.51	59.96±0.10	2.81±0.07	3.88±0.08	25.28±0.28	4.75±0.36	2.03±0.26	20.08±0.00	0.16±0.00	0.00±0.00
JOAO+KM	88.25±0.00	38.02±0.00	44.62±0.00	59.48±0.00	0.64±0.05	-0.06±0.00	21.66±0.37	1.60±0.01	0.94±0.02	20.34±0.00	0.60±0.00	0.00±0.00
JOAO+SC	80.13±0.02	0.84±0.15	0.80±0.14	59.75±0.00	0.47±0.00	0.17±0.00	24.65±0.44	4.85±0.37	2.07±0.18	20.39±0.49	0.08±0.00	0.01±0.01
GWF+KM	96.43±1.71	74.48±9.15	84.71±7.02	66.87±2.36	9.07±1.21	11.43±3.19	28.55±0.20	6.02±0.55	3.16±0.20	–	–	–
GWF+SC	96.44±2.92	76.01±15.23	83.54±13.61	68.79±2.05	10.17±1.74	13.88±2.72	25.66±1.57	5.24±1.28	1.78±0.61	–	–	–
GLCC	79.02±0.62	4.18±2.01	5.05±2.13	60.65±2.69	2.08±1.43	4.16±2.28	19.89±1.09	2.42±0.18	0.19±0.12	23.50±0.48	6.57±3.56	4.00±0.80
DCGLC	96.77±0.33	73.51±2.30	85.74±1.45	68.89±2.04	10.90±1.35	14.32±2.88	28.43±1.28	6.57±0.20	3.78±0.47	33.24±2.34	8.81±2.28	7.16±1.67
MMD	50.10±0.00	0.00±0.00	0.03±0.00	52.56±0.00	0.08±0.00	0.14±0.00	22.90±1.14	1.79±0.28	0.47±0.22	33.31±0.91	17.68±0.17	11.20±0.84
GWD	88.30±0.00	49.73±0.00	56.45±0.00	68.82±0.00	12.42±0.00	12.37±0.00	23.08±0.37	3.91±0.69	0.41±0.11	–	–	–
GED	89.55±0.00	43.33±0.00	51.02±0.00	52.24±0.07	3.92±0.23	-0.23±0.03	25.50±0.00	5.05±0.18	2.24±0.06	–	–	–
MMFD	98.80±0.00	88.37±0.00	94.49±0.00	72.60±0.00	14.18±0.00	19.67±0.00	23.68±1.31	5.99±0.78	1.96±0.33	35.97±0.48	18.12±0.61	13.74±1.97
MMFD$_{LR}$	98.80±0.00	88.37±0.00	94.49±0.00	72.49±0.13	13.98±0.25	19.49±0.23	23.62±0.83	6.55±0.42	2.13±0.22	36.54±0.56	18.31±0.74	13.99±2.18
MMFD$_{LR}$-KM	98.96±0.02	89.62±0.18	95.25±0.11	71.87±0.18	12.74±0.34	18.51±0.28	25.67±0.77	6.34±0.68	2.06±0.37	35.62±0.46	19.80±0.06	13.74±0.30
MFD	99.45±0.00	93.82±0.00	97.47±0.00	72.60±0.00	14.18±0.00	19.67±0.00	30.33±1.16	8.45±0.26	4.42±0.33	35.35±0.01	17.28±0.03	14.90±0.02
MFD-KD	99.02±0.00	90.01±0.34	95.51±0.18	72.39±0.30	14.06±0.40	19.24±0.57	26.25±1.44	7.54±0.69	2.21±0.60	34.86±0.22	19.70±0.29	14.46±0.36

What do distances buy us?

- They define **retrieval geometry**: nearest graphs, cluster boundaries, novelty scores.
- They provide **evaluation tools**: if a learned embedding destroys a trusted distance notion, that is a warning sign.
- They provide **scientific interpretability**: what kind of structural change counts as significant?
- They serve as **training signals**: metric learning, contrastive objectives, and OT-based regularizers.

Forward transition

Graph kernels take a crucial next step: instead of comparing graphs directly each time, we map them into an implicit feature space where comparison becomes an inner product.

Graph kernels: from similarity scores to feature maps

A graph kernel is a positive semidefinite similarity function

$$K(G, G') = \langle \phi(G), \phi(G') \rangle_{\mathcal{H}}$$

for some feature map ϕ in an RKHS $\mathcal{H}$.

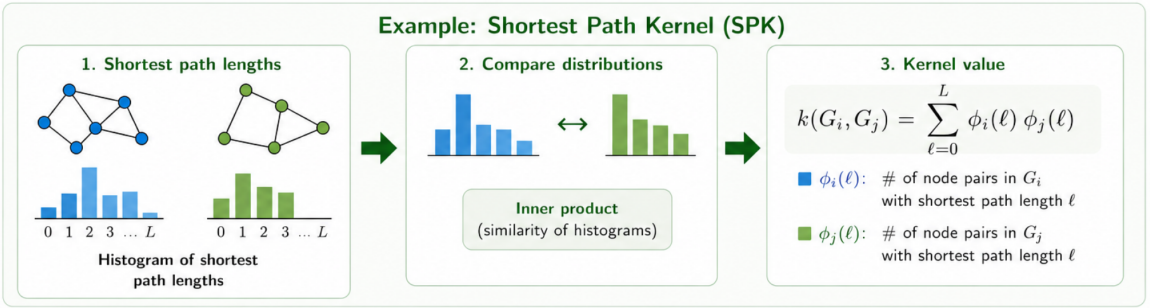
What kernels buy us

- mathematically controlled similarities,
- direct use with SVMs and kernel PCA,
- nonparametric learning with small datasets.

The core design choice

What graph substructures should be counted: walks, paths, graphlets, subtree patterns, or something learned?

Example: shortest-path kernel



A practical recipe for building a graph kernel

A large family of graph kernels can be viewed through the R-convolution principle:

$$K(G, G') = \sum_{r \in R(G)} \sum_{r' \in R(G')} \kappa(r, r'),$$

where $R(G)$ is a multiset of graph parts and κ is a PSD base kernel on parts. Then K is also PSD.

- Random walk kernels choose parts r as walks.
- Shortest-path kernels choose parts r as labeled shortest paths.
- Graphlet kernels choose parts r as small induced subgraphs.
- WL subtree kernels choose parts r as WL labels across refinement rounds.

Normalization often matters

$$\widehat{K}(G, G') = \frac{K(G, G')}{\sqrt{K(G, G) K(G', G')}}$$

mitigates graph-size effects and makes similarity scores easier to compare across graph pairs.

Classical graph kernels: a design space of counted substructures

Kernel family	Main idea	Typical strength	Complexity
Random walk	count matching walks in product graph	rich local connectivity patterns	$\mathcal{O}(n^3)$
Shortest path	compare all-pairs shortest path label tuples	more robust than unrestricted walks	$\mathcal{O}(n^4)$
Graphlet	count small induced subgraphs	explicit local motif statistics	$\mathcal{O}(n^k)$
Subtree / WL	iterative neighborhood hashing and counting	strong trade-off of power and efficiency	$\mathcal{O}(hl)$

Takeaway: Classical graph kernels differ mainly in which substructures are treated as features.

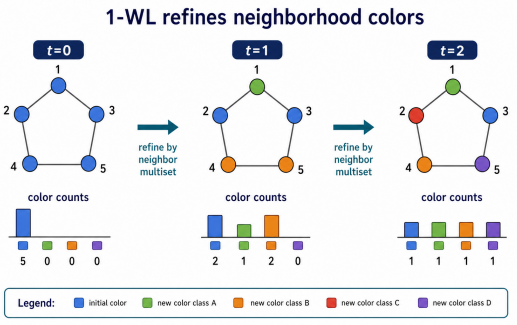
The decisive bridge: Weisfeiler-Lehman (WL) color refinement

1-WL update

For node colors $c_v^{(t)}$,

$$c_v^{(t+1)} = h\Big(c_v^{(t)},\ \big\{\!\big\{ c_u^{(t)} : u \in \mathcal{N}(v) \big\}\!\big\}\Big).$$

Repeated refinement gives a progressively richer rooted-subtree description.



Why this matters

WL is simultaneously a graph isomorphism heuristic, a feature extractor for kernels, and the theoretical yardstick for message-passing GNNs.

WL kernel: powerful because it is simple

The WL subtree kernel counts color histograms generated across refinement rounds:

$$K_{\mathrm{WL}}(G, G') = \sum_{t=0}^{T} \left\langle h_G^{(t)}, h_{G'}^{(t)} \right\rangle .$$

- Efficient and highly competitive for graph classification.
- Captures increasingly large rooted neighborhoods.
- Provides an explicit bridge from counted substructures to neighborhood aggregation.

Takeaway: WL became central because it offers a rare combination of efficiency, interpretability, and theory. Shervashidze et al. (JMLR 2011).

Kernel view versus message-passing view

Kernel perspective

Count subtree patterns explicitly, then compare via an RKHS inner product.

GNN perspective

Learn to aggregate neighborhood information using trainable functions.

Deep connection

The expressive power of standard message-passing GNNs is upper-bounded by 1-WL. So the same refinement principle governs both a classical kernel and a modern neural architecture.

Modern kernels: from handcrafted counts to learned similarities

- **Deep graph kernels:** learn latent similarities between substructures instead of treating them as unrelated tokens. [Yanardag & Vishwanathan, KDD 2015]
- **Wasserstein WL kernels:** replace histogram matching by transport-aware matching over WL features. [Togninalli et al., NeurIPS 2019]
- **MMD graph kernels:** compare learned node distributions using maximum mean discrepancy. [Sun & Fan, ICLR 2024]
- **Learnable kernel density ideas:** bridge graph kernels and nonparametric density estimation. [Wang et al., ICML 2026]

Trend

The kernel family did not vanish; it absorbed ideas from representation learning, metric learning, and distribution matching.

Modern graph kernels in formula form (I)

Deep graph kernels

Learn similarities among substructure embeddings rather than using a strict one-hot basis:

$$K_{\mathrm{DGK}}(G, G') = \phi(G)^{\top} M \, \phi(G').$$

Here M encodes similarity among latent substructure tokens rather than treating each substructure as unrelated.

Modern graph kernels in formula form (I)

Deep graph kernels

Learn similarities among substructure embeddings rather than using a strict one-hot basis:

$$K_{\mathrm{DGK}}(G, G') = \phi(G)^{\top} M \, \phi(G').$$

Here M encodes similarity among latent substructure tokens rather than treating each substructure as unrelated.

Wasserstein-WL kernels

Convert WL features to empirical measures μ_G and compare them by optimal transport:

$$K_{\mathrm{WWL}}(G, G') = \exp\big(-\gamma\, W(\mu_G, \mu_{G'})\big).$$

The key step is to replace exact histogram overlap by transport between refined subtree embeddings.

Modern graph kernels in formula form (II)

MMD graph kernels

Treat node embeddings as samples from a graph-specific distribution:

$$\mu_G = \frac{1}{|V|} \sum_{v \in V} k(z_v, \cdot), \qquad \mathrm{MMD}^2(G, G') = \|\mu_G - \mu_{G'}\|^2_{\mathcal{H}}.$$

A kernelized similarity is then

$$K_{\mathrm{MMD}}(G, G') = \exp\big(-\tau\, \mathrm{MMD}^2(G, G')\big).$$

Interpretation

These methods keep the *kernel viewpoint* but enrich it with learned embeddings, distribution matching, and transport-aware geometry.

Kernels in 2026: when should we still use them?

Good use cases

- graph classification with limited labels,
- interpretable similarity search,
- strong handcrafted domain priors,
- hybrid systems where kernels supervise or calibrate neural models.

Less ideal

- billion-edge graphs,
- highly dynamic graphs,
- settings requiring end-to-end feature learning from raw multimodal inputs.

Takeaway: Kernels remain highly relevant as strong baselines, small-data methods, and theoretical anchors.

Why revisit pre-neural graph embeddings?

Because their biases still drive modern models

- **Graph cuts:** preserve community separation.
- **Spectral embeddings:** preserve edge smoothness.
- **Random walks:** preserve contextual co-occurrence.
- **Graph2Vec:** preserve substructure statistics.

Neural models make the bias trainable

- GCN-like layers learn low-pass smoothing.
- MPNNs learn multi-hop neighborhood summaries.
- Graph transformers learn global attention geometry.
- SSL objectives learn which perturbations preserve meaning.

Bridge intuition

Before GNNs, graph learning already knew how to turn cuts, walks, spectra, and patterns into geometry. Modern representation learning keeps these principles but learns them end to end.

Spectral clustering as graph embedding

Let $\mathbf{L} = \mathbf{D} - \mathbf{A}$ be the combinatorial Laplacian and $\mathbf{L}_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ the normalized Laplacian.

A standard spectral relaxation solves

$$\min_{F \in \mathbb{R}^{n \times k}} \operatorname{Tr}(F^\top \mathbf{L} F) \quad \text{s.t.} \quad F^\top F = \mathbf{I}$$

or, in normalized form,

$$\min_{Y^\top \mathbf{D} Y = \mathbf{I}} \operatorname{Tr}(Y^\top \mathbf{L} Y).$$

The solution is given by the bottom-k eigenvectors of a Laplacian variant, after which we cluster rows by k-means.

Interpretation

Each row of the eigenvector matrix is already a **node embedding**. Nearby rows mean nodes have similar low-frequency graph signals.

Connection to GNNs

Laplacian smoothing and low-frequency filtering reappear later in spectral GNNs and GCN-like layers.

Takeaway: Spectral methods expose a core graph-learning prior: useful representations often live in low-frequency eigenspaces of the graph. Spectral clustering [Ng, Jordan, and Weiss, NeurIPS 2002]; Laplacian Eigenmaps [Belkin and Niyogi, Neural Computation 2003]; tutorial [von Luxburg, Statistics and Computing 2007].

Random-walk embeddings: DeepWalk and node2vec

A shallow random-walk method learns node embeddings by a Skip-Gram-style objective:

$$\max_{\{z_u, \tilde{z}_u\}} \sum_{u \in V} \sum_{v \in \mathcal{C}_S(u)} \log p(v \mid u), \qquad p(v \mid u) = \frac{\exp(\tilde{z}_v^\top z_u)}{\sum_{w \in V} \exp(\tilde{z}_w^\top z_u)}.$$

Here $\mathcal{C}_S(u)$ denotes random-walk context nodes; in practice the softmax is approximated by negative sampling. Node2vec modifies the transition probability by a second-order bias:

$$\pi_{vx} \propto \alpha_{pq}(t, x)\, w_{vx}, \qquad \alpha_{pq}(t, x) = \begin{cases} 1/p, & d_{tx} = 0, \\ 1, & d_{tx} = 1, \\ 1/q, & d_{tx} = 2, \end{cases}$$

where t is the previous node, v is the current node, and $x \in \mathcal{N}(v)$ is the candidate next node.

Interpretation

These methods turn *graph co-occurrence statistics* into Euclidean geometry, much like word2vec turns token contexts into lexical geometry.

Takeaway: Changing the walk changes which graph relations are made close in embedding space.

DeepWalk [Perozzi, Al-Rfou, and Skiena, KDD 2014]; node2vec [Grover and Leskovec, KDD 2016]; Skip-Gram [Mikolov et al., NeurIPS 2013].

Node2vec intuition: BFS versus DFS bias

Neighborhood bias

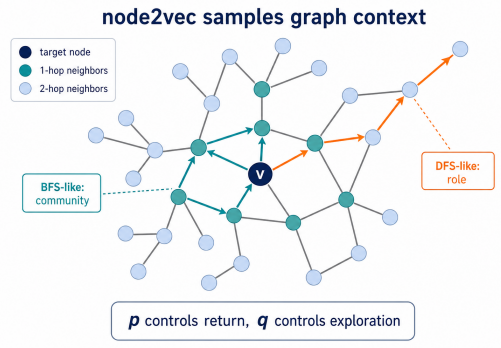
A **BFS-like** walk keeps returning to nearby nodes, which tends to preserve *homophily* and community membership.

Role bias

A **DFS-like** walk pushes farther away, which tends to preserve *structural equivalence* or similar graph roles.

Why this is historically important

The graph community had already learned that *how we sample context* determines *what geometric relations an embedding will preserve*. GNNs later made that context extractor trainable.



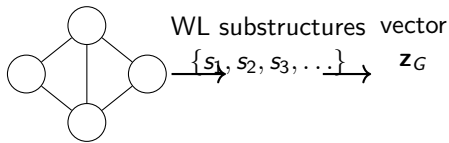
Graph2Vec: from node context to graph context

Goal: learn a fixed-dimensional vector for each graph:

$$G_i \longrightarrow \mathbf{z}_i \in \mathbb{R}^d$$

Key Idea

- Treat each graph as a "document"
- Treat local substructures as "words"
- Learn graph embeddings using a Doc2Vec-style objective



Graph2Vec learns graph embeddings by predicting local structural patterns from the whole-graph representation.

Substructures

- Usually extracted by Weisfeiler–Lehman relabeling
- Capture local topology around nodes

Takeaway: Graph-level representation learning can be viewed as choosing a vocabulary of substructures and learning which graph-level contexts explain them. Graph2Vec [Narayanan et al., 2017]; Doc2Vec analogy [Le and Mikolov, ICML 2014]; WL substructures [Shervashidze et al., JMLR 2011].

Classical embedding methods: three geometries

Method family	Preserved relation	Limitation that motivates neural models
Spectral / cut	nodes connected through low-conductance communities and smooth graph signals	usually transductive; expensive or unstable on changing graphs
Random walk	nodes appearing in similar sampled contexts	topology-centric; context distribution is mostly hand-designed
Graph2Vec / WL	graphs sharing local substructure vocabularies	sparse/high-dimensional patterns; weak feature adaptation

Transition

GNNs keep the same ambition but replace fixed feature maps and fixed sampling rules by trainable, permutation-respecting encoders.

From shallow graph embedding to neural graph models

Family	Core mechanism	Main limitation / role
Spectral embedding	Laplacian eigenvectors / cut relaxation	transductive, global, and weakly feature-conditioned
Random-walk embedding	context prediction from sampled walks	topology-centric co-occurrence; limited task adaptation
Message-passing GNNs	trainable local aggregation over features and edges	strong locality bias; vulnerable to over-smoothing and over-squashing
Graph transformers	global attention with structural bias	higher cost; positional / structural encoding becomes decisive

Takeaway: Modern graph neural models unify spectral smoothness, contextual proximity, and task-driven feature learning; the main evolution is where structural bias is injected.

Representative roots: spectral CNNs [Bruna et al., ICLR 2014; Defferrard et al., NeurIPS 2016], random-walk embeddings [Perozzi et al., KDD 2014; Grover and Leskovec, KDD 2016], original GNNs [Scarselli et al., IEEE TNN 2009].

Unified notation and graph-level readout

For layer ℓ, node states are

$$\mathbf{H}^{(\ell)} = [h_1^{(\ell)}, \dots, h_n^{(\ell)}]^\top \in \mathbb{R}^{n \times d_\ell}, \qquad \mathbf{H}^{(0)} = \mathbf{X}.$$

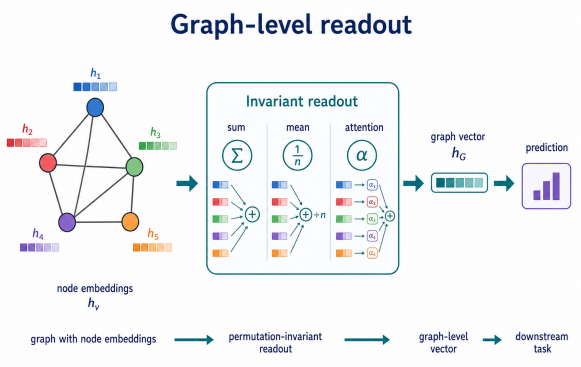
- Local models use $\mathbf{A}$ and neighborhoods $\mathcal{N}(v)$.
- Global-attention models form pairwise interactions between tokens.
- A graph embedding must be permutation-invariant:

$$h_G = \mathrm{READOUT}\{h_v^{(L)} : v \in V\}.$$

A universal set-function form is

$$h_G = \rho\left(\sum_{v \in V} \psi(h_v^{(L)})\right).$$

Takeaway: Node encoders must be permutation-equivariant; graph encoders additionally require a permutation-invariant readout.



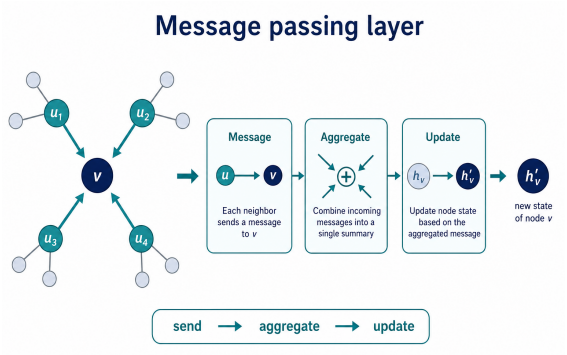
- **Simple readouts:** sum, mean, max.
- **Adaptive readouts:** attention pooling (2016), graph token (2021), Set Transformer (2019).
- **Hierarchical readouts:** DiffPool (2018), SAGPool (2019), MinCutPool (2020).

Message passing as the core abstraction

A generic edge-aware MPNN layer is

$$m_{uv}^{(\ell)} = M_\ell\big(h_u^{(\ell)}, h_v^{(\ell)}, e_{uv}\big), \qquad m_v^{(\ell)} = \bigoplus_{u \in \mathcal{N}(v)} m_{uv}^{(\ell)},$$

$$h_v^{(\ell+1)} = U_\ell\big(h_v^{(\ell)}, m_v^{(\ell)}\big).$$



- $\bigoplus$: sum, mean, max, or attention-weighted sum.
- Each layer expands the receptive field by one hop.
- The architecture is determined by M_ℓ, $\bigoplus$, and U_ℓ.

Takeaway: MPNNs are neural versions of iterative neighborhood summarization. Message Passing Neural Networks [Gilmer et al., ICML 2017].

From spectral filters to GCN

Spectral graph convolution starts from the Laplacian $L = U\Lambda U^\top$:

$$g_\theta(L)x = U\, g_\theta(\Lambda)\, U^\top x.$$

Chebyshev approximation avoids explicit eigendecomposition:

$$g_\theta(L)x \approx \sum_{k=0}^{K} \theta_k\, T_k(\widetilde{L})x, \quad \text{Scaled Laplacian } \widetilde{L} = \frac{2}{\lambda_{\max}} L - \mathbf{I}.$$

The first-order simplification with renormalization gives GCN:

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\widetilde{\mathbf{D}}^{-1/2}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(\ell)}W^{(\ell)}\right), \qquad \widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}.$$

Interpretation

GCN is a learned low-pass graph filter: smooth over local neighborhoods, then mix feature channels.

Spectral CNN [Bruna et al., ICLR 2014]; ChebNet [Defferrard et al., NeurIPS 2016]; GCN [Kipf and Welling, ICLR 2017].

Representative MPNN designs: sampling, attention, and injectivity

GraphSAGE [Hamilton et al., NeurIPS 2017]: sampled neighborhood aggregation

$$h_v^{(\ell+1)} = \sigma\left(W^{(\ell)} \left[h_v^{(\ell)} \,\|\, \mathrm{MEAN}_{u \in \mathcal{N}(v)} \, h_u^{(\ell)} \right] \right).$$

- Samples neighborhoods for scalable mini-batch training.
- Supports inductive generalization to unseen nodes.

GAT [Veličković et al., ICLR 2018]: attention-weighted neighborhood aggregation

$$h_v^{(\ell+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \alpha_{vu}^{(\ell)} W^{(\ell)} h_u^{(\ell)} \right),$$

$$\alpha_{vu}^{(\ell)} \propto \exp\left(\mathrm{LeakyReLU}(a^\top [Wh_v \| Wh_u]) \right).$$

- Learns neighbor-specific importance weights.
- Replaces fixed degree normalization with content-adaptive aggregation.

GIN [Xu et al., ICLR 2019]: injective multiset aggregation

$$h_v^{(\ell+1)} = \mathrm{MLP}_\ell\left((1+\epsilon_\ell) h_v^{(\ell)} + \sum_{u \in \mathcal{N}(v)} h_u^{(\ell)} \right).$$

- Uses sum aggregation to preserve multiset information.
- Matches 1-WL expressive power under suitable conditions.

Takeaway: GraphSAGE emphasizes scalable inductive aggregation, GAT introduces adaptive neighbor weighting, and GIN clarifies why injective multiset aggregation is central to WL-level expressiveness.

MPNNs through the WL lens

Landmark result

Standard message-passing GNNs are at most as powerful as the 1-WL color refinement test in distinguishing non-isomorphic graphs.

- If 1-WL cannot distinguish two graphs, many standard MPNNs cannot distinguish them either.
- GIN reaches this upper bound with injective aggregation and sufficiently expressive MLPs.

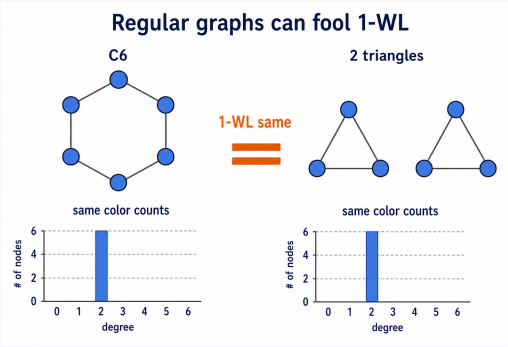
The neighborhood is a multiset, so aggregation should preserve multiset information:

$$F(\{x_1, \ldots, x_m\}) = \rho\left(\sum_{i=1}^{m} \psi(x_i)\right).$$

Mean aggregation may collapse different multisets with the same average; sum plus a rich MLP is less lossy.

Takeaway: The WL connection explains both why message passing works and where its expressive ceiling comes from. Expressive power of GNNs [Xu et al., ICLR 2019]; WL-GNN correspondence and higher-order GNNs [Morris et al., AAAI 2019].

What 1-WL misses, and how to go beyond it



1-WL may fail on: certain regular graphs, cycle and counting structures, long-range dependencies, and symmetries invisible to local color refinement.

Ways to increase power

- higher-order / k-GNNs on tuples or tensors,
- positional and structural encodings,
- subgraph, motif, or counting features,
- rewiring and global attention,
- geometric or Riemannian inductive bias.

Trade-off

Greater expressive power usually costs more computation, weaker locality bias, or more delicate regularization.

Representative routes: higher-order networks [Maron et al., NeurIPS 2019; Morris et al., AAAI 2019], structural/positional encodings [Dwivedi et al., ICLR 2022; Lim et al., ICLR 2023], subgraph aggregation [Bevilacqua et al., ICLR 2022], Riemannian geometry [Wang et al., AAAI 2026].

GNN summary before moving to graph transformers

Question	MPNN answer	Limitation
How does information move?	local message passing over edges	depth needed for long-range reasoning
How is permutation handled?	equivariant aggregation and invariant readout	aggregation may be non-injective
How much structure is captured?	at most 1-WL for standard MPNNs	misses some regular, counting, and global patterns
Why still useful?	efficient $\mathcal{O}(\lvert E\rvert d)$ local inductive bias	may need structural augmentation

Takeaway: GNNs are best understood as efficient, local, WL-style representation learners; graph transformers add a global communication path but must reintroduce graph structure explicitly. MPNN abstraction [Gilmer et al., ICML 2017]; WL expressiveness [Xu et al., ICLR 2019; Morris et al., AAAI 2019]; long-range bottlenecks [Alon and Yahav, ICLR 2021].

Why graph transformers?

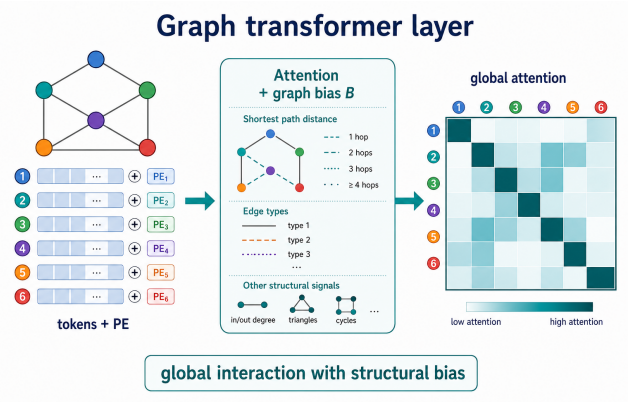
Motivation

Message passing is local by construction. Attention offers global interaction, content-adaptive weighting, and easier integration with foundation-model-style pretraining.

- capture long-range dependencies,
- integrate diverse structural encodings,
- use tokenized graph representations,
- support large-scale pretraining recipes.

Design consequence

Attention alone is graph-agnostic; structural bias must be injected back into the model.



Transformer attention [Vaswani et al., NeurIPS 2017]; graph transformers and structural encodings [Ying et al., NeurIPS 2021; Dwivedi et al., ICLR 2022; Rampášek et al., NeurIPS 2022].

A graph transformer layer: attention plus structural bias

Given node states $\mathbf{H} \in \mathbb{R}^{n \times d}$,

$$Q = \mathbf{H}W_Q, \qquad K = \mathbf{H}W_K, \qquad V = \mathbf{H}W_V.$$

Attention with graph-aware bias is

$$\mathbf{P} = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_h}} + B\right), \qquad \mathbf{H}' = \mathbf{P}VW_O.$$

A practical parameterization is

$$B_{ij} = b^{\mathrm{SPD}}_{d_G(i,j)} + b^{\mathrm{edge}}_{e_{ij}} + b^{\mathrm{PE}}_{ij} + b^{\mathrm{global}}_{ij}.$$

- Without B, attention is permutation-equivariant but graph-agnostic.
- With B, attention can encode shortest paths, edge types, centrality, Laplacian/random-walk relations, or graph tokens.

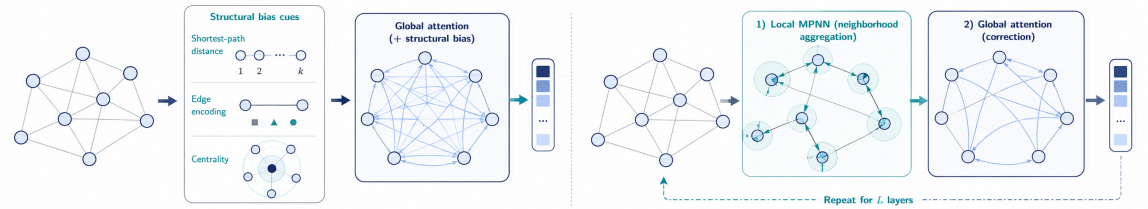
Takeaway: Graph transformers encode inductive bias in the attention geometry rather than in a fixed local aggregator. Scaled dot-product attention [Vaswani et al., NeurIPS 2017]; shortest-path and centrality bias [Ying et al., NeurIPS 2021]; learnable structural/positional encodings [Dwivedi et al., ICLR 2022].

Graph transformer design patterns

Design choice	Examples	Purpose
Structural bias	shortest path, edge type, centrality, Laplacian / random-walk PE	restore graph inductive bias
Hybrid blocks	MPNN + attention, sparse + dense attention	combine local stability with global reach
Tokenization	nodes, edges, subgraphs, random walks, graph-text pairs	decide what the transformer sees
Readout mechanism	class token, graph token, pooling, Set Transformer	support graph-level tasks
Scaling strategy	sparse attention, landmarks, batching by size	control n^2 memory and compute

Takeaway: Most graph transformer variants differ less by attention itself than by tokenization, structural encoding, and scalability design. Design anchors: Transformer [Vaswani et al., NeurIPS 2017], Graphormer [Ying et al., NeurIPS 2021], LSPE [Dwivedi et al., ICLR 2022], GPS/GraphGPS [Rampášek et al., NeurIPS 2022], Set Transformer readout [Lee et al., ICML 2019].

Graphormer and GPS as two useful archetypes



Graphormer: global attention with rich bias [Ying et al., NeurIPS 2021]

Pure transformer backbone plus carefully designed structural encodings such as shortest-path distance, edge encoding, and centrality.

$$\mathbf{H}^{(\ell+1)} = \mathrm{FFN}_\ell\big(\mathrm{Attn}_\ell(\mathbf{H}^{(\ell)}; B) + \mathbf{H}^{(\ell)}\big).$$

GPS: local-global hybridization [Rampášek et al., NeurIPS 2022]

Local message passing plus global attention in each layer:

$$\widetilde{\mathbf{H}}^{(\ell)} = \mathrm{MPNN}_\ell(\mathbf{H}^{(\ell)}, \mathbf{A}, E),$$

$$\mathbf{H}^{(\ell+1)} = \mathrm{FFN}_\ell\big(\widetilde{\mathbf{H}}^{(\ell)} + \mathrm{Attn}_\ell\big(\widetilde{\mathbf{H}}^{(\ell)} + \mathrm{PE}\big)\big).$$

Takeaway: Graphormer pushes the burden onto global attention plus structural bias; GPS keeps a strong local prior and adds global correction.

Complexity and scaling: graph transformers are not free

Family	Typical layer cost	Main bottleneck
MPNN / GCN / GIN	$\mathcal{O}(\lvert E\rvert d)$	local propagation; limited long-range reach
Dense graph transformer	$\mathcal{O}(n^2 d)$	all-pairs attention memory and compute
Sparse / hybrid transformer	between $\mathcal{O}(\lvert E\rvert d)$ and $\mathcal{O}(n^2 d)$	balancing structural bias and reach
Higher-order GNN	often superlinear or combinatorial	tuple / subgraph explosion

Practical moral

Long-range power is valuable, but scalability, regularization, and careful positional design usually decide whether that power is realized.

Linear-edge GCN-style propagation [Kipf and Welling, ICLR 2017]; dense structural attention [Ying et al., NeurIPS 2021]; scalable hybrid recipe [Rampášek et al., NeurIPS 2022].

Model selection: GNN or graph transformer?

Scenario	MPNN family	Graph transformer	Why
Small homophilous graph tasks	strong	moderate	local bias often enough
Long-range dependency	weak to moderate	strong	global interaction matters
Resource-limited training	strong	weaker	attention is heavier
Foundation-model-style pre-training	moderate	strong	tokenized sequence view is convenient

How to choose?

Does the task need long-range interaction? Is the graph large enough that n^2 attention is problematic? Do we have reliable structural encodings?

- Carefully tuned MPNNs can close much of the gap on some benchmarks when training, depth, and positional design are handled well.
- Global attention helps most when the label depends on distant interactions, weak locality, or rich structural side information.
- Hybrid local-global designs are often more stable because they retain the locality bias of GNNs while adding a global communication path.

Takeaway: Architecture choice should follow the dependency pattern of the task.
Graph transformers are not replacements for GNNs; they are a different way to allocate inductive bias, communication range, and computation.

Why self-supervised graph learning?

Motivation

Graph labels are expensive, sparse, and domain-specific, while unlabeled structure is abundant. Pretraining gives the encoder a useful geometry before downstream supervision.

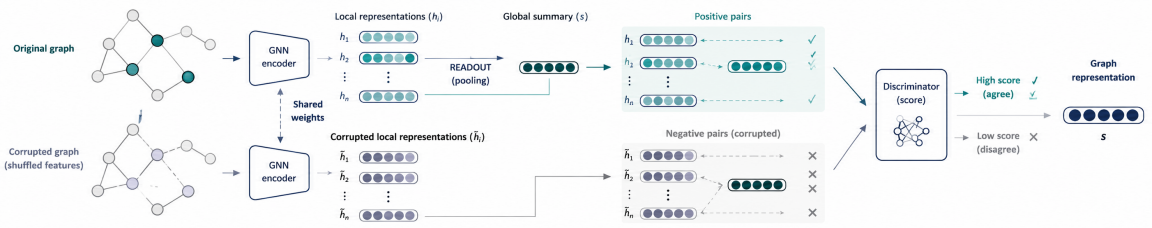
The key question

What should remain close under the pretext task, and what information must not be collapsed?

Objective	Intuition
Contrastive	augmentations of the same graph should be close
Predictive	masked nodes, edges, or motifs should be recoverable
InfoMax	local and global summaries should agree
Graph-theoretic	invariants / entropy should be preserved

Takeaway: SSL is where graph distances quietly re-enter: the loss specifies what should stay close, what should separate, and what structural information survives compression.

Deep Graph Infomax (DGI): local-global mutual information



Core idea

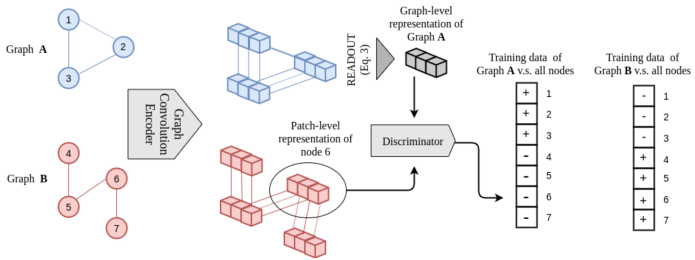
Maximize mutual information between local node representations and a global graph summary, while contrasting against corrupted views.

$$\max \sum_i \log \sigma(h_i^\top s) + \sum_i \log\left(1 - \sigma(\tilde{h}_i^\top s)\right).$$

- elegant and influential,
- shows that graph structure itself can supervise representation quality.

Velickovic et al. (ICLR 2019).

InfoGraph: graph-level mutual information



Main move

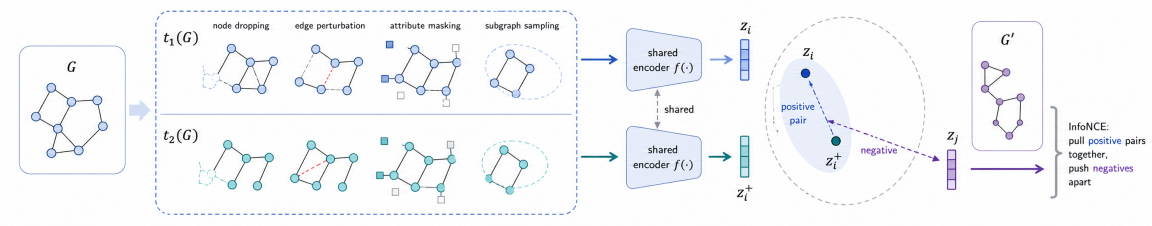
Maximize mutual information between graph-level embeddings and embeddings of local patches or substructures.

Why useful

Directly targets graph-level tasks rather than only node-level representations.

Takeaway: InfoGraph makes explicit the idea that a good graph embedding should preserve informative substructure content. Sun et al. (ICLR 2020).

GraphCL: contrastive learning with graph augmentations



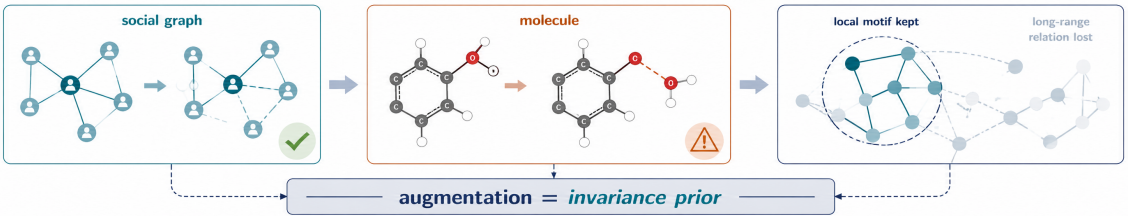
Given two augmentations t_1, t_2 of the same graph, maximize agreement between $f(t_1(G))$ and $f(t_2(G))$ while separating different graphs.

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\sum_j \exp(\text{sim}(z_i, z_j)/\tau)}.$$

- augmentations include node dropping, edge perturbation, attribute masking, subgraph sampling;
- simple idea, broad impact, strong transfer baseline.

You et al. (NeurIPS 2020).

The augmentation question is the real modeling question



Hidden assumption behind contrastive learning

Two augmented views of the same graph should preserve the *same semantics*. But what counts as semantics is domain-dependent.

- random edge dropping may be harmless in social graphs,
- but disastrous in molecules if a functional bond disappears.
- subgraph sampling may preserve local motifs,
- but remove long-range constraints critical for reasoning.

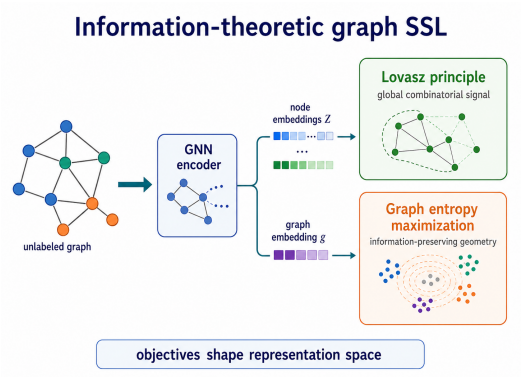
Takeaway: In graph SSL, augmentation design is equivalent to specifying an invariance prior.

Information-theoretic perspectives beyond contrast

What changes from contrastive SSL?

Contrastive learning specifies invariances by augmentations. Information-theoretic objectives instead ask what global graph information the representation should retain.

- **Lovász principle:** uses a tractable global graph invariant as the organizing signal.
- **Graph entropy maximization:** encourages informative and diverse node/graph representations through a graph entropy objective.



Takeaway: The objective shapes representation geometry before labels: similar graphs should be close and information-preserving. Lovász principle: Sun, Ding, and Fan (NeurIPS 2023). Graph entropy maximization: Sun, Wang, Ding, and Fan (ICML 2024).

GeMax motivation: graph entropy is structural

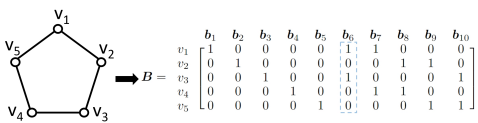
Körner graph entropy [Körner, 1973]

Let B be the indicator matrix of independent sets and the vertex-packing polytope for G:

$$\mathrm{VP}(G) = \{a \in \mathbb{R}^{|V|} : a = B\lambda, \lambda \geq 0, \sum_i \lambda_i = 1\}.$$

For a probabilistic graph (G, P),

$$H_k(G, P) = \min_{a \in \mathrm{VP}(G)} \sum_{i=1}^{|V|} -P_i \log(a_i).$$

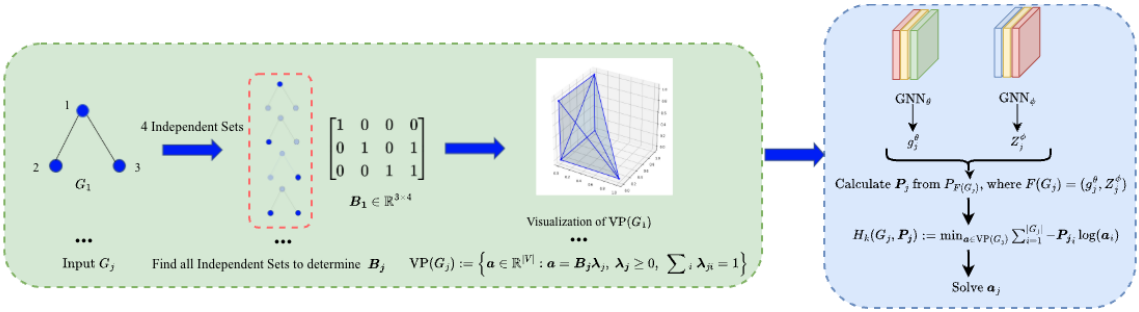


From InfoMax to GeMax

- **InfoMax:** maximizes estimated MI between graph and node embeddings; structure enters only through the encoder and augmentations.
- **GeMax:** maximizes a graph entropy whose feasible region depends explicitly on independent sets through $\mathrm{VP}(G)$.
- **Key difference:** the objective carries non-adjacency and combinatorial constraints, not just distributional spread.

Takeaway: Graph entropy depends on $\mathrm{VP}(G)$, so the objective is structural rather than only distributional.

GeMax: learnable objective



Let $F = (F_g, F_Z)$ produce graph and node representations:

$$g_j^\theta = F_g(A_j, X_j; \theta), \qquad Z_j^\phi = F_Z(A_j, X_j; \phi).$$

Define a vertex distribution by closeness to the graph representation:

$$P_i(g_j^\theta, Z_j^\phi) = \frac{\exp(-\|z_{j,i}^\phi - g_j^\theta\|_2^2)}{\sum_{\ell \in V_j} \exp(-\|z_{j,\ell}^\phi - g_j^\theta\|_2^2)}.$$

Then the GeMax objective is: $\max_{\theta,\phi} \sum_{j=1}^N H_k(G_j, P_j)$, $\quad H_k(G_j, P_j) = \min_{a_j \in \mathrm{VP}(G_j)} \sum_{i=1}^{n_j} -P_{j,i} \log(a_{j,i})$.

GeMax: approximation and alternating optimization

Direct GeMax is hard because $\mathrm{VP}(G)$ requires independent-set structure. The paper relaxes it using orthonormal representations and a sub-vertex-packing regularizer:

$$\mathcal{L}_{H_k} = \sum_{j=1}^{N} \sum_{i=1}^{n_j} -P_i(g_j^{\theta}, Z_j^{\phi}) \log(a_{j(i)}),$$

$$\mathcal{L}_{\mathrm{orth}} = \sum_j \left\| M_j \odot (Z_j^{\phi}(Z_j^{\phi})^{\top} - I_{n_j}) \right\|_F^2, \qquad \mathcal{L}_{\text{s-vp}} = \sum_j \left\| D_{a_j} Z_j^{\phi}(Z_j^{\phi})^{\top} D_{a_j} - D_{a_j}^2 \right\|_F^2.$$

Representation update

$$(\theta^{t+1}, \phi^{t+1}) = \arg\max_{\theta,\phi} \left[\mathcal{L}_{H_k} - \mu \mathcal{L}_{\mathrm{orth}} - \gamma \mathcal{L}_{\text{s-vp}} \right].$$

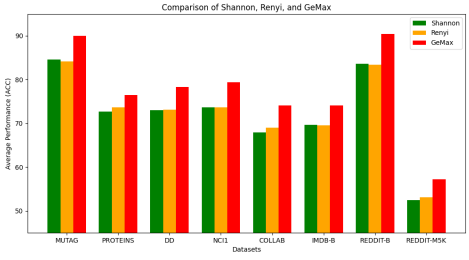
Packing-vector update

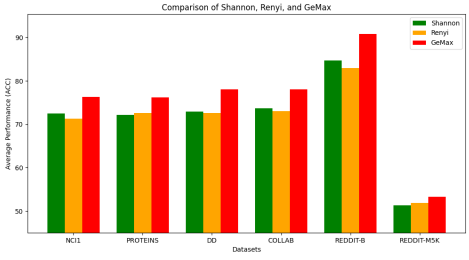
$$\mathcal{A}^{t+1} = \arg\min_{\mathcal{A} \in \mathcal{C}} \left[\mathcal{L}_{H_k} + \gamma \mathcal{L}_{\text{s-vp}} \right], \quad 0 \le a_{ij} \le 1.$$

Takeaway: The algorithm alternates between learning representations and fitting the entropy geometry; mini-batch training makes it scalable in the number of graphs.

GeMax: graph entropy versus generic entropies

GeMax improves average accuracy across multiple unsupervised and semi-supervised pipelines.



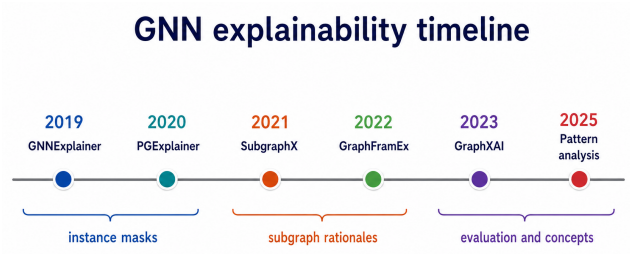


Interpretation

Maximizing Shannon or Renyi entropy spreads probability mass over vertices, but it does not encode the independent-set geometry of G. GeMax keeps the entropy objective tied to graph topology through $\mathrm{VP}(G)$ or its tractable approximation.

Takeaway: The empirical advantage is not merely "more entropy"; it is entropy constrained by graph structure.

A brief timeline of GNN explainability

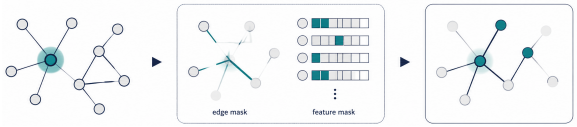


Three classical milestones to remember

- **GNNExplainer (2019):** optimize sparse masks for a single prediction.
- **PGExplainer (2020):** learn a reusable parametric explainer across instances.
- **SubgraphX (2021):** search directly for a human-interpretable explanatory subgraph.

Takeaway: The methodological shift is from instance-wise attribution to generalizable rationale generation and then to subgraph-level explanation and evaluation.

GNNExplainer (NeurIPS 2019): sparse instance-wise mask optimization



For a trained predictor Φ, GNNExplainer learns a soft edge mask M and feature mask m for one target instance:

$$\max_{M,m} \; I(Y;\, G \odot \sigma(M),\, X \odot m)$$

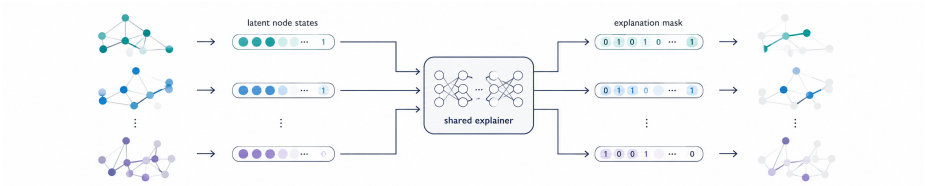
which is implemented through a prediction-preserving objective such as

$$\max_{M,m} \; \log P_\Phi(y \mid G \odot \sigma(M), X \odot m) - \lambda_1 \|\sigma(M)\|_1 - \lambda_2 H(\sigma(M)).$$

- **Strength:** first general post-hoc framework for node- and graph-level GNN explanations.
- **Limitation:** explanations are optimized per-instance and can be unstable or expensive.

Ying et al., NeurIPS 2019: "the first general, model-agnostic approach" for GNN prediction explanations.

PGExplainer (NeurIPS 2020): learn an explainer that generalizes



Instead of optimizing a fresh mask for each instance, PGExplainer trains an explainer network on latent node states:

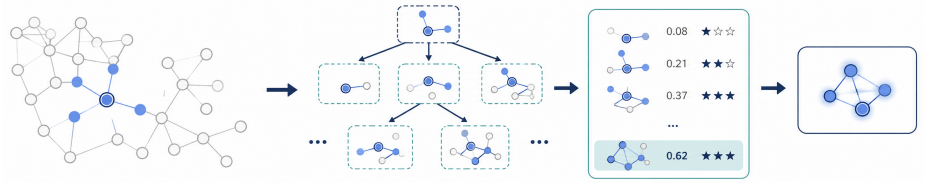
$$s_{uv} = \mathrm{MLP}\left([h_u^{(L)} \| h_v^{(L)} \| h_c]\right), \qquad \tilde{M}_{uv} \sim \mathrm{Concrete}(s_{uv}, \tau),$$

and maximizes the expected prediction score under sampled explanatory subgraphs.

- **Strength:** explanation generation amortizes across instances and naturally supports inductive settings.
- **Design lesson:** explanations can be *learned objects*, not just outputs of a post-hoc optimizer.

Luo et al., NeurIPS 2020: parametric explainer with improved generalization across instances.

SubgraphX (ICML 2021): explanation as a subgraph search problem



SubgraphX asks directly for an explanatory subgraph S rather than a node or edge heatmap. Its score is based on the marginal contribution of S:

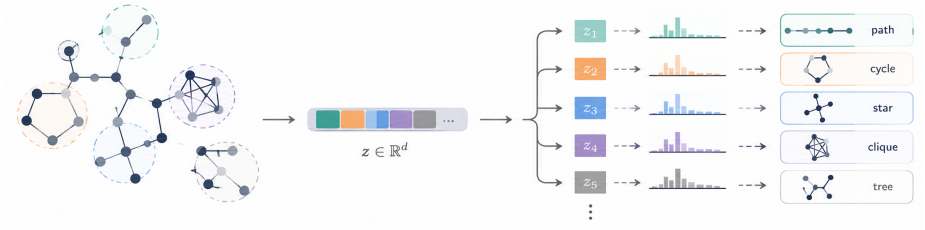
$$\phi(S) = \sum_{T \subseteq V \setminus S} \frac{|T|!\,(n - |T| - |S|)!}{(n - |S| + 1)!} \Big(f(T \cup S) - f(T) \Big),$$

with Monte Carlo tree search used to navigate the combinatorial search space.

- **Strength:** more human-intelligible rationales at the subgraph level.
- **Limitation:** computationally heavier than mask-based explainers.

Yuan et al., ICML 2021: Shapley-informed subgraph exploration for GNN explanations.

A modern example: graph pattern analysis



- decompose graph representations into interpretable pattern channels,
- attribute embedding dimensions or channels to human-readable structural motifs,
- support explanation at the representation and graph levels.

Broader lesson

Explainability is strongest when interpretation is built into the representation space itself, not bolted on after training.

Representative example: Wang et al. (IJCAI 2025).

PXGL: representation-level graph explainability

Representation-level question

Most XGL methods explain a model or an individual prediction. PXGL instead asks:

Which structural pattern families contribute to $\mathbf{g}_G$?

- Pattern families such as paths, trees, cycles, cliques, and graphlets carry domain semantics.
- Graph kernels expose pattern counts but ignore features and can be high-dimensional.
- GNNs use features and implicit structure but hide the pattern content of $\mathbf{g}_G$.

Pattern view

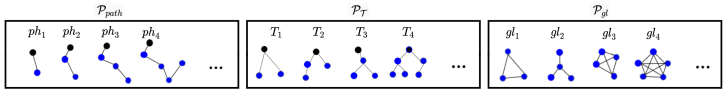
For a pattern family $\mathcal{P}_m$, define a pattern descriptor

$$\mathbf{h}^{(m)}(G) = \phi(G; \mathcal{P}_m),$$

where each coordinate counts or summarizes occurrences of patterns in $\mathcal{P}_m$.

PXGL Key Idea

Make the representation itself a weighted mixture of interpretable pattern channels.



Wang, Sun, Ding, Fan (IJCAI 2025): Explainable Graph Representation Learning via Graph Pattern Analysis.

PXGL-EGK: pattern weights as explanations

Learnable ensemble graph kernel
Given M pattern kernels $\{K_{\mathcal{P}_1}, \ldots, K_{\mathcal{P}_M}\}$,

$$K_{ij}(\boldsymbol{\lambda}) = \sum_{m=1}^{M} \lambda_m K_{\mathcal{P}_m}(G_i, G_j), \qquad \boldsymbol{\lambda} \in \Delta_M.$$

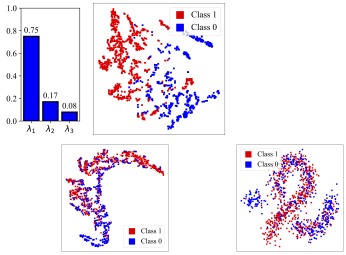
The simplex coordinate λ_m is an explicit task-level importance score for $\mathcal{P}_m$.
Kernel training objectives

$$\boldsymbol{\lambda}^* = \underset{\mathbf{1}^\top \boldsymbol{\lambda}=1,\ \boldsymbol{\lambda}\geq 0}{\arg\min} \ \mathcal{L}_{\mathrm{ker}}(\boldsymbol{\lambda}).$$

$$Z_i^+ = \sum_{k\neq i} \mathbb{I}[y_i = y_k] K_{ik}, \qquad Z_i^- = \sum_{k} \mathbb{I}[y_i \neq y_k] K_{ik},$$

$$\mathcal{L}_{\mathrm{SCL}} = -\sum_{i\neq j} \mathbb{I}[y_i = y_j] \Big(\log K_{ij} - \log(Z_i^+ + \mu Z_i^-) \Big).$$

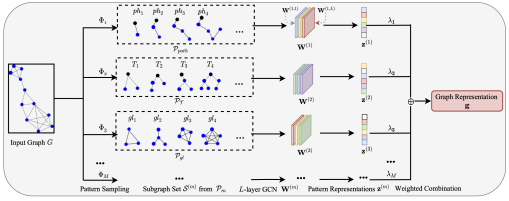
$$\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}(\mathbf{K}(\boldsymbol{\lambda}), \mathbf{K}'(\boldsymbol{\lambda})), \quad K'_{ij} = \frac{K_{ij}^2 / r_j}{\sum_{j'} K_{ij'}^2 / r_{j'}}.$$



PROTEINS example: random-walk/path, subtree, and graphlet kernels are combined by learned weights.

PXGL-EGK: supervised contrastive kernel learning and unsupervised KL self-training over kernel matrices.

PXGL-GNN: feature-aware pattern channels



Algorithmic structure

1. Sample subgraphs $\mathcal{S}^{(m)}(G)$ matching pattern family $\mathcal{P}_m$.
2. Encode each sampled pattern set with a dedicated GNN $F(\cdot; \mathcal{W}^{(m)})$.
3. Combine pattern channels with interpretable weights $\boldsymbol{\lambda}$.

Interpretation: λ_m measures how much family $\mathcal{P}_m$ contributes to $\mathbf{g}_G$.

Pattern-channel representation

$$\mathbf{z}_G^{(m)} = \frac{1}{|\mathcal{S}^{(m)}(G)|} \sum_{S \in \mathcal{S}^{(m)}(G)} F(\mathbf{A}_S, \mathbf{X}_S; \mathcal{W}^{(m)}),$$

$$\mathbf{g}_G = \sum_{m=1}^{M} \lambda_m \mathbf{z}_G^{(m)}, \qquad \mathbf{1}^\top \boldsymbol{\lambda} = 1,\ \boldsymbol{\lambda} \geq 0.$$

Optimization

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\lambda}, \mathcal{W}) = -\frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \sum_{c=1}^{C} y_c \log \hat{y}_c, \qquad \boldsymbol{\lambda}^*, \mathcal{W}^* = \underset{\mathcal{W},\ \boldsymbol{\lambda} \in \Delta_M}{\arg\min}\ \mathcal{L}(\boldsymbol{\lambda}, \mathcal{W}).$$

PXGL-GNN keeps the pattern-weight explanation while incorporating node features and implicit GNN representations.

PXGL evidence: pattern weights and embeddings

Learned supervised PXGL-GNN pattern weights λ

Pattern	MUTAG	PROTEINS	DD	NCI1
paths	0.095 ± 0.014	**0.550** ± 0.070	0.093 ± 0.012	0.022 ± 0.002
trees	0.046 ± 0.005	0.074 ± 0.009	0.054 ± 0.006	0.063 ± 0.008
graphlets	0.062 ± 0.008	0.081 ± 0.011	0.125 ± 0.015	0.101 ± 0.013
cycles	**0.654** ± 0.085	0.099 ± 0.013	0.094 ± 0.012	0.176 ± 0.022
cliques	0.082 ± 0.011	0.098 ± 0.012	**0.572** ± 0.073	**0.574** ± 0.075
wheels	0.026 ± 0.003	0.039 ± 0.005	0.051 ± 0.007	0.012 ± 0.002
stars	0.035 ± 0.005	0.056 ± 0.007	0.011 ± 0.002	0.052 ± 0.007

Interpretation

- MUTAG: cycles dominate, consistent with ring structures.
- PROTEINS: path-like structures receive the largest weight.
- DD/NCI1: clique-like structures dominate the representation.

Empirical message: PXGL-GNN improves supervised classification and unsupervised clustering across eight benchmarks while exposing which pattern families drive $\mathbf{g}_G$.

λ

$\sum_m \lambda_m \mathbf{z}^{(m)}$

path, $\lambda_1 = 0.5504$

tree, $\lambda_2 = 0.0746$

graphlet, $\lambda_3 = 0.08103$

PXGL-GNN pattern importance analysis and PROTEINS t-SNE visualizations from the IJCAI 2025 PXGL slides/poster.

Graph foundation models: from graph corpora to transferable graph intelligence

Working definition

A graph foundation model is a broadly pretrained graph model intended to produce transferable representations or task behavior across domains, tasks, and possibly modalities.

Large graph corpora
$\rightarrow$ Unified graph interface
$\rightarrow$ Pretrained backbone
$\rightarrow$ Prompt / fine-tuning / in-context adaptation
$\rightarrow$ Node / edge / graph / reasoning tasks

Conceptual overview: Liu et al., TPAMI 2025 survey; recent surveys in 2025.

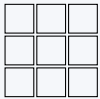
Why GFMs are harder: graphs lack a universal interface

Language and vision have natural interfaces

Text w_1 w_2 w_3 w_4

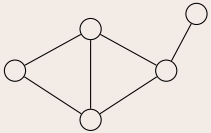
ordered token sequence

Image



regular patch grid

But graphs are different



unordered, relational,
domain-dependent structure

Five missing standards for graph foundation models

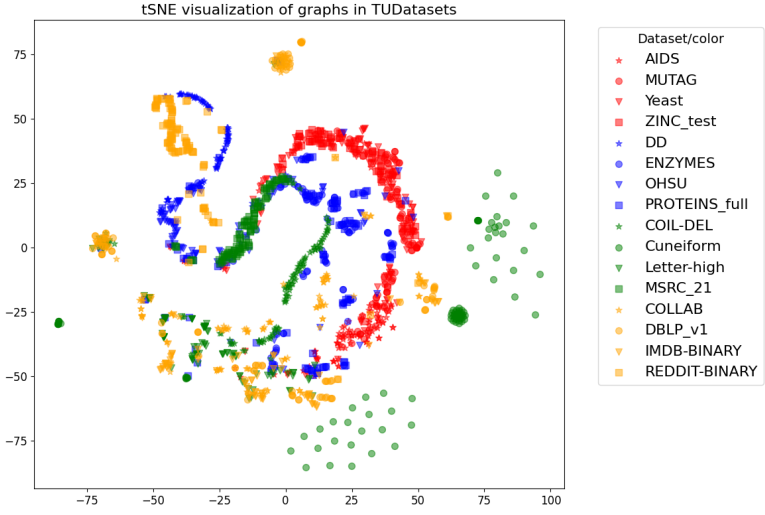
- **Tokenization:** nodes, edges, subgraphs, walks, or graph-text pairs?
- **Task interface:** node, edge, graph, retrieval, generation, or reasoning?
- **Domain semantics:** molecules, social graphs, knowledge graphs, and circuits have different meanings.
- **Position / identity:** graph symmetry and isomorphism make absolute positions ambiguous.
- **Evaluation:** zero-shot, few-shot, transfer, and cross-domain tests are not yet standardized.

Key message

Unlike text and images, graphs do not come with a canonical sequence, grid, task format, or semantic space. This makes universal graph pretraining substantially harder.

Domain heterogeneity: graphs from different worlds (TUDatasets)

small molecules, bioinformatics, computer vision, social networks: MFD+tSNE [Fan 2025]



Key message: A GFM must align structurally and semantically different graph distributions.

GFM vs. LLM: similar ambition, different data structure

	Large language models	Graph foundation models
Basic unit	tokens in a sequence	nodes, edges, subgraphs, walks, or whole graphs
Input structure	ordered text sequence	unordered, relational, and permutation-sensitive graph structure
Pretraining signal	next-token prediction, masked language modeling, instruction tuning	masked graph modeling, contrastive learning, generative graph modeling, cross-task prompting
Transfer target	QA, generation, reasoning, coding, dialogue	node, edge, graph, retrieval, generation, and reasoning tasks
Main difficulty	semantic knowledge and long-context reasoning	structural invariance, graph heterogeneity, task heterogeneity, and domain transfer

Key message

GFM is not simply "LLM for graphs." LLMs can provide semantic knowledge and instruction interfaces, but graph-native structural modeling is still essential.

Graph foundation models by the role of LLMs

Graph-native GFMs

Graph corpus
↓
GNN / Graph Transformer
↓
Transferable graph representation

- no LLM is required
- learn mainly from graph structure and attributes
- focus on graph representation transfer

Examples: GraphMAE, random-walk pretraining, PRODIGY, BRIDGE, RiemannGFM, GraphVec

Graph foundation models by the role of LLMs

Graph-native GFMs

Graph corpus

↓

GNN / Graph Transformer

↓

Transferable graph representation

- no LLM is required
- learn mainly from graph structure and attributes
- focus on graph representation transfer

Examples: GraphMAE, random-walk pretraining, PRODIGY, BRIDGE, RiemannGFM, GraphVec

LLM-assisted GFMs

Graph + text attributes

↓

LM/LLM encoder + graph encoder

↓

Unified graph task prediction

- LLM provides semantic features or prompts
- graph model still captures structure
- useful for cross-domain alignment

Examples: OFA, text-attributed graph models

Graph foundation models by the role of LLMs

Graph-native GFMs

Graph corpus
↓
GNN / Graph Transformer
↓
Transferable graph representation

- no LLM is required
- learn mainly from graph structure and attributes
- focus on graph representation transfer

Examples: GraphMAE, random-walk pretraining, PRODIGY, BRIDGE, RiemannGFM, GraphVec

LLM-assisted GFMs

Graph + text attributes
↓
LM/LLM encoder + graph encoder
↓
Unified graph task prediction

- LLM provides semantic features or prompts
- graph model still captures structure
- useful for cross-domain alignment

Examples: OFA, text-attributed graph models

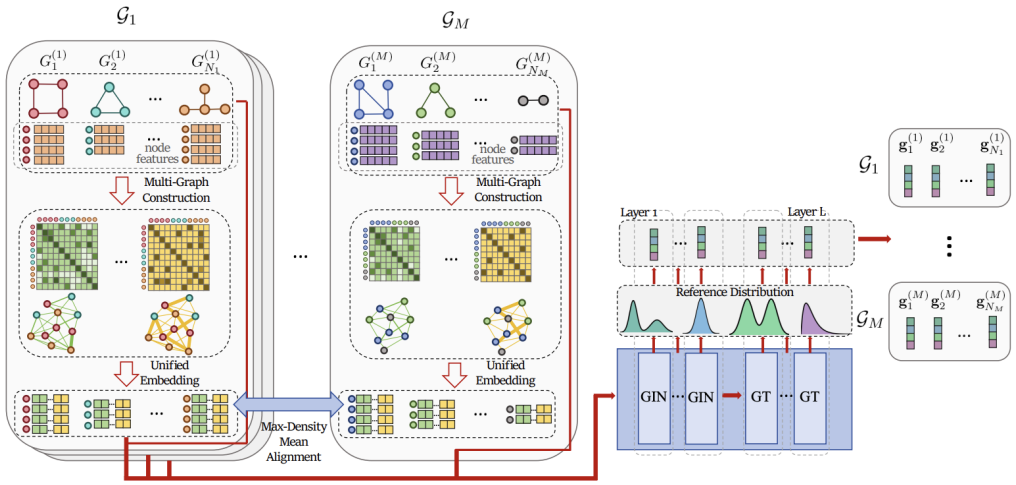
LLM-centric GFMs

Graph + text + question
↓
Graph-language model
↓
Answer / label / explanation

- LLM is the main backbone or interface
- graph tasks become generation or instruction following
- useful for graph QA, explanation, and reasoning

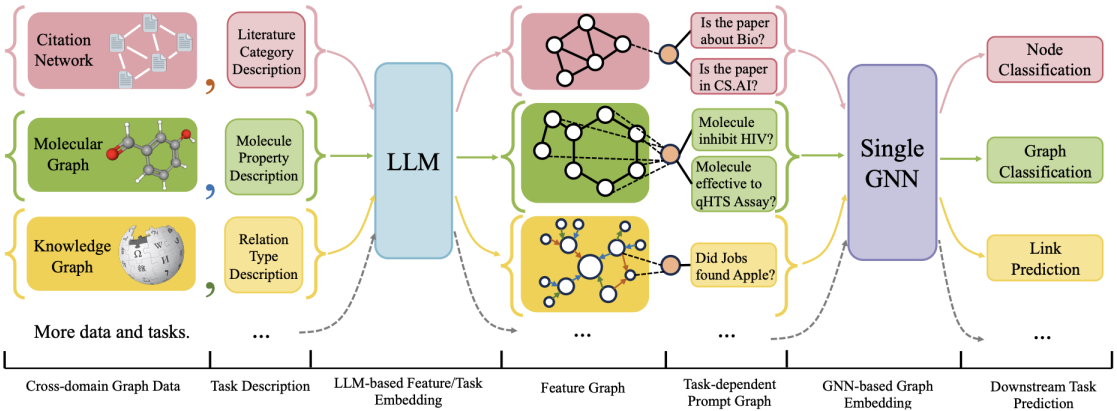
Examples: GraphLLM, GraphGPT, HiGPT, GOFA

An example of Graph-native GFMs



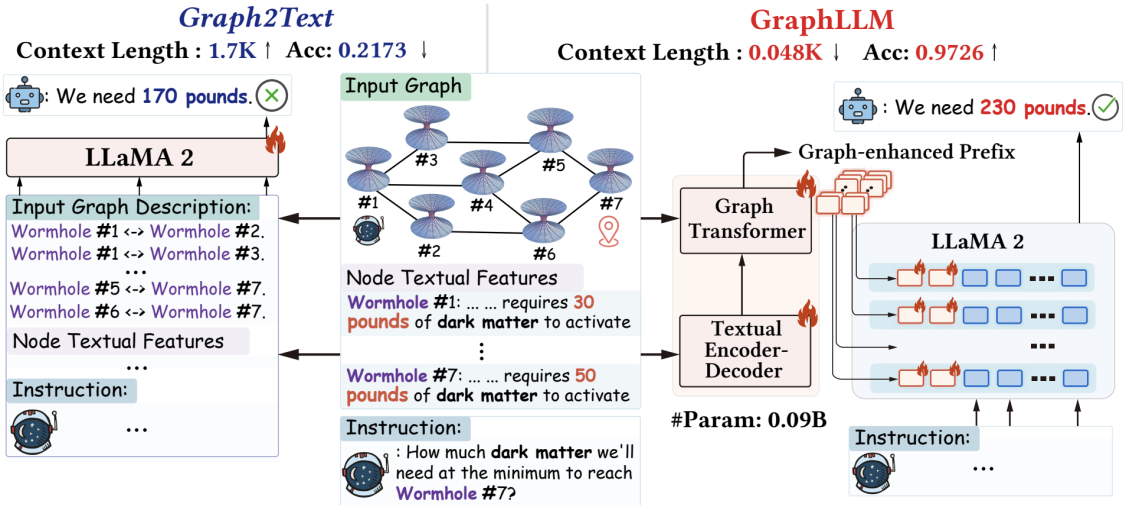
Feng & Fan. GraphVec: Cross-Domain Graph Vectorization for Graph-Level Representation Learning. arXiv 2026.

An example of LLM-assisted GFMs



Liu et al. One for All: Towards Training One Graph Model for All Classification Tasks. ICLR 2024.

An example of LLM-centric GFMs



Chai et al. GraphLLM: Boosting Graph Reasoning Ability of Large Language Model. arXiv 2023.

Graphs meet LLMs: what is genuinely promising?

Promising

- graph-text alignment,
- reasoning over graph queries,
- multimodal graph understanding,
- using language to specify tasks and constraints.

Graphs meet LLMs: what is genuinely promising?

Promising

- graph-text alignment,
- reasoning over graph queries,
- multimodal graph understanding,
- using language to specify tasks and constraints.

Still unclear

- whether language-token pretraining alone is enough for structural generalization,
- how to preserve permutation-aware graph structure inside sequence models,
- whether universal graph prompting is achievable across domains.

Key message

Regardless of whether LLMs are used, the core of a graph foundation model is **transferable graph representation** across domains, tasks, and modalities.

1. Introduction
2. Graph Comparison
3. Graph Representation
4. Explainability and Generalizability
5. Discussion
   - Where is graph learning going?

Where is graph learning going?

From comparison	To representation	Toward generalization
Graph edit distance Graph kernels Optimal transport Factorization distance	GNNs and graph transformers Graph theory Self-supervised representation learning	Prompting Pretraining Graph-language models Graph foundation models
• explicit similarity • interpretable but often costly	• learned latent geometry • scalable but less explicit	• transfer across tasks/domains • still far from universal

Discussion point

The central question is shifting from **"How should we compare graphs?"** to **"How can models learn transferable graph geometry?"**

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SCHOOL OF
DATA SCIENCE
數據科學學院

Thank you!

Jicong Fan, Xudong Wang, Qi Feng, Chris Ding
School of Data Science, The Chinese University of Hong Kong, Shenzhen

Tutorial 3 @ Tai Po I Room 3
Time: 09:00-12:30, June 9 (Tue)
PAKDD 2026, Hong Kong

References I

C. Agarwal, O. Queen, H. Lakkaraju, and M. Zitnik.
Evaluating explainability for graph neural networks.
Scientific Data, 10(1):144, 2023.

U. Alon and E. Yahav.
On the bottleneck of graph neural networks and its practical implications.
In *International Conference on Learning Representations*, 2021.

K. Amara, Z. Ying, Z. Zhang, Z. Han, Y. Zhao, Y. Shan, U. Brandes, S. Schemm, and C. Zhang.
Graphframex: Towards systematic evaluation of explainability methods for graph neural networks.
In *Learning on Graphs Conference*, pages 44:1–44:23, 2022.

M. Belkin and P. Niyogi.
Laplacian eigenmaps for dimensionality reduction and data representation.
Neural Computation, 15(6):1373–1396, 2003.

B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, and H. Maron.
Equivariant subgraph aggregation networks.
In *International Conference on Learning Representations*, 2022.

K. M. Borgwardt and H.-P. Kriegel.
Shortest-path kernels on graphs.
In *IEEE International Conference on Data Mining*, pages 74–81, 2005.

J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun.
Spectral networks and locally connected networks on graphs.
In *International Conference on Learning Representations*, 2014.

References II

M. Cuturi.
Sinkhorn distances: Lightspeed computation of optimal transport.
In *Advances in Neural Information Processing Systems*, 2013.

M. Defferrard, X. Bresson, and P. Vandergheynst.
Convolutional neural networks on graphs with fast localized spectral filtering.
In *Advances in Neural Information Processing Systems*, 2016.

V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson.
Graph neural networks with learnable structural and positional representations.
In *International Conference on Learning Representations*, 2022.

J. Fan.
Graph minimum factor distance and its application to large-scale graph data clustering.
In *International Conference on Machine Learning*, 2025.

T. Gärtner, P. Flach, and S. Wrobel.
On graph kernels: Hardness results and efficient alternatives.
In *Learning Theory and Kernel Machines*, pages 129–143, 2003.

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl.
Neural message passing for quantum chemistry.
In *International Conference on Machine Learning*, pages 1263–1272, 2017.

A. Grover and J. Leskovec.
node2vec: Scalable feature learning for networks.
In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

References III

W. L. Hamilton, R. Ying, and J. Leskovec.
Inductive representation learning on large graphs.
In *Advances in Neural Information Processing Systems*, 2017.

Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang.
Graphmae: Self-supervised masked graph autoencoders.
In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.

H. Kashima, K. Tsuda, and A. Inokuchi.
Marginalized kernels between labeled graphs.
In *International Conference on Machine Learning*, pages 321–328, 2003.

T. N. Kipf and M. Welling.
Semi-supervised classification with graph convolutional networks.
In *International Conference on Learning Representations*, 2017.

L. Kong, J. Feng, H. Liu, C. Huang, J. Huang, Y. Chen, and M. Zhang.
Gofa: A generative one-for-all model for joint graph language modeling.
In *International Conference on Learning Representations*, 2025.

J. Körner.
Coding of an information source having ambiguous alphabet and the entropy of graphs.
Transactions of the Sixth Prague Conference on Information Theory, Statistical Decision Functions, Random Processes, pages 411–425, 1973.

D. Lim, J. Robinson, L. Zhao, T. Smidt, S. Sra, H. Maron, and S. Jegelka.
Sign and basis invariant networks for spectral graph representation learning.
In *International Conference on Learning Representations*, 2023.

References IV

H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, and M. Zhang.
One for all: Towards training one graph model for all classification tasks.
In *International Conference on Learning Representations*, 2024.

J. Liu, C. Yang, Z. Lu, J. Chen, Y. Li, M. Zhang, T. Bai, Y. Fang, L. Sun, P. S. Yu, and C. Shi.
Graph foundation models: Concepts, opportunities and challenges.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 47(6):5023–5044, 2025.

S. Liu, X. Li, D. Su, R. Zhang, H. Qin, R. Li, and G. Wang.
Toward effective multimodal graph foundation model: A divide-and-conquer based approach.
arXiv preprint arXiv:2602.04116, 2026.

L. Lovász.
On the shannon capacity of a graph.
IEEE Transactions on Information Theory, 25(1):1–7, 1979.

D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang.
Parameterized explainer for graph neural network.
In *Advances in Neural Information Processing Systems*, 2020.

H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman.
Provably powerful graph networks.
In *Advances in Neural Information Processing Systems*, 2019.

F. Mémoli.
Gromov-wasserstein distances and the metric approach to object matching.
Foundations of Computational Mathematics, 11(4):417–487, 2011.

References V

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe.
Weisfeiler and leman go neural: Higher-order graph neural networks.
In *AAAI Conference on Artificial Intelligence*, pages 4602–4609, 2019.

A. Y. Ng, M. I. Jordan, and Y. Weiss.
On spectral clustering: Analysis and an algorithm.
In *Advances in Neural Information Processing Systems*, 2002.

B. Perozzi, R. Al-Rfou, and S. Skiena.
Deepwalk: Online learning of social representations.
In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

G. Peyré and M. Cuturi.
Computational optimal transport, volume 11 of *Foundations and Trends in Machine Learning*.
Now Publishers, 2019.

L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini.
Recipe for a general, powerful, scalable graph transformer.
In *Advances in Neural Information Processing Systems*, 2022.

K. Riesen and H. Bunke.
Approximate graph edit distance computation by means of bipartite graph matching.
Image and Vision Computing, 27(7):950–959, 2009.

T. K. Rusch, M. M. Bronstein, and S. Mishra.
A survey on oversmoothing in graph neural networks.
arXiv preprint arXiv:2303.10993, 2023.

References VI

A. Sanfeliu and K.-S. Fu.
A distance measure between attributed relational graphs for pattern recognition.
IEEE Transactions on Systems, Man, and Cybernetics, 13(3):353–362, 1983.

V. G. Satorras, E. Hoogeboom, and M. Welling.
E(n) equivariant graph neural networks.
In *International Conference on Machine Learning*, pages 9323–9332, 2021.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini.
The graph neural network model.
IEEE Transactions on Neural Networks, 20(1):61–80, 2009.

A. Shehzad, F. Xia, S. Abid, C. Peng, S. Yu, D. Zhang, and K. Verspoor.
Graph transformers: A survey.
arXiv preprint arXiv:2407.09777, 2024.

N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt.
Weisfeiler-lehman graph kernels.
Journal of Machine Learning Research, 12:2539–2561, 2011.

J. Shi and J. Malik.
Normalized cuts and image segmentation.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888–905, 2000.

F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang.
Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization.
In *International Conference on Learning Representations*, 2020.

References VII


Y. Sun and J. Fan.
Mmd graph kernel: Effective metric learning for graphs via maximum mean discrepancy.
In *International Conference on Learning Representations*, 2024.


Z. Sun, C. Ding, and J. Fan.
Lovász principle for unsupervised graph representation learning.
In *Advances in Neural Information Processing Systems*, 2023.


Z. Sun, X. Wang, C. Ding, and J. Fan.
Learning graph representation via graph entropy maximization.
In *International Conference on Machine Learning*, 2024.


S. Suresh, P. Li, C. Hao, and J. Neville.
Adversarial graph augmentation to improve graph contrastive learning.
In *Advances in Neural Information Processing Systems*, 2021.


Z. Tang and J. Chen.
Toward a graph foundation model: Pre-training transformers with random walks.
arXiv preprint arXiv:2506.14098, 2025.


M. Togninalli, E. Ghisu, F. Llinares-López, B. Rieck, and K. Borgwardt.
Wasserstein weisfeiler-lehman graph kernels.
In *Advances in Neural Information Processing Systems*, 2019.


J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein.
Understanding over-squashing and bottlenecks on graphs via curvature.
In *International Conference on Learning Representations*, 2022.

References VIII


T. Vayer, L. Chapel, R. Flamary, R. Tavenard, and N. Courty.
Optimal transport for structured data with application on graphs.
In *International Conference on Machine Learning*, pages 6275–6284, 2019.


T. Vayer, L. Chapel, R. Flamary, R. Tavenard, and N. Courty.
Fused gromov-wasserstein distance for structured objects.
Algorithms, 13(9):212, 2020.


P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio.
Graph attention networks.
In *International Conference on Learning Representations*, 2018.


P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm.
Deep graph infomax.
In *International Conference on Learning Representations*, 2019.


S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt.
Graph kernels.
Journal of Machine Learning Research, 11:1201–1242, 2010.


U. von Luxburg.
A tutorial on spectral clustering.
Statistics and Computing, 17(4):395–416, 2007.


X. Wang, C. Ding, T. Li, and J. Fan.
Adaptive riemannian graph neural networks.
arXiv preprint arXiv:2508.02600, 2025.

References IX


X. Wang, Z. Sun, C. Ding, and J. Fan.
Explainable graph representation learning via graph pattern analysis.
In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 3426–3434, 2025.


K. Xu, W. Hu, J. Leskovec, and S. Jegelka.
How powerful are graph neural networks?
In *International Conference on Learning Representations*, 2019.


P. Yanardag and S. V. N. Vishwanathan.
Deep graph kernels.
In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.


C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu.
Do transformers really perform badly for graph representation?
In *Advances in Neural Information Processing Systems*, 2021.


R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec.
Gnnexplainer: Generating explanations for graph neural networks.
In *Advances in Neural Information Processing Systems*, 2019.


Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen.
Graph contrastive learning with augmentations.
In *Advances in Neural Information Processing Systems*, 2020.


C. Yuan, K. Zhao, E. E. Kuruoglu, L. Wang, T. Xu, W. Huang, D. Zhao, H. Cheng, and Y. Rong.
A survey of graph transformers: Architectures, theories and applications.
arXiv preprint arXiv:2502.16533, 2025.

References X

H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji.
On explainability of graph neural networks via subgraph explorations.
In *International Conference on Machine Learning*, pages 12241–12252, 2021.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola.
Deep sets.
In *Advances in Neural Information Processing Systems*, 2017.