# UC Santa Barbara

**UC Santa Barbara Electronic Theses and Dissertations** 

## Title

Transparent Representation Learning for Graphs and Human-AI Collaboration

Permalink

https://escholarship.org/uc/item/5vw9j8pz

**Author** Kosan, Mert

Publication Date 2023

Peer reviewed|Thesis/dissertation

University of California Santa Barbara

# Transparent Representation Learning for Graphs and Human-AI Collaboration

A dissertation submitted in partial satisfaction of the requirements for the degree

Doctor of Philosophy in Computer Science

by

### Mert Kosan

Committee in charge:

Professor Ambuj Singh, Chair Professor Francesco Bullo Professor Xifeng Yan

June 2023

The Dissertation of Mert Kosan is approved.

Professor Francesco Bullo

Professor Xifeng Yan

Professor Ambuj Singh, Committee Chair

June 2023

Transparent Representation Learning for Graphs and Human-AI Collaboration

Copyright  $\bigodot$  2023

by

Mert Kosan

To my family for their unconditional love and support.

#### Acknowledgements

Throughout my Ph.D. journey at UCSB, I have experienced a wide range of emotions, from excitement to happiness. I am deeply grateful to all those who have provided me with support, motivation, and guidance along the way.

I would like to express my sincere gratitude to my Ph.D. advisor, Prof. Ambuj Singh, who provided me with exceptional mentorship. His vision and wisdom have enabled me to become the person I always aspired to be. I would also like to extend my appreciation to Prof. Francesco Bullo and Prof. Xifeng Yan for their expertise and invaluable mentorship as members of my Ph.D. committee.

I consider myself fortunate to have had the opportunity to collaborate with some of the most brilliant researchers in the world: Zexi, Sourav, Arlei, Sayan, Marianne, Kha-Dinh, Saurabh, and Rasta. I am also grateful for having a great group of labmates: Rachel, Richika, Chandana, Ashwini, Haraldur, Wei, Omid, Hongyuan, Furkan, Yuning, Sikun, Christos, Danish, Sean, and other collaborators from other labs or universities: Abed, Anand, Kittiphat, Yibei, Danqing, Antonis, and Aritra.

Special thanks to Zexi Huang, Sourav Medya, and Arlei Silva for being my most influential mentors and closest friends during my Ph.D. study. I find it difficult to envision what would have occurred in their absence.

I consider myself very lucky to have received the support and companionship of my close friends in the US and from Turkey. I witnessed Lionel Messi winning the World Cup 2022 with them.

Last but not least, I would like to express my heartfelt gratitude to my dear mother Sevgi, father İlhan, brother Faik Emre, sweet nephew Barış, and all remaining family members in Turkey for their unwavering love and support throughout my Ph.D. studies.

### Curriculum Vitæ Mert Kosan

### Education

2023	Ph.D. in Computer Science, University of California, Santa Barbara.
2023	M.S. in Computer Science, University of California, Santa Barbara.
2018	B.S. in Computer Science & Engineering, Sabanci University, Istanbul

### Publications

Peer-reviewed papers:

- Mert Kosan, Zexi Huang, Sourav Medya, Sayan Ranu, Ambuj Singh. Global Counterfactual Explainer for Graph Neural Networks. ACM International Conference on Web Search and Data Mining, WSDM 2023. (Selected among Top 10 in WSDM 2023, Best Paper Award in MLoG-WSDM 2023)
- Mert Kosan, Linyun He, Shubham Agrawal, Hongyi Liu, Chiranjeet Chetia. AI Decision Systems with Feedback Loop Active Learner. WSDM 2023 Crowd Science Workshop on Collaboration of Humans and Learning Algorithms for Data Labeling.
- Mert Kosan, Arlei Silva, Sourav Medya, Brian Uzzi, Ambuj Singh. Event Detection on Dynamic Graphs. Deep Learning on Graphs: Method and Applications, Association for the Advancement of Artificial Intelligence, DLG-AAAI 2023.
- Debajyoti Kar, <u>Mert Kosan</u>, Debmalya Mandal, Sourav Medya, Arlei Silva, Palash Dey, Swagato Sanyal. Feature-based Individual Fairness in k-Clustering. International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023 Extended Abstract)

Working papers:

- Mert Kosan, Arlei Silva, Ambuj Singh. Robust Ante-hoc Graph Explainer with Bilevel Optimization. 2023.
- Exi Huang, Mert Kosan, Arlei Silva, Ambuj Singh. Link Prediction without Graph Neural Networks. 2023.
- ▷ Marianne Arriola, Saurabh Sharma, <u>Mert Kosan</u>, Zexi Huang, Ambuj Singh. Multiscale Anomaly Detection with Graph Autoencoders. 2023.
- Danqing Wang, Antonis Antoniades, Kha-Dinh, <u>Mert Kosan</u>, Ambuj Singh. Domainconstraint Global Counterfactual Explainer. 2023.
- Mert Kosan, Zexi Huang, Francesco Bullo, Noah Friedkin, Ambuj Singh. Robustness of Human Behavior under Risk. 2023.

### **Research Internships**

$\triangleright$	Machine Learning Scientist, Visa Research	Summer 20	22
	Project: Improving AI Decisions with Feedback Loop Active Learner		
	Mentors: Linyun He, Chiranjeet Chetia		
	Outcome: Patent filed, Paper published.		
$\triangleright$	Machine Learning Scientist, Visa Research	Summer 20	21
	Project: Peer Group Analysis with Anomaly Detection		
	Mentors: Shubham Agrawal, Linyun He, Chiranjeet Chetia		
	Outcome: Patent filed.		
$\triangleright$	Machine Learning Scientist, Visa Research	Summer 20	20
	Project: Fraud Detection and Profiling		
	Mentors: Linyun He, Chiranjeet Chetia		
	Outcome: Patent filed.		
Тe	eaching Experience		

#### Teaching Experience

$\triangleright$	Teaching Assistant, Data Structures and Algorithms, UCSB	Fall 2021
$\triangleright$	Instructor, LMU/UCSB Nanotech PhD Exchange and Symposium	Spring 2019
$\triangleright$	Teaching Assistant, Data Structures and Algorithms, UCSB	Winter 2018
$\triangleright$	Teaching Assistant, Data Structures and Algorithms, UCSB	Fall 2018

### **Academic Services**

 $\triangleright$  Registration Chair: KDD'23

▷ Reviewer: NeurIPS, ICLR, KDD, WebConf, WSDM, ICDM, SDM, AAAI, TIST, TKDD, TKDE.

#### Abstract

#### Transparent Representation Learning for Graphs and Human-AI Collaboration

by

#### Mert Kosan

Graph data show relationships between entities in a variety of domains including but not limited to communication, social, and interaction networks. Representation learning makes graph data easier to use for graph tasks such as graph classification, link prediction, and clustering. The decisions on graphs depend on complex patterns combining rich structural and attribute data. Therefore, explaining these decisions made by representation learning models for high-stakes applications (e.g., anomaly detection and drug discovery) is critical for increasing transparency and guiding improvements. Moreover, human expertise can guide machine learning decisions, raising questions about the interactions between humans and artificial intelligence that require further analysis.

This dissertation focuses on our research on two key topics: transparent representation learning on graphs and human-AI collaboration. Firstly, we present our proposed frameworks for graph anomaly detection, which have been developed to enhance accuracy and transparency. Subsequently, we scrutinize explainability in graph machine learning, where we discuss our novel post-hoc global counterfactual and robust ante-hoc graph explainers. Fairness is also a crucial aspect of transparent machine learning, and we propose a new individual fairness method for clustering. Finally, we investigate the impact of collaboration between humans and artificial intelligence on decision-making under risk and feedback loop systems.

#### **Permissions and Attributions**

The content of this dissertation includes the part of the previously published or ongoing research. The author asserts their primary contributions to the development of the research works which have proceedings described below:

- The part of Chapter 3 has been previously published as: Zexi Huang, Mert Kosan, Sourav Medya, Sayan Ranu, Ambuj Singh. Global Counterfactual Explainer for Graph Neural Networks. Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM), 2023. DOI: 10.1145/3539597.3570376.
- The part of Chapter 6 has been previously published as: Debajyoti Kar, Mert Kosan, Debmalya Mandal, Sourav Medya, Arlei Silva, Palash Dey, Swagato Sanyal. Featurebased Individual Fairness in k-Clustering. Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2023. DOI: 10.5555/3545946.3599073.
- ▷ The part of Chapter 8 has been previously published as: Mert Kosan, Linyun He, Shubham Agrawal, Hongyi Liu, and Chiranjeet Chetia. AI Decision Systems with Feedback Loop Active Learner. Proceedings of the 4th Crowd Science Workshop on Collaboration of Humans and Learning Algorithms for Data Labeling (Crowd Science WSDM), 2023. CEUR-WS: Vol-3357/2

The ACM Author Rights page (https://authors.acm.org/author-resources/authorrights) states that authors are permitted to include portions or entire papers of their own work in a dissertation without any additional fees, provided that they include citations and DOI links to the Versions of Record in the ACM Digital Library. Additionally, authors may freely use their own work in classroom teaching and presentations, without any fee required. The CUER-WS Author Rights page (https://ceurws.org/HOWTOSUBMIT.html#AUTHORRIGHTS) states the following: "The author has the right to publish a pre-print, post-print or the published version of his CEUR-WS.org paper on his homepage, an institutional repository, or elsewhere".

# List of Figures

2.1	(a) Event detection on dynamic graphs based on a generic deep learning	
	architecture. (b) At the micro scale, the dynamics is captured at node	
	level using a temporal GNN architecture and then pooled for graph-level	
	classification. (c) At the macro scale, the dynamics is captured at the	
	graph level using an RNN over pooled (static) GNN node embeddings.	
	Our work investigates how the dynamics at different scales affects event	
	detection performance	11
	(a) Conoric architecture for event detection	11
	(b) Learning graph embeddings based on micro dynamics	11
	(b) Learning graph embeddings based on means dynamics	11
იი	(c) Learning graph embeddings based on macro dynamics $\ldots$	11
2.2	Scaled prediction scores $([0, 1])$ given by some of the event detection meth-	
	ods for Hedge Fund. Events are marked as vertical (red) lines. The first	
	(grey) half covers some of the training events, while the remaining (white)	
	is in the test set. Some events are annotated for illustration. Hedge Fund	
	annotations are headlines from CNBC on the day of the event. Our method	
	DyGED can identify several of the events while keeping the false positive	
	rate lower.	25
2.3	Top 10 taxi zones (in red) based on various metrics for NYC Cab dataset.	
	Attention mechanism finds closer taxi zones to the stadiums compared to	
	some node statistics such as betweenness centrality, clustering coefficient.	
	and node degree. In other words, DyGED detects more crucial nodes for	
	Baseball game detection	$\overline{27}$
	(a) DyCED Attentions	21
	(a) DyGLD Attentions	21
	$(D)  \text{Detweenliess} \dots $	21
	(c) $\bigcup$ Unsterning $\ldots$	27
	(d) Degree	27

2.4	Illustration of the attention weights (normalized) of current and previous three snapshots for all datasets. While in NYC Cab, the current snapshot has more attention weights than the previous ones, attention weights in the Hedge Fund and Twitter datasets reveal more about the importance of temporal information. Results show that history plays a role in predicting the event. 0 <sup>th</sup> refers to the current snapshot
2.5	(c) Twitter Weather
	(a)       NYC Cab       2         (b)       Hedge Fund       2         (c)       Twitter Weather       2
2.6	Spectral energy distributions of normal nodes, node-level anomalies, and three subgraph-level anomalies at different scales for Cora with injected anomalies. The spectral energy distributions of anomalous nodes concen- trate more on the high frequency regions
2.7	Spectral properties of heat kernels and Beta kernels are compared, with the latter incorporating diverse band-pass filters to enable the detection of anomalies at multiple scales
3.1	Formaldehyde (a) is classified by a GNN to be an undesired mutagenic molecule with its important subgraph found by factual reasoning high- lighted in red. Formic acid (b) is its non-mutagenic counterfactual exam- ple obtained by removing one edge and adding one node and two edges.
3.2 3.3	(a) Formaldehyde3(b) Formic acid3Edits between graphs.3Edits between graphs.4Coverage and cost performance comparison between GCFEXPLAINERand baselines based on different counterfactual summary sizes.GCF-EXPLAINER consistently outperforms the baselines across different sizes.5

3.4	Recourse coverage comparison between GCFEXPLAINER and baselines based on different distance threshold values ( $\theta$ ). GCFEXPLAINER consis-	
3.5 3.6	tently outperforms the baselines across different $\theta$	55
	50000 iterations	58
4.1	Explanations generated by our approach (RAGE) in two case studies: PLANTED CLIQUE (graphs with and without cliques) and SUNGLASSES (headshots with and without sunglasses). RAGE explanations identify edges in the clique and in the region around the sunglasses for both diffi- culties. For PLANTED CLIQUE (a), the heatmap and sizes show node and edge influences, and the nodes with a thick border are members of the planted clique. For SUNGLASSES (b-c), the red dots show the influential	
	pixel connections to the detection.	65
	(a) PLANTED CLIQUE	65
	(b) SUNGLASSES - EASY	65 65
4.2	<ul> <li>(a) Post-hoc models generate explanations for a pre-trained GNN classifier using its predictions.</li> <li>(b) Ante-hoc models, as our approach, learn GNNs and explanations jointly. This enables ante-hoc models to identify GNNs that are both explainable and accurate.</li> <li>(a) Post-hoc Models</li> <li>(b) Ante-hoc Models</li> </ul>	68 68 68
4.3	Illustration of an edge-based ante-hoc explainer that uses bilevel optimiza- tion. Explainer generates an explanation graph from the input graph by assigning an influence value to each edge. Edge influences are incorporated to edge weights on the explanation graph, the input of GNN Classifier. The inner problem optimizes GNN Classifier with $T$ iterations, while the outer problem updates Explainer using gradients from inner iterations. The dot- ted edges in the explanation graph show that they do not influence the classification, while others have different degraph of influence.	71
4.4	Reproducibility comparison between methods for different explanation	(1
	sizes (in percentage) using four datasets. RAGE outperforms or has com-	
	parable results to the baselines across different sizes and datasets	80

4.5	Pearson correlation (higher is better) between explanations for Muta- genicity and MutagenicityNoisyX generated by RAGE and the baselines.	
	RAGE is significantly more stable than the baselines, with GIB as the best	
1.0		81
4.6	Two examples with different levels of difficulty due to poses from the Sun-	
	glasses dataset. RAGE is able to detect the edges between the sunglasses	
	and the numan faces, which are the most relevant for the prediction task	00
	(1.e., whether the human is wearing sunglasses) for both examples	82
	(a) ORIGINAL IMAGE $\dots$	82
	(b) $PGEXPLAINER$	82
	(c) $GIB \dots$	82
	$(d) RAGE \dots	82
	(e) ORIGINAL IMAGE	82
	(f) $PGEXPLAINER \dots$	82
	(g) $GIB$	82
	(h) RAGE	82
4.7	Training and validation loss over iterations of bilevel optimization training	
	of Mutagenicity dataset. Even though our training process is not unstable,	~
	there is still room for improvement	85
5.1	GNN incorporates topology into attributes via message-passing which is	
0.1	effective for tasks <b>on</b> the topology Link prediction however is a task <b>for</b>	
	the topology which motivates the design of Gelato—a novel framework	
	that leverages graph learning to incorporate attributes into topology	88
	(a) Link prediction for	00
	attributed graphs	88
	(b) GNN:	00
	$topology \rightarrow attributes$	88
	(c) Gelato:	00
	(c) details.	88
52	Gelate applies graph learning to incorporate attribute information into	00
0.2	the topology via an MLP. The learned graph is given to a topological	
	heuristic that predicts edges between node pairs with high Autocovariance	
	similarity. The parameters of the MLP are optimized end-to-end using the	
	N-pair loss Experiments show that Celato outperforms state-of-the-art	
	GNN-based link prediction methods	92
53	Link prediction performance in terms of $mec@k$ for varying values of $k$ (as	52
0.0	percentages of test edges) With few exceptions Celato outperforms the	
	baselines across different values of $k$	101
5 /	Link prediction performance in terms of $hits@k$ for varying values of k	101
0.4	With faw executions Calato outperforms the baselines across different	
	when it we exceptions, detail outperforms the baselines across different values of $k$	109
	values of $\kappa$ .	102

5.5	Illustration of the link prediction process of Gelato, AC, and the best	
	GNN-based approach, Neo-GNN, on a subgraph of PHOTO. Gelato effec-	
	tively incorporates node attributes into the graph structure and leverages	
	topological heuristics to enable state-of-the-art link prediction.	104
	(a) Input adjacency matrix	104
	(b) Enhanced adjacency matrix	104
	(c) Attribute Euclidean distance	104
	(d) AC scores	104
	(e) Gelato scores	104
	(f) Neo-GNN predictions	104
	(g) AC predicted edges $\ldots \ldots	104
	(h) Gelato predicted edges	104
	(i) Neo-GNN predicted edges	104
5.6	Training of Gelato based on different losses and training settings for	
	Photo with test AP (mean $\pm$ std) shown in the titles. Compared with	
	the cross entropy loss, the N-pair loss with <i>unbiased training</i> is a more	
	consistent proxy for <i>unbiased testing</i> metrics and leads to better peak per-	
	formance.	105
5.7	Performance of Gelato with different values of $\eta$	107
	(a) PHOTO performance	107
	(b) CORA performance	107
5.8	Performance of Gelato with different $\alpha$ and $\beta$	108
	(a) PHOTO AP scores	108
	(b) Photo $prec@100\%$ scores	108
	(c) CORA AP scores	108
	(d) CORA $prec@100\%$ scores	108
5.9	Receiver operating characteristic and precision-recall curves for the bad	
	link prediction model that ranks 1M false positives higher than the 100k	
	true edges. The model achieves 0.99 in AUC and 0.95 AP under biased	
	testing, while the more informative performance evaluation metric, Aver-	
	age Precision (AP) under <i>unbiased testing</i> , is only 0.05.	110
	(a) $\operatorname{ROC}$	110
	(b) PR under biased testing	110
- 10	(c) PR under <i>unbiased testing</i>	110
5.10	Link prediction performance in terms of $prec@k$ (in percentage) for vary-	
	ing values of $k$ with baselines using <i>unbiased training</i> . While we observe	
	noticeable improvement for some baselines (e.g., BScNets), Gelato still	110
	consistently and significantly outperform the baselines	113
5.11	Link prediction performance in terms of $hits@k$ (in percentage) for vary-	
	ing values of $k$ with baselines using <i>unbiased training</i> . While we observe	
	noticeable improvement for some baselines (e.g., BScNets), Gelato still	110
	consistently and significantly outperform the baselines	113

6.1	Cost and fairness results, varying the number of clusters $(k)$ for $30k$ ran- dom data points using all datasets and methods (better seen in color). LP-FAIR outperforms or achieves results comparable to the baselines for all $k$	137
7.1	Overview of our human experiments. The individual phase (IND) in- volves each participant independently responding to a series of $m$ issues. Following this, groups are formed and the group phase begins. In the pre- discussion (PRE) phase, each group member individually answers an issue, engages in group discussions, and subsequently provides their responses again in the post-discussion (POST) phase. This cycle of pre-discussion and post-discussion repeats for a total of $n$ iterations. At the end of each cycle, each member indicates the perceived influence from other group members.	146
7.2	An issue example. Each issue contains a choice dilemma of a binary gamble selection between option $A$ and $B$ . Each option includes gain and loss with corresponding probabilities. Individuals record their answers by selecting either option	140
7.3	Stubbornness level of individuals over group issues. UCSB participants	141
	are more stubborn than FB participants.	156
7.4	Contributions of different features in making predictions in POST choice for UCSB dataset.	158
7.5	Confusion matrix showing the shift between low and high-risk choices based on different models. Significant changes occur from IND to POST. The predictions are based on the PT models.	159
	(a) IND $\rightarrow$ PRE	159 159
76	(c) $PRE \rightarrow POST$	159
7.0	consensus	$\begin{array}{c} 160 \\ 160 \end{array}$
	(b) Rolling Average	160
7.7	Average pairwise distance between individuals in each group for IND, PRE and POST parameters. For most groups, pairwise IND distances > pair- wise PRE distances > pairwise POST distance. This shows that the be- havior of individuals shifts towards consensus in a group setting	161
7.8	Importance of features in predicting consensus. Explicit and implicit low- risk influence are effective features	163
7.9	Importance of features for influence predictions <i>lambda</i> parameters are	100
	the most effective parameter to decide influence.	165

7.10	Robustness of groups based on cosine distance between group behavior with AI agent and group behavior without AI agent. Missing data shows there are no agents found using grid search to reach a consensus on the same issues. Group 5 is the least robust and groups 10, 15, 16, 21, 22, 23,	
7.11	and 24 are the most robust	166 167
7.12	VAE Diagram for Generative Model.	169
7.13	Comparison of two agent selection methods in finding optimal AI agent for attacking the groups. Notably, the grid search method identifies an AI	
7.14	agent with a more impactful attack for groups	170 171
8.1	A short illustration of Feedback Loop Active Learner. It starts with mul- tiple entities at which a black box AI system generates decisions. FLAL uses these decisions and entities to send queries to a human, who eval- uates them and generates ground truth labels and feedback. FLAL uses this feedback (including human interest and expertise) in active learning training and stores the ground truth labels for future updates in the AI	
8.2	decision system	176
	detectors.	180

- 8.3 An illustration of Stream Data Embedder. The pre-trained unsupervised embedder takes input from all entities' time series data (up to  $\tau$  timestamp history) and embeds them into d dimensional space. This allows a better and more compact representation of time-series data. . . . . . . . . . . .
- 8.4 The usefulness example of user embedding mapper. Since the global embedding space will group related entities together and a specific user may be interested in different types of entities, the user embedding mapper will learn how to regroup entities. This example shows that the algorithm may detect the anomalies for different reasons: data issues, seasonal change, and authentication-related. Therefore, if only data issues and authentication-related anomalies (red entities) are relevant to the user, the algorithm groups them together to make the embedding space personalized.182

182

- 8.7 Ablation study of user embedding mapper. The linear and non-linear layers have competing performances at both metrics. The linear mapping converges slowly, and 2 non-linear layers suffer from the lack of available data points.
  191

# List of Tables

2.1	Time complexities of our methods: $m, n, h, l_1, k, l_2$ denote numbers of edges, nodes, embedding dimension, layers in GCN, past snapshots, layers	
	in MLP. We assume the initial node feature dimension for GCN, input,	
	cell, and output dimension for LSTM equal to embedding dimension $h$ .	20
2.2	The statistics of the datasets. TW is Twitter Weather.	21
2.3	AUC scores of event detection methods. The highest and second highest values for each column are in bold and underlined, respectively. Our methods, accounting for macro dynamics, achieve the best results, outperforming the best baseline (ASTGCN) by 4.5% on average and up to 8.5%	
	(on Twitter Weather). We performed a paired t-test comparing the best model against the others (markers $**$ and $*$ indicate p-value $< 01$ and $<$	
	.05. respectively).	24
2.4	Testing times (in secs.) for all the methods with windows length $k+1$ set	
	to 4. DyGED is scalable and up to 206 times faster than the best baseline,	
	ASTGČN.	31
2.5	AUC scores of DyGED and the best baseline method ASTGCN with different sets of node featuresS, -D, -SD suffixes are for static-only,	
	dynamic-only, static+dynamic node features, respectively. TW is the Twitter Weather dataset.	31
2.6	AUC scores of our event detection method DyGED equipped with different pooling operators. Attention pooling outperforms mean and max pooling for all datasets using our method DyGED. TW is the Twitter Weather	
	data	32
3.1	The statistics of the datasets.	52
3.2	Accuracy of the GNN graph classifier.	53
3.3	Recourse coverage ( $\theta = 0.1$ ) and median recourse cost comparison between GCFEXPLAINER and baselines for a 10-graph global explanation. GCF-EXPLAINER consistently and significantly outperforms all baselines across	
	different datasets.	54
3.4	Ablation study results based on recourse coverage	58

3.5	Sensitivity analysis on $\alpha$ , the weight between individual coverage and gain of coverage in the importance function.	59
3.6	Counterfactual candidates generation time comparison. GCFEXPLAINER	00
	(-S) has competitive running time albeit exploring more counterfactual	
	graphs.	59
3.7	Fragment-based editing enhances the search efficiency and promotes valid	
	molecule generation. Despite slightly lower coverage and cost, the validity	
	utilization. Time is calculated based on minutes	63
		00
4.1	The statistics of the datasets.	77
4.2	Test scores for graph classification/regression. Best and second-best values	
	results on average outperforming the baselines	79
4.3	Test scores (AUC and MSE) of RAGE and its single-level variant RAGE-	
	single for all datasets. RAGE outperforms RAGE-single in all datasets.	
	Nevertheless, RAGE-single still has consistent performance across datasets.	83
4.4	Test scores (AUC and MSE) of RAGE and its without reinitialization	
	all datasets except PLANTED CLIQUE which has the same performance	
	The usefulness of reinitialization is more obvious in larger datasets	84
51	A summary of dataset statistics	07
$5.1 \\ 5.2$	Reference of baseline code repositories.	98
5.3	Link prediction performance comparison (mean $\pm$ std AP). Gelato consis-	
	tently outperforms GNN-based methods, topological heuristics, and two-	
F 4	stage approaches combining attributes and topology.	101
5.4	Results of the ablation study based on AP scores. Each component of Colate plays an important role in enabling state of the art link prediction	
	performance.	106
5.5	Selected hyperparameters of Gelato.	107
5.6	Training and inference time comparison between supervised link prediction	
	methods for PHOTO. Gelato has competitive training time (even under	
	of inference, especially compared to the best GNN model. Neo-GNN	109
5.7	Link prediction performance comparison (mean $\pm$ std AUC). AUC results	105
	conflict with other evaluation metrics, presenting a misleading view of the	
	model performance for link prediction.	111
5.8	Link prediction performance comparison (mean $\pm$ std AP) with super-	
	vised link prediction methods using <i>unbiased training</i> . While we observe noticeable improvement for some baselines (e.g. BScNets). Colate still	
	consistently and significantly outperform the baselines	112

5.9	Training time comparison between supervised link prediction methods for PHOTO under <i>unbiased training</i> . Gelato, while achieving the best performance, is also the second most efficient method in terms of total training time, slower only than the vanilla MLP	114
6.1	Normalized cost and fairness comparison between LP-FAIR (ours) and competing baselines. The best and second-best values for each column are in bold and underlined, respectively. Our method outperforms or has performance comparable to the baselines in terms of the three evaluation matrice	196
6.2	The mean and standard deviation of the number of clusters generated by the methods (with $k = 5$ ). Generating fewer clusters generally leads to	100
6.3	The mean and standard deviation of cluster imbalance. Imbalanced clus- ters result in small clusters where the members might not be individually fair. LP-FAIR generates clusters with lower imbalance compared to the	139
6.4	baselines	139
6.5	Normalized cost and fairness comparison between LP-FAIR (ours) and competing baselines with random feature selections. The best and second- best values for each column are in bold and underlined, respectively. Our method outperforms or has performance comparable to the baselines in	140
6.6	terms of fairness and cost	141 142
7.1	The statistics of datasets from human experiments	148
<b>T</b> .2	individuals.	154
7.3 7.4	Statistics of influence matrices from human experiments. $\forall$ denotes every issue, and $\exists$ denotes at least one issue	155
	of accuracy	157

# Contents

Curriculum Vitae					
Abstract					
Pe	Permissions and Attribution				
List of Figures					
Lis	of Tables	xix			
1	ntroduction	1			
2	Event Detection on Dynamic Graphs2.1 Introduction2.2 Related Work2.3 Problem Definition2.4 Proposed Model: DyGED2.5 Experiments2.6 Conclusions2.7 Future Works - Multi-scale Anomalies [1]	8 8 12 14 15 20 32 33			
3	Global Counterfactual Explainer for Graph Neural Networks         1 Introduction	<ul> <li>36</li> <li>36</li> <li>39</li> <li>44</li> <li>52</li> <li>60</li> <li>61</li> <li>65</li> </ul>			
-	.1       Introduction	65 68			

	4.3	Methodology	70	
	4.4	Experiments	76	
	4.5	Preliminary Results	79	
	4.6	Ablation Study	83	
	4.7	Training Stability	85	
	4.8	Conclusions	85	
5	Lin	k Prediction without Graph Neural Networks	87	
	5.1	Introduction	87	
	5.2	Limitations in supervised link prediction evaluation and training	89	
	5.3	Method	91	
	5.4	Experiments	96	
	5.5	Related work	114	
	5.6	Conclusion	115	
6	Fea	ture-based Individual Fairness in k-Clustering	117	
U	6 1	Introduction	117	
	6.2	Preliminaries	121	
	6.3	Results	121	
	6.4	Experiments	134	
	6.5	Running Time	140	
	6.6	Additional Experiments	141	
	6.7	Conclusions	142	
7	Rok	oustnoss of Human Decision Making under Risk	1/2	
1	7 1	Introduction	1/12	
	7.2	Human Experiments	145	
	7.2	Proliminarios	140	
	7.0	Analysis and Exporiments	140	
	$7.4 \\ 7.5$	Conclusions	171	
0	А.Т	Desision Sectores with Fredhesh Lean Asting Leanner	175	
0	$\mathbf{AI}$	Introduction	175	
	0.1	Polotod Works	170	
	0.4 0.2	Methodology	170	
	0.0	Fur entry entry	107	
	0.4 0 E	Conclusions	100	
	8.0		192	
9	Cor	nclusions	194	
Bibliography				

# Chapter 1

# Introduction

We inhabit a world that is rapidly propelled by data. With each passing day, we produce massive volumes of data through our engagements with digital devices, social media platforms, and various technologies. Recent calculations<sup>1</sup> indicate that we generate 329.77 billion gigabytes of data every day, and this quantity is increasing exponentially every year. The vast availability of data has given rise to new areas of research such as data mining and data science that can be applied to various domains ranging from healthcare to finance.

In this context, the abundance of data necessitates the extraction of key information to utilize the data more effectively in downstream tasks, which has led to the emergence of representation learning. This widely used technique in data science employs deep learning to identify critical features that are invisible to humans, allowing for more informative and comprehensive features to be extracted from the data. These features can then be applied to decision tasks such as credit card applications or molecular property prediction.

One of the biggest challenges with deep learning is its lack of transparency and explainability. Without understanding how a deep learning model reaches its decisions,

 $<sup>^{1}</sup> https://exploding topics.com/blog/data-generated-per-day$ 

there can be serious consequences in high-stake application sectors such as healthcare, finance, and law enforcement. There are multiple recent articles<sup>2</sup> discussing that explainability and transparency should be the main core of deep learning models. This will build trust between humans and AI, and also ethical problems using AI systems will not be an issue. A recent article from Nature<sup>3</sup> claims that some scientists are already using chatbots such as ChatGPT as their research assistants. However, the quality and correctness of these tools are questionable. Therefore, we need to explain these deep black-box models.

Furthermore, deep learning algorithms may struggle with ambiguous or uncertain situations, which are often encountered in decision-making tasks in high-stakes applications. This "gray area" problem highlights the importance of incorporating human expertise into AI systems. While AI can make decisions quickly, humans can provide a more nuanced understanding of the situation and help improve the accuracy and effectiveness of AI systems. Conversely, AI can be leveraged to gain a better understanding of human behavior and improve decision-making in fields such as psychology and sociology. Thus, it is essential to explore ways in which humans and AI can collaborate and mutually benefit from each other.

This dissertation focuses on transparent representation learning on graphs and explores collaborations between humans and AI. Graphs also referred to as networks, are extensively utilized in modeling various applications such as social networks, molecular interactions, road networks, and financial transactions. Additionally, graphs can serve as a representation of diverse data types, including tabular data, images, and text. The nodes and edges of graphs can possess attributes, which enhance their capability for effective representation and modeling. Furthermore, graphs can also incorporate a tem-

 $<sup>^{2}</sup> https://www.forbes.com/sites/forbestechcouncil/2023/01/23/why-explainability-should-be-the-core-of-your-ai-application/?sh=16a1493f753f$ 

<sup>&</sup>lt;sup>3</sup>https://www.nature.com/articles/d41586-023-00191-1

poral dimension, enabling the representation of dynamic graphs where nodes, edges, or attributes change over time. For instance, in social networks, nodes can represent individuals, each with their specific attributes, while edges capture the relationships between individuals, featuring attributes associated with those relationships. These relationships can evolve or new individuals may join the network over time.

As a powerful data structure, graphs are applicable to a range of downstream applications, including graph classification, node classification, link prediction, and clustering. Transparent representation learning is essential for encoding the graph in a representation space while also understanding the model's decision-making process. However, representation learning alone may not solve all problems and may require human assistance. Collaborating with AI can lead to even better and more effective models. On the other hand, fairness is a critical issue in some applications, such as clustering, where groups should be formed based on the similarity of individuals or the representativeness of the community.

The rest of the dissertation is organized by our contribution to the advancement of transparent graph representation learning including explainability and fairness, and human and AI collaboration, summarized as follows:

▷ Event Detection on Dynamic Graphs [2] (Chapter 2): Event detection is a critical task for timely decision-making in graph analytics applications. Despite the recent progress towards deep learning on graphs, event detection on dynamic graphs presents particular challenges to existing architectures. Real-life events are often associated with sudden deviations of the normal behavior of the graph. However, existing approaches for dynamic node embedding are unable to capture the graph-level dynamics related to events. In this paper, we propose DyGED, a simple yet novel deep learning model for event detection on dynamic graphs. DyGED learns correlations between the graph macro dynamics—i.e. a sequence of graph-level representations—and labeled events. Moreover, our approach combines structural and temporal self-attention mechanisms to account for application-specific node and time importances effectively. Our experimental evaluation, using a representative set of datasets, demonstrates that DyGED outperforms competing solutions in terms of event detection accuracy by up to 8.5% while being more scalable than the top alternatives. We also present case studies illustrating key features of our model.

- $\triangleright$  Global Counterfactual Explainer for Graph Neural Networks [3] (Chapter 3): Graph neural networks (GNNs) find applications in various domains such as computational biology, natural language processing, and computer security. Owing to their popularity, there is an increasing need to explain GNN predictions since GNNs are black-box machine learning models. One way to address this is *counterfactual* reasoning where the objective is to change the GNN prediction by minimal changes in the input graph. Existing methods for counterfactual explanation of GNNs are limited to instancespecific *local* reasoning. This approach has two major limitations of not being able to offer global recourse policies and overloading human cognitive ability with too much information. In this work, we study the *qlobal* explainability of GNNs through global counterfactual reasoning. Specifically, we want to find a *small* set of representative counterfactual graphs that explains all input graphs. Towards this goal, we propose GCFEXPLAINER, a novel algorithm powered by *vertex-reinforced random walks* on an *edit map* of graphs with a *greedy summary*. Extensive experiments on real graph datasets show that the global explanation from GCFEXPLAINER provides important high-level insights of the model behavior and achieves a 46.9% gain in recourse coverage and a 9.5% reduction in recourse cost compared to the state-of-the-art local counterfactual explainers.
- ▷ Robust Ante-hoc Graph Explainer using Bilevel Optimization [4] (Chapter 4): Explaining the decisions made by machine learning models for high-stakes applications

is critical for increasing transparency and guiding improvements to these decisions. This is particularly true in the case of models for graphs, where decisions often depend on complex patterns combining rich structural and attribute data. While recent work has focused on designing so-called post-hoc explainers, the question of what constitutes a good explanation remains open. One intuitive property is that explanations should be sufficiently informative to enable humans to approximately reproduce the predictions given the data. However, we show that post-hoc explanations do not achieve this goal as their explanations are highly dependent on fixed model parameters (e.g., learned GNN weights). To address this challenge, this paper proposes RAGE (Robust Ante-hoc Graph Explainer), a novel and flexible ante-hoc explainer designed to discover explanations for a broad class of graph neural networks using bilevel optimization. RAGE is able to efficiently identify explanations that contain the full information needed for prediction while still enabling humans to rank these explanations based on their influence. Our experiments, based on graph classification and regression, show that RAGE explanations are more robust than existing post-hoc and ante-hoc approaches and often achieve similar or better accuracy than state-of-the-art models.

▷ Link Prediction without Graph Neural Networks [5] (Chapter 5): Link prediction, which consists of predicting edges based on graph features, is a fundamental task in many graph applications. As for several related problems, Graph Neural Networks (GNNs), which are based on an attribute-centric message-passing paradigm, have become the predominant framework for link prediction. GNNs have consistently outperformed traditional topology-based heuristics, but what contributes to their performance? Are there simpler approaches that achieve comparable or better results? To answer these questions, we first identify important limitations in how GNN-based link prediction methods handle the intrinsic class imbalance of the problem—due to

the graph sparsity—in their training and evaluation. We then propose Gelato, a novel topology-centric framework that applies a topological heuristic to a graph enhanced by attribute information via graph learning. Gelato is more robust to class imbalance as it requires significantly fewer parameters than the GNN-based alternatives. Our model is trained end-to-end with an N-pair loss on an unbiased training set to address the class imbalance. Experiments show that Gelato is 84% more accurate, trains 3 times faster, and infers 10,000 times faster compared to state-of-the-art GNN for link prediction.

- Feature-based Individual Fairness in k-Clustering [6] (Chapter 6): Ensuring fairness in machine learning algorithms is a challenging and essential task. We consider the problem of clustering a set of points while satisfying fairness constraints. While there have been several attempts to capture group fairness in the k-clustering problem, fairness at an individual level is relatively less explored. We introduce a new notion of individual fairness in k-clustering based on features not necessarily used for clustering. We show that this problem is NP-hard and does not admit any constant factor approximation. Therefore, we design a randomized algorithm that guarantees approximation both in terms of minimizing the clustering distance objective and individual fairness under natural restrictions on the distance metric and fairness constraints. Finally, our experimental results against six competing baselines validate that our algorithm produces individually fairer clusters than the fairest baseline by 12.5% on average while also being less costly in terms of the clustering objective than the best baseline by 34.5% on average.
- ▷ Robustness of Human Decision Making under Risk [7] (Chapter 7): Prospect Theory has been extensively studied in the field of decision-making, providing valuable insights into individual risk preferences. However, the exploration of Group Prospect Theory, which examines decision-making within a collective context, remains relatively scarce.

This study highlights the need to investigate Group Prospect Theory and presents human experiments conducted to understand the dynamics of decision-making in group settings. Our research includes a comprehensive analysis that specifically aims to measure the behavioral shifts observed as individuals transition from making decisions in isolation to making decisions within a group setting. Additionally, the study explores the robustness of groups, particularly in relation to potential attacks facilitated by artificial intelligence (AI) with a design of a generative model of human behavior. By addressing these research questions, this study aims to contribute to a deeper understanding of Group Prospect Theory and shed light on the vulnerabilities and potential impacts of AI on group decision-making processes.

 $\triangleright$  AI Decision Systems with Feedback Loop Active Learner [8] (Chapter 8): Making precise decisions for high-stakes applications such as finance, health, and self-driving is critical for increasing the economy of an entity or the quality of life. In most scenarios, decision quickness is also as essential as accuracy. This is particularly true in the case of event detection problems, where late detection can cause financial or physical damage. While recent work focuses on combining fast unsupervised AI decision systems and precise human decisions to solve this problem, the quality of this cooperation remains questionable. A human can generate ground-truth labels for the AI decision systems for future improvements. However, having noisy ground truth can worsen the performance. To address this challenge, this paper proposes FLAL (Feedback Loop Active Learner), a novel bridge system between the AI decision system and human/s, designed to understand human expertise and interest using a recommender mechanism and improve AI system performance using an active learning mechanism. FLAL is able to identify human behavior and makes entity recommendations to users who can generate better ground-truth labels for these entities. Our experiments show that FLAL performs better than competing baselines and converges fast.

# Chapter 2

# **Event Detection on Dynamic Graphs**

## 2.1 Introduction

Event detection on dynamic graphs is a relevant task for effective decision-making in many organizations [9, 10]. In graphs, entities and their interactions are represented as (possibly attributed) nodes and edges, respectively. The graph dynamics, which changes the interactions and attributes over time, can be represented as a sequence of snapshots. Events, identified as snapshot labels, are associated with a short-lived deviation from normal behavior in the graph.

As an example, consider the communication inside an organization, such as instant messages and phone calls [11]. Can the evolution of communication patterns reveal the rise of important events—e.g., a crisis, project deadline—within the organization? While one would expect the content of these communications to be useful for event detection, this data is highly sensitive and often private. Instead, can events be discovered based only on structural information (i.e. message participants and their attributes)? For example, Romero et al. [11] have shown that stock price shocks induce changes (e.g., higher clustering) in the structure of a hedge fund communication network. Shortly after the 2011 earthquake and tsunami, Japanese speakers expanded their network of communication on Twitter [12].

Given the recent success of deep learning on graphs [13, 14, 15, 16] in node/graph classification, link prediction, and other tasks, it is natural to ask whether the same can also be useful for event detection. In particular, such an approach can combine techniques for graph classification [17, 18] and dynamic representation learning on graphs [19, 20, 21, 22]. However, a key design question in this setting is whether to detect events based on the micro (node) or macro (graph) level dynamics. More specifically, the micro dynamics is captured via the application of a pooling operator to dynamic node embeddings [19, 21]. For the macro dynamics, static snapshot embeddings are computed via pooling and their evolution is modeled via a recurrent architecture (e.g. an LSTM) [20, 22]. Each of these approaches has implicit assumptions about the nature of events in the data.

Figure 2.1 shows two event detection architectures, one based on micro and another based on macro dynamics. While they both apply a generic architecture shown in Figure 2.1a, they differ in the way dynamic representations for each graph snapshot are generated.

To illustrate the difference between micro and macro dynamics, let us revisit our organization example. For simplicity, we will assume that the pooling operator is the average. Dynamic node embeddings are learned (non-linear) functions of the evolution of an employee's attributed neighborhood. These local embeddings are expected to be revealing of an employee's communication over time. Thus, (pooled) micro embeddings will capture average dynamic communication patterns within the organization. On the other hand, by pooling static node embeddings, we learn macro representations for the communication inside the organization at each timestamp. The recurrent architecture will then capture dynamic communication patterns at the organization level. Pooling and the RNN thus act as (spatial/temporal) functions that can be composed in different ways—e.g. f(g(x)) vs g(f(x))—each encoding specific inductive biases for event detection. We will show that the choice between micro and macro models has significant implications for event detection performance.

This paper investigates the event detection problem on dynamic graphs. We propose DyGED (Dynamic Graph Event Detection), a graph neural network for event detection. DyGED combines a macro model with structural and temporal self-attention to account for application-specific node and time importances. To the best of our knowledge, our work is the first to apply either macro dynamics or self-attention for the event detection task. Despite its simplicity, differing from more recent approaches based on micro dynamics, DyGED outperforms state-of-the-art solutions in three representative datasets. These findings also have implications for other graph-level analytics tasks on dynamic graphs, such as anomaly detection, regression, and prediction.

One of the strengths of our study is its extensive experimental evaluation. While the event detection problem has been studied by a recent paper [23]—based on micro dynamics—our work provides key insights into some of the challenges and possible strategies for effective event detection. This is partly due to our representative list of datasets covering mobility, communication, and user-generated content data. Moreover, we present a few case studies illustrating the key features of our approach. Our main contributions are as follows:

- ▷ We present the first study comparing micro and macro deep learning architectures for event detection on dynamic graphs, showing the importance of this design choice for event detection performance;
- ▷ We propose DyGED, a simple yet novel deep learning architecture for event detection based on macro dynamics. DyGED applies both structural and temporal



(c) Learning graph embeddings based on macro dynamics

Figure 2.1: (a) Event detection on dynamic graphs based on a generic deep learning architecture. (b) At the micro scale, the dynamics is captured at node level using a temporal GNN architecture and then pooled for graph-level classification. (c) At the macro scale, the dynamics is captured at the graph level using an RNN over pooled (static) GNN node embeddings. Our work investigates how the dynamics at different scales affects event detection performance.

self-attention to enable the effective learning of node and time dependent weights;

▷ We compare DyGED against several baselines—mostly based on a micro model using three datasets. Our results show that DyGED outperforms the baselines by up to 8.5% while being scalable. We also provide case studies illustrating relevant features of our model (e.g. how its embeddings can be used for diagnosis).

## 2.2 Related Work

Event detection on graphs: There is a diverse body of work on event detection using graphs and other types of structured data [24, 25, 26] in the literature. Moreover, other popular tasks such as anomaly [27, 28], change point [29, 30], and intrusion [31] detection are related to unsupervised event detection [32, 28, 33]. Here, we focus on a supervised version of the problem where graph snapshots are labeled depending on whether an event happened within the snapshot's time window. We assume that events are defined based on an external source of information, not being fully identifiable from but still being correlated with—the observed data [9, 23]. Some studies distinguish the event detection from the event forecasting problem, as the second also allows for predictions of events far into the future [25]. However, such a distinction is less relevant when events are external to the data [23]. Recently in [34], the authors apply a dynamic graph to identify events on time series. We focus on the detection of events with graphs as inputs.

The classical approach for event detection, including in the case of graphs, is to rely on features designed by experts [35, 36]. A framework for automatically combining multiple social network metrics, such as modularity and clustering coefficient, as features for terrorist activity detection using a neural network is introduced in [9]. However, these metrics, which are based on expert knowledge, may not generalize to many applications, and they are potentially non-exhaustive—i.e. other relevant metrics might be missing.

**Graph kernels:** In machine learning, kernel methods, such as Support Vector Machines, have motivated a long literature on graph kernels [37]. A kernel is a function that computes the similarity between two objects in a particular application and thus their design often requires expert-knowledge. In the case of graph kernels, graphs are compared based on their substructures—e.g., node neighborhoods [38], graphlets [39],
random walks [40]. However, recent studies have shown that such features are often outperformed by those learned directly from data [17].

**Temporal node embeddings:** Modeling temporal evolution of nodes based on their connections over time is a well-studied problem in the literature. The temporal information can be encoded using temporal point process [41], random walks on temporal edge orderings [42] or a joint optimization of temporal node embeddings [43]. While these methods focus on micro embeddings, [44] proposes taking into account macro information by incorporating the number of modified edges in the graph. However, we notice that [44] does not apply to our setting since it does not find node embeddings for every timestamp.

Graph neural networks (GNNs) for dynamic graphs: Deep learning on graphs is an effort to reproduce the success achieved by deep neural networks (e.g. CNNs, RNNs) on graphs [45, 46]. A key advance was the introduction of Graph Convolutional Networks (GCNs) [13], which outperform traditional approaches for semi-supervised learning. Later, GraphSage [14] and Graph Attention Networks (GATs) [47] were proposed to increase the scalability and exploit the attention mechanism in GNNs, respectively.

Following the work listed above, there has been an outburst of extensions of GNNs for dynamic graphs [48]. One approach is to apply standard GNNs to a static graph where multiple snapshots are represented as a single multi-layered graph [49, 50] with additional temporal edges. More recently, several studies have combined recursive architectures, such as LSTMs, with GNNs. This can be achieved via macro models, which stack graphlevel GNN representations as a sequential process [51, 20]. Another alternative are micro models, which apply the sequential process at the node level to generate dynamic node embeddings [52, 21, 50].

Dynamic GNNs have been applied mostly for local tasks (e.g. node classification). On the other hand, the most popular graph-level task for GNNs is graph classification [53, 54, 55], which assumes the input to be static. Our paper is focused on event detection on dynamic graphs, a challenging graph-level task also studied in [23]—one of our baselines. Different from their work, DyGED applies self-attention at node and time domains to capture the macro dynamics correlated with events. Self-attention was also recently applied in [56] and [52], which were focused on graph classification and link prediction, respectively. We show that DyGED equipped with self-attention outperforms state-ofthe-art event detection methods.

## 2.3 Problem Definition

Supervised event detection on dynamic graphs consists of learning how to detect events based on a few recent graph snapshots using training data (i.e. labeled events).

**Definition 1.** Dynamic Graph: A dynamic graph  $\mathbb{G}$  is a sequence of T discrete snapshots  $\langle G_1, G_2, \ldots, G_T \rangle$  where  $G_t$  denotes the graph at timestamp t.  $G_t$  is a tuple  $(V, E_t, W_t, X_t)$  where V is a fixed set of n vertices,  $E_t$  is a set of  $m_t$  undirected edges,  $W_t : E_t \to \mathbb{R}_+$  are edge weights, and  $X_t : V \to \mathbb{R}^d$  gives d features for each node.

In our earlier example regarding an organization, nodes in V represent members of the organization. An edge is created in  $E_t$  whenever their associated members exchange a message during a time interval t and weights  $W_t$  might be numbers of messages exchanged. Finally, features  $X_t$  might include an individual's job position (static) and the total number of messages received by them during the time interval t (dynamic).

**Definition 2.** Event Label Function: We define the function  $\ell(G_{t-k:t}) \in \{0, 1\}$  to be an event labelling function of order k, where  $G_{t-k:t} = \langle G_{t-k}, G_{t-k+1}, \ldots, G_t \rangle$ , and such that:

$$\ell(G_{t-k:t}) = \begin{cases} 1, & \text{if an event occurs at time } t \\ 0, & \text{otherwise} \end{cases}$$

Events might depend not only on the current graph snapshot  $G_t$  but also on the k previous snapshots  $G_{t-k}, \ldots, G_{t-1}$ . This allows the function  $\ell$  to model events that depend on how the graph changes. One can define a similar function  $\ell_{\Delta}$  for the early detection (or forecasting) of events  $\Delta$  snapshots into the future.

**Definition 3.** Event Detection Problem: Given a set of training instances  $\mathbb{D}$ , composed of pairs  $(G_{t-k:t}, \ell(G_{t-k:t}))$ , learn a function  $\hat{\ell}$  that approximates the true  $\ell$  for unseen snapshots.

We treat event detection as a classification problem with two classes. To evaluate the quality of the learned function  $\hat{\ell}$  we apply a traditional evaluation metric (AUC [57]) from supervised learning. In this paper, we propose  $\hat{\ell}$  to be a neural network.

# 2.4 Proposed Model: DyGED

We describe DyGED (Dynamic Graph Event Detection), a simple yet novel deep learning architecture for event detection on dynamic graphs. DyGED combines a Graph Convolutional Network and a Recurrent Neural Network to learn the macro (i.e. graphlevel) dynamics correlated with labeled events. This backbone architecture is further enhanced by self-attention mechanisms in the structural and temporal domains. First, we introduce the main components of our architecture. Next, we describe DyGED and some of its variations.

We introduce notations for a few basic operations to describe our architectures in a compact form. A column-wise concatenation  $[M_1, \ldots, M_t] : \mathbb{R}^{n \times m_1} \times \ldots \times \mathbb{R}^{n \times m_t} \to$   $\mathbb{R}^{n \times (m_1 + \ldots + m_t)}$  maps a sequence of matrices  $M_1, \ldots, M_t$  to a new matrix M such that  $(M_t)_{i,j} = M_{i,\sum_{r=1}^{t-1} m_r+j}$ . Similarly, we denote a row-wise concatenation as  $[M_1; \ldots; M_t] = [M_1^{\mathsf{T}}, \ldots, M_t^{\mathsf{T}}]^{\mathsf{T}}$ .

#### 2.4.1 Main Components

We describe the main components of our neural network architecture for event detection on dynamic graphs (DyGED).

#### Graph Convolutional Network:

GCNs are neural network architectures that support the learning of *h*-dimensional functions  $\mathbf{GCN}(A, X) : \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times h}$  over vertices based on the graph adjacency matrix A and features X. For instance, a 2-layer GCN can be defined as follows:

$$\mathbf{GCN}(A, X) = \sigma \left( \hat{A} \ \sigma \left( \hat{A} X W^{(0)} \right) \ W^{(1)} \right)$$

where  $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$  is the normalized adjacency matrix with  $\tilde{D}$  as weighted degree matrix and  $\tilde{A} = I_n + A$  with  $I_n$  being an  $n \times n$  identity matrix.  $W^{(i)}$  is a weight matrix for the *i*-th layer to be learned during training, with  $W^{(1)} \in \mathbb{R}^{d \times h'}$ ,  $W^{(2)} \in \mathbb{R}^{h' \times h}$ , and h (h') being the embedding size in the output (hidden) layer. Moreover,  $\sigma(.)$  is a non-linear activation function such as **ReLU**.

#### Pooling

The output of the GCN described in the previous section is an embedding matrix  $Z_t$ for each graph snapshot  $G_t$ . In order to produce an embedding  $\mathbf{z}_t$  for the entire snapshot, we apply a pooling operator v-Att $(Z_t)$  :  $\mathbb{R}^{n \times h} \to \mathbb{R}^h$ . In particular, our model applies the self-attention graph pooling operator proposed in [56]:

$$\mathbf{z}_{\mathbf{t}} = v-\operatorname{Att}(Z_t) = \operatorname{softmax}(\mathbf{w}.\operatorname{tanh}(\Phi Z_t^T))Z_t$$

where  $\Phi \in \mathbb{R}^{h \times h}$  and  $w \in \mathbb{R}^h$  are learned attention weights.

Intuitively, v-Att re-weights the node embeddings enabling some nodes to play a larger role in the detection of events. In our experiments, we will show that these attention weights can be used to identify the most important nodes for our task.

#### **Recurrent Neural Network**

We assume that events are correlated with the graph (i.e. macro) dynamics. Thus, our model applies an RNN to learn dynamic graph representations. More specifically, we give the pooled snapshot embeddings  $\mathbf{z}_t$  as input to a standard Long Short-Term Memory  $\mathbf{LSTM}(\mathbf{z}'_{t-1}, \mathbf{z}_t) : \mathbb{R}^h \times \mathbb{R}^h \to \mathbb{R}^h$  to produce dynamic graph representations:

$$\mathbf{z}'_{\mathbf{t}} = \mathbf{LSTM}(\mathbf{z}'_{\mathbf{t}-1}, \mathbf{z}_{\mathbf{t}})$$

Notice that  $\mathbf{z}'_t$  is based on a sequence of static embeddings, instead of each node's (micro) dynamics. In our experiments, we will compare these two approaches using a diverse collection of datasets.

#### **Temporal Self-Attention**

The RNN component described in the previous section enables our architecture to capture the graph dynamics via embeddings  $\mathbf{z}'_{\mathbf{t}}$ . However, complex events might not be correlated only with the current graph representation but a window  $Z'_t = [\mathbf{z}'_{\mathbf{t}-\mathbf{k}}; \ldots; \mathbf{z}'_{\mathbf{t}}]$ . For instance, in mobility-related events (e.g. sports games), changes in the mobility dynamics will arise a few hours before the event takes place. Moreover, these correlations

might vary within a dataset due to the characteristics of each type of event. Thus, we propose a self-attention operator  $t-\operatorname{Att}(Z'_t) : \mathbb{R}^{(k+1)\times h} \to \mathbb{R}^h$  for aggregating multiple dynamic embeddings:

$$\mathbf{z}''_{\mathbf{t}} = \text{t-Att}(Z'_t) = \mathbf{softmax}(\mathbf{w}'.\mathbf{tanh}(\Phi'Z'_t))Z'_t$$

where  $\Phi' \in \mathbb{R}^{h \times h}$  and  $w' \in \mathbb{R}^h$  are learned attention weights.

Similar to v-Att (Section 2.4.1), t-Att enables the adaptive aggregation of dynamic embeddings. To the best of our knowledge, we are the first to apply a similar self-attention mechanism—which might be of independent interest—in dynamic GNN architectures.

#### **Classifier and Loss Function**

The final component our model is a Multi-Layer Perceptron  $MLP(z''_t) : \mathbb{R}^h \to \mathbb{R}^2$ that returns (nonlinear) scores for each possible outcome (i.e., event/no event)  $Y_t$ . Given training data with event labels  $\ell(G_{t-k:t})$ , the parameters of our model are learned (endto-end) by minimizing the *cross-entropy* of event predictions  $Y = \{Y_{k+1}, \ldots, Y_T\}$ . Note that event detection is a highly imbalanced problem—i.e. events are rare. We address this challenge by weighting our loss function terms with class ratios [58]. As a result, false negatives are more penalized than false positives.

$$-\sum_{t=k+1}^{T} (1-x)\ell(G_{t-k:t})\log(Y_{t,1}) + x(1-\ell(G_{t-k:t}))\log(Y_{t,2})$$

where  $\ell$  is the event label from Definition 2. Moreover, x and 1 - x positive (i.e., events) and negative sample ratios in the training set, respectively.

Algorithm 1 DyGED Forward Algorithm

**Require:** Sequence of snapshots  $G_{t-k:t}$ , previous dynamic state  $z'_{t-k-1}$  **Ensure:** Event probability 1: for  $\tau \in \{t - k, \dots, t\}$  do 2:  $Z_{\tau} \leftarrow GCN(G_{\tau}, X_{\tau})$ 3:  $z_{\tau} \leftarrow v$ -Att $(Z_{\tau})$ 4:  $z'_{\tau} \leftarrow LSTM(z'_{\tau-1}, z_{\tau})$ 5: end for 6:  $z''_{t} \leftarrow t$ -Att $([z'_{t-k}; \dots; z'_{t}])$ 7: return  $MLP(z''_{t}) = 0$ 

### 2.4.2 DyGED and its Variants

Algorithm 1 provides an overview of the forward steps of DyGED. It receives a sequence of snapshots  $G_{t-k:t}$  and the previous dynamic (LSTM) state  $z'_{t-k-1}$  as inputs. The output is the event probability for  $G_t$ . Notice that our assumption that macro dynamics of the graph is correlated with events of interest leads to a simple and modular model. Steps 3 and 5 correspond to the structural and temporal self-attention, respectively. In order to evaluate some of the key decisions involved in the design of DyGED, we also consider the following variations of our model:

- ▷ DyGED-CT (with contatenation): Replaces the LSTM (step 4) and t-Att (step 5) operators by a concatenation, with  $z''_t = ([z_{t-k}, \ldots, z_t]).$
- ▷ DyGED-NL (no LSTM): Removes the LSTM operator (step 4) from Algorithm 1, with  $z''_t = t-Att([z_{t-k}; ...; z_t]).$
- ▷ DyGED-NA (no attention): Removes the temporal self-attention operator t-Att (step 5), with  $z''_t = z'_t$ .

#### Time Complexity

Table 2.1 shows the time complexities for different variations of DyGED discussed in this section. Our methods are scalable as the time complexities are linear with the number of vertices (n) and edges (m) in the input graph.

	GCN + Pooling	LSTM	t-Att	MLP
DyGED-CT	$O((mh + nh^2)l_1)$	-	-	$O(kh^2 + h^2 l_2)$
DyGED-NL	$O((mh + nh^2)l_1)$	-	$O(kh^2)$	$O(h^2 l_2)$
DyGED-NA	$O((mh + nh^2)l_1)$	$O(h^2)$	-	$O(h^2 l_2)$
DyGED	$O((mh + nh^2)l_1)$	$O(h^2)$	$O(kh^2)$	$O(h^2 l_2)$

Table 2.1: Time complexities of our methods:  $m, n, h, l_1, k, l_2$  denote numbers of edges, nodes, embedding dimension, layers in GCN, past snapshots, layers in MLP. We assume the initial node feature dimension for GCN, input, cell, and output dimension for LSTM equal to embedding dimension h.

# 2.5 Experiments

We evaluate DyGED—which is our approach for event detection on dynamic graphs and its variations using a diverse set of datasets. We compare our solutions against state-of-the-art baselines for event detection, graph classification, and dynamic GNNs in terms of accuracy (Sec. 2.5.3) and efficiency (Sec. 2.5.7). We also provide a visualization of prediction scores to give more insight into the results (Sec. 2.5.4. Furthermore, we present more studies to have a more detailed picture of the effectiveness of DyGED across representative application scenarios using event embeddings (Sec. 2.5.6), importance via attention (Sec. 2.5.5), and ablation study including feature and pooling operator variations (Sec. 2.5.8). The implementation of DyGED and datasets are available online: https://www.github.com/mertkosan/DyGED.

	NYC Cab	Hedge Fund	$\mathbf{T}\mathbf{W}$	TW-Large
#Nodes (avg)	263	330	300	1000
#Edges (avg)	3717	557	1142	10312
$\# static \ features$	6	5	300	300
$\# dynamic \ features$	3	3	3	3
#Snapshots	4464	690	2557	2557
Snap. Period	hour	day	day	day
#Events	162	55	287	287

Table 2.2: The statistics of the datasets. TW is Twitter Weather.

#### 2.5.1 Datasets

Table 2.2 shows the main statistics of our datasets. The snapshot period is the interval  $[time_t, time_t + \Delta p)$  covered by each snapshot  $G_t$ , where  $\Delta p$  denotes the period. These datasets are representatives of relevant event detection applications. NYC Cab is an example of a mobility network with geo-tagged mass-gathering events (e.g., concerts, protests). Hedge Fund is a communication network for decision making in high-risk environments—as in other business settings and emergency response. Twitter Weather relates user-generated content with extreme events (e.g., terrorist attacks, earthquakes).

**NYC Cab:** Dataset based on sports events and hourly numbers of passengers transported between cab zones in NYC.<sup>1</sup> Nodes, edges, and their weights represent cab zones, inter-zone trips, and numbers of passengers transferred, respectively. Static node features are lat-long coordinates, boroughs, lengths, areas, and service zones. Dynamic features are the node degree, betweenness centrality, and clustering coefficient within a snapshot (i.e. one hour period). Baseball games involving the Yankees or the Mets in NYC are the events of interest.

Hedge Fund [11]: Dynamic network of employees at a hedge fund and their communications. Stock market shocks between January 2009 and September 2011 are the events. Nodes, edges, and edge weights represent employees, communications, and the

<sup>&</sup>lt;sup>1</sup>https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

total number of messages exchanged, respectively. Node features are employee's personal information such as company name, hiring time, gender, and position. We consider the node degree, betweenness centrality, and clustering coefficient as dynamic features. Each snapshot covers activities in a day, and events are price shocks—unexpected changes [11]— in the S&P500.

Twitter Weather: Dataset integrating weather-related tweets and significant weather events in the US from 2012 to 2018. Tweets were extracted from a large corpus made available by the Internet Archive<sup>2</sup>. Nodes, edges, and edge weights represent English words, word co-occurrences, and the number of co-occurrences, respectively. Starting from a small set of weather-related words, we employed an existing algorithm [59] to expand the set to 300 words. Pre-trained word2vec<sup>3</sup> [60] vectors were applied as static node features, whereas dynamic features are the node degree, betweenness centrality, and clustering coefficient during a one day interval. Weather events—with monetary damage of at least \$50M—were collected from the US National Climatic Data Center records.<sup>4</sup> We also created a larger version of Twitter Weather with 1000 words.

#### 2.5.2 Experimental Settings

#### **Baselines:**

We consider recent approaches that either focus on micro (node-level) dynamics [23, 21, 50] or are designed for graph classification [55]. If it is necessary, we apply our v-Att module (see Section 2.4.1) to get graph embeddings from node embeddings.

▷ **DynGCN** [23]: State-of-the-art architecture for event detection that combines representations from a GCN at each snapshot with a temporal encoder that carries in-

<sup>&</sup>lt;sup>2</sup>https://archive.org/details/twitterstream

<sup>&</sup>lt;sup>3</sup>https://code.google.com/archive/p/word2vec/

<sup>&</sup>lt;sup>4</sup>https://www.ncdc.noaa.gov/stormevents/

formation from the past.

- ▷ EvolveGCN [21]: Combines recurrent and graph convolutional neural networks to generate dynamic node embeddings.
- ▷ **ASTGCN** [50]: Graph convolutional network originally proposed for traffic prediction. It combines spatial and temporal-attention mechanisms. We adapt ASTGCN to our problem setting by considering k previous time dependencies instead of daily, weekly, and monthly ones.
- ▷ DiffPool [55]: Computes graph embeddings via a differentiable graph pooling method. Because this model is designed for classification, it does not account for the dynamics.

#### Other Settings

**Train/Test Splits:** We evaluate the methods using p-fold nested cross-validation [61], where p was set based on event frequency. Each method runs 20 times per train/test split (at least 100 repetitions/method), and we report average results. Train/test splits of 3720/744, 575/115, and 2192/365 snapshots are applied for the NYC Cab, Hedge Fund, Twitter Weather datasets, respectively.

Ratio of positive and negative samples: Overall percentages of positive samples (events) are 3.62%, 7.97%, and 11.22% for the NYC Cab, Hedge Fund, and Twitter Weather dataset, respectively. As we use nested cross-validation, train/test event ratios vary (depending on the fold) from/to 3.46/4.18%-3.61/3.37%, 3.57/12.5%-8.37/7.14% and 11.17/18.23%-13.53/4.14% for NYC Cab, Hedge Fund, and Twitter Weather respectively.

**Hyper-parameters:** We tune the hyper-parameters of our methods and baselines with a grid search. We find that training using Adam optimization with learning rate, dropout rate, and the batch size set to 0.005, 0.2, and 100, respectively, works well for our methods. The number of GCN layers, MLP layers, the size of the embedding set to 2, 2, and 64, respectively, are good choices for model parameters.

**Quality metric:** We compare the quality of the predictions by the methods using the Area under the ROC curve (AUC) [57].

Hardware: We run our experiments on a machine with NVIDIA GeForce RTX 2080 GPU (8GB of RAM) and 32 Intel Xeon CPUs (2.10GHz and 128GB of RAM).

	Method	NYC Cab	Hedge Fund	$\mathbf{TW}$	TW Large
Baselines: Micro Dynamics	EvolveGCN ASTGCN DynGCN	$\begin{array}{c} 0.842^{**} \pm 0.008 \\ 0.903^{**} \pm 0.003 \\ 0.901^{**} \pm 0.003 \end{array}$	$\begin{array}{c} 0.718^{**} \pm 0.011 \\ 0.753^* \pm 0.022 \\ 0.679^{**} \pm 0.030 \end{array}$	$\begin{array}{c} 0.782^{**}\pm 0.012\\ 0.747^{**}\pm 0.018\\ 0.713^{**}\pm 0.012 \end{array}$	$\begin{array}{c} 0.731^{**} \pm 0.011 \\ 0.722^{**} \pm 0.014 \\ 0.709^{**} \pm 0.007 \end{array}$
Classification	DiffPool	$0.887^{**}\pm 0.003$	$0.690^{**}\pm 0.020$	$0.766^{**}\pm 0.014$	$0.728^{**}\pm 0.007$
Proposed: Macro Dynamics	DyGED-CT DyGED-NL DyGED-NA DyGED	$\begin{array}{c} \underline{0.910} \pm 0.009 \\ \textbf{0.912} \pm 0.004 \\ 0.896^{**} \pm 0.004 \\ 0.905^{*} \pm 0.004 \end{array}$	$\begin{array}{c} 0.776 \pm 0.012 \\ 0.779^* \pm 0.012 \\ \underline{0.784}^* \pm 0.014 \\ 0.787 \pm 0.015 \end{array}$	$\begin{array}{c} 0.775^{**} \pm 0.014 \\ 0.791^{**} \pm 0.014 \\ \underline{0.800}^{*} \pm 0.009 \\ \textbf{0.810} \pm 0.012 \end{array}$	$\begin{array}{c} 0.743^{**}\pm 0.014\\ \underline{0.752}^{*}\pm 0.020\\ 0.734^{**}\pm 0.012\\ \textbf{0.760}\pm 0.014 \end{array}$

Table 2.3: AUC scores of event detection methods. The highest and second highest values for each column are in bold and underlined, respectively. Our methods, accounting for macro dynamics, achieve the best results, outperforming the best baseline (ASTGCN) by 4.5% on average and up to 8.5% (on Twitter Weather). We performed a paired t-test comparing the best model against the others (markers \*\* and \* indicate p-value < .01 and < .05, respectively).

#### 2.5.3 Event Detection Accuracy

Table 2.3 shows the event detection accuracy results in terms of AUC. For the approaches that consider a sliding window, reported results are the best ones among window sizes (k + 1) varying from one to five. The optimal window size for all these methods is either four or five. We use only static node features in our main experiments, and the effect of dynamic features will be shown in Section 2.5.8.

Results show that DyGED outperforms the competing approaches in all datasets. In particular, DyGED outperforms ASTGCN—best baseline—by 0.2%, 4.5%, 8.5%, and 5.2% for NYC Cab, Hedge Fund, Twitter Weather, and Twitter Weather Large, respectively (4.5% on average). Notice that, different from most baselines, our approach captures the macro-dynamics correlated with events. DyGED-NL and DyGED, which adopt temporal self-attention, achieve the best results indicating that it enables the learning of adaptive weights for different snapshots/times. Moreover, DyGED-NA and DyGED—using recurrent neural network to capture the macro dynamics—achieve better performance for the Hedge Fund and Twitter Weather datasets.



Figure 2.2: Scaled prediction scores ([0,1]) given by some of the event detection methods for Hedge Fund. Events are marked as vertical (red) lines. The first (grey) half covers some of the training events, while the remaining (white) is in the test set. Some events are annotated for illustration. Hedge Fund annotations are headlines from CNBC on the day of the event. Our method DyGED can identify several of the events while keeping the false positive rate lower.

#### 2.5.4 Visualization of Event Scores

Besides evaluating event detection approaches in terms of accuracy, it is also important to analyze how event scores are related to true events in the data. This is particularly relevant when events are determined based on a threshold.

Figure 2.2 shows the prediction scores for a set of events from the Hedge Fund dataset. It illustrates how events are distributed over time and also how specific events are predicted by the methods. Events are marked as vertical (red) lines. Moreover, scores are (min-max) scaled versions of the event (yes) class score (i.e.  $Y_{t,1}$ , as defined in Section 2.4.1).

We show events and scores for part of the training window (in gray) and also the complete test window (in white) of a particular fold. We also show the value of the S&P500 index and some headlines of the day from the business section of the CNBC News website.<sup>5</sup> These visualizations provide important insights regarding the nature of events in these datasets and also about the performance of event detection schemes.

Notice that prediction scores are correlated with events across methods, particularly the top-performing ones. However, most methods tend to suffer from false positive errors. This shows that the relationship between the graph dynamics and events is often weak (or noisy), which is a challenge for event detection methods.

Prediction scores also give us a better understanding of the differences between our method (DyGED) and baselines EvolveGCN and ASTGCN. Different from DyGED, these baselines capture the micro (or node-level) dynamics of the graph. On the other hand, our approach focuses on the macro (or graph-level) dynamics. First, DyGED is effective at fitting the training data. Moreover, DyGED also achieves better performance during testing, producing significantly fewer false positives. These results are consistent

<sup>&</sup>lt;sup>5</sup>https://www.cnbc.com

with the quantitative analysis provided in Section 2.5.3.



Figure 2.3: Top 10 taxi zones (in red) based on various metrics for NYC Cab dataset. Attention mechanism finds closer taxi zones to the stadiums compared to some node statistics such as betweenness centrality, clustering coefficient, and node degree. In other words, DyGED detects more crucial nodes for Baseball game detection.

#### 2.5.5 Importance via Attention

DyGED applies node and time self-attention weights for event detection. Here, we analyze these attention weights as a proxy to infer *importance* node and time importance. We notice that the use of attention weights for interpretability is a contentious topic in the literature [62, 63, 64]. Still, we find that these learned weights to be meaningful for our datasets. They also provide interesting insights regarding the role played by self-attention in our model.

#### Node Attention

A critical task in event detection on graphs is to measure the importance of nodes and subgraphs [65] based on the events of interest. As a step towards answering this question, we analyze attention weights learned the v-Att(.) operator—normalized by the softmax function. For each node, we compute the average weight learned. For a comparison, we also consider the following classical node importance measures from the network science



Figure 2.4: Illustration of the attention weights (normalized) of current and previous three snapshots for all datasets. While in NYC Cab, the current snapshot has more attention weights than the previous ones, attention weights in the Hedge Fund and Twitter datasets reveal more about the importance of temporal information. Results show that history plays a role in predicting the event.  $0^{th}$  refers to the current snapshot.

#### literature: degree, betweenness centrality, and clustering coefficient.

Figure 2.3 shows the top 10 most important taxi zones based on each importance measure for the NYC Cab dataset. Our attention weights find taxi zones near the baseball stadiums, whereas topology-based baseline measures select stations in downtown Manhattan and the airports. In Twitter Weather, where nodes are words, our solution set contains "fire", "warm", "tree", and "snow" as the top words while the topologybased baselines have "weather", "update", "fire", and "barometer". The results show that words found using our measure are more strongly associated with events of interest.

#### Time (Snapshot) Attention

We also propose a time importance module that uses temporal self-attention weights via the function t-Att(.), to measure how the past snapshots (time) affect event detection. Similar to the node attention weights, the attention values here also signify the value of the information from the snapshots. We use three previous (i.e., k = 4) and the current snapshot in our experiments. Figure 2.4 shows the attention weights (output of softmax) for snapshots (with mean and standard deviation). For NYC Cab, the current snapshot has significantly higher weights. However, the remaining datasets reveal more interesting attention patterns. For instance, in Hedge Fund, the importance of earlier weights can be associated with the definition of an event—a stock market shock. For Twitter Weather, events often last several days, and thus weights are expected to be more uniformly distributed.



Figure 2.5: An illustration of how DyGED graph embeddings support event diagnosis for three datasets. For each snapshot window  $G_{t-k:t}$ , we assign an event case (e.g. game at Yankees or Mets stadium for NYC Cab) or None. Results show that event causes tend to form clusters in the embedding for all datasets. Thus, events can be automatically diagnosed based on a few manually diagnosed ones in their embedding neighborhood. Better seen in color.

#### 2.5.6 Event Embeddings and Diagnosis

Figure 2.5 shows graph embeddings produced by DyGED for our three datasets. Each point corresponds to a sequence of snapshots  $G_{t-k:t}$ . These are the same embeddings given as input to the MLP in our architecture. We set the number of dimensions hof the embeddings originally to 64 and then project them to 2-D using tSNE [66]. We also annotate each embedding with the type/cause of the event. For NYC Cab, we consider the stadium (*Yankees* or Mets) where the game takes place. For Hedge Fund, we distinguish between up and down shocks. For Twitter Weather, we use the classification from the US National Weather Service (*blizzard*, *hurricane*, *flood*, *wildfire*, *tornado*, *storm* and *extreme*).<sup>6</sup> Notice that this additional information is not used for event detection.

As desired, events often form clusters in the embedding space for all datasets. That illustrates the ability of DyGED to produce discriminative representations that enable the classifier to identify events. However, notice that events and non-events are not completely separated in 2-D embeddings produced by DyGED which outperform all the baselines. This suggests that the event detection problem on dynamic graphs might be non-trivial.

It is also interesting to analyze whether DyGED embeddings can be useful for *event diagnosis*, which consists of providing users with information that enables the identification of causes for the events [67, 68]. More specifically, embeddings might capture not only whether an event occurs or not but also the nature of the event—i.e. events with the same cause are embedded near each other. Our results show that events of the same type/cause tend to cluster in the embedding space. In particular, in the Twitter Weather dataset, events of type 'Extreme' and 'Storm' could be potentially diagnosed using a simpler nearest neighbors scheme.

#### 2.5.7 Event Detection Efficiency

Table 2.4 shows testing times (in secs.) for a batch size of 100 data points for NYC Cab, Hedge Fund, Twitter Weather, and Twitter Weather Large, respectively. The window size is set to 4. Results show that DyGED is scalable—up to 206 and 29 times faster than the top 2 baselines (ASTGCN and EvolveGCN, respectively).

<sup>&</sup>lt;sup>6</sup>https://www.ncdc.noaa.gov/stormevents/pd01016005curr.pdf

	NYC Cab	Hedge Fund	$\mathbf{T}\mathbf{W}$	TW-Large
EvolveGCN	1.912	0.303	0.949	4.65
ASTGCN	13.57	2.131	6.916	24.41
DynGCN	0.064	0.019	0.080	0.622
DiffPool	0.057	0.016	0.045	0.482
DyGED	0.066	0.017	0.048	0.479

Table 2.4: Testing times (in secs.) for all the methods with windows length k+1 set to 4. DyGED is scalable and up to 206 times faster than the best baseline, ASTGCN.

#### 2.5.8 Ablation Study

#### **Feature Variation**

We also calculate three graph statistics as dynamic node features: node degree, betweenness centrality, and clustering coefficient. Table 2.5 shows results for static-only, dynamic-only, and static+dynamic node features for DyGED and the best baseline (AST-GCN) using all datasets. Results show that static features are often more relevant than dynamic ones—NYC Cab is the exception. Moreover, combining static and dynamic features often improves performance. We notice that for Twitter Weather (TW), static features (word embeddings) are quite expressive.

Method	NYC Cab	Hedge Fund	Twitter Weather
ASTGCN-S	0.903	0.753	0.774
DyGED-S	0.905	0.787	0.810
ASTGCN-D	0.894	0.741	0.747
DyGED-D	0.911	0.786	0.765
ASTGCN-SD	0.905	0.763	0.751
DyGED-SD	0.925	0.798	0.815

Table 2.5: AUC scores of DyGED and the best baseline method ASTGCN with different sets of node features. -S, -D, -SD suffixes are for static-only, dynamic-only, static+dynamic node features, respectively. TW is the Twitter Weather dataset.

Mean and max operators are two of the most common pooling operators in convolutional networks [69]. They have been used for down-sampling the data and their usefulness depends on the application and the dataset [70]. We compare these operators against self-attention graph pooling (Section 2.4.1). Table 2.6 shows that the self-attention operator (DyGED) outperforms the other operators for all datasets. These results highlight the importance of adaptive node weighting when capturing the macro dynamics for event detection, one of the essentials of our approach.

Method	NYC Cab	Hedge Fund	Twitter Weather
DyGED-Mean	0.879	0.704	0.773
DyGED-Max	0.880	0.729	0.729
DyGED	0.905	0.787	0.810

Table 2.6: AUC scores of our event detection method DyGED equipped with different pooling operators. Attention pooling outperforms mean and max pooling for all datasets using our method DyGED. TW is the Twitter Weather data.

# 2.6 Conclusions

This paper is focused on event detection on dynamic graphs. We have proposed a deep learning based method, DyGED, which learns correlations between the graph macro dynamics—i.e. a sequence of temporal graph representations—and events. We compared DyGED against multiple baselines using a representative set of datasets. Our approach outperformed the baselines in accuracy while being more scalable than the most effective one. We also showed how our method could be applied for event diagnosis as well as to provide interpretability via self-attention on nodes and snapshots. In future work, we want to develop hierarchical event detection architectures that are able to combine macro and micro dynamics. We are also interested in designing even more interpretable

models for event detection on graphs so that the discovered events can be associated with subgraphs and their dynamics.

# 2.7 Future Works - Multi-scale Anomalies [1]

Detecting suspicious patterns in attributed networks, such as transaction networks, is crucial for addressing vulnerabilities, yet challenging due to limited knowledge of anomaly characteristics. We focus on identifying cohesive anomalous communities operating at different scales based on size and deviation from normal node behavior. To address this, we propose an unsupervised framework using graph autoencoders to reconstruct the graph at multiple scales, flagging anomalous nodes based on unstable embeddings and poor reconstructions.

In their work, Tang et al. [71] analyze anomalous graphs by evaluating the spectral energy distribution. To further explore this analysis, we conduct spectral analysis to determine if multi-scale anomalies exhibit distinct spectral behavior. Figure 2.6 illustrates that anomalous graphs at different scales exhibit dominant frequency bands on CORA dataset [72], supporting the hypothesis that anomaly scale influences the spectral characteristics. While Tang et al. assume high spectral energy around  $\lambda = 1$ , our experimental results demonstrate that anomalies of varying scales exhibit unique spectral behavior. For instance, anomalies with scale 1 (anomalous clusters of size 10) dominate at  $\lambda = 1.2$ , scale 2 (anomalous clusters of size 50) at  $\lambda = [0.8, 0.1]$ , scale 3 (anomalous clusters of size 150) at  $\lambda = 0.6$ , and single-node anomalies dominate the higher frequency band at  $\lambda = 1.4$ . Therefore, our proposed method should account for distinguishing between anomalies of different scales by leveraging their distinctive spectral characteristics.

In order to focus on unsupervised learning, our proposed approach utilizes graph autoencoders to reconstruct the graph structure, training the encoder using the recon-



Spectral energy distribution on Cora with injected multi-scale anomalies

Figure 2.6: Spectral energy distributions of normal nodes, node-level anomalies, and three subgraph-level anomalies at different scales for Cora with injected anomalies. The spectral energy distributions of anomalous nodes concentrate more on the high frequency regions.

struction error. The decoder simply employs the inner product of node embeddings. The difference between the actual graph and the reconstructed graph yields the reconstruction error, calculated for each element (edge) individually. To compute node-level reconstruction, we consider the edges connected to each specific node.

To capture distinctive spectral patterns of multi-scale anomalies, we introduce the use of Beta Wavelet Graph Neural Networks (GNNs) [71] for learning node embeddings in our encoders. The Beta wavelet kernel, characterized by hyperparameters p and q, is defined as  $W_{p,q} = \beta_{p,q}(L) = \frac{(L/2)^p(I-L/2)^q}{2B(p+1,q+1)}$ , where  $B(p+1,q+1) = \frac{p!q!}{(p+q+1)!}$  is a constant. Beta kernels can effectively capture anomalies at different scales compared to the more commonly used heat kernels in GNNs. Figure 2.7 provides a visual comparison of the spectral properties between heat kernels and Beta kernels with varying hyperparameters.

We employ  $\tau$  Beta wavelets, where p and q are integers satisfying  $p, q \in \mathbb{Z}^+$  and  $p + q \leq \tau$ , to generate embeddings  $Z_{0,\tau}, Z_{1,\tau-1}, \cdots, Z_{\tau,0}$ . In Eq. 2.1, a scale-specific



Figure 2.7: Spectral properties of heat kernels and Beta kernels are compared, with the latter incorporating diverse band-pass filters to enable the detection of anomalies at multiple scales.

wavelet, parameterized by unique values of p and q, is applied to the transformed input features. In contrast, we utilize these scale-specific embeddings for scale-specific graph reconstruction. The final embedding is obtained as a weighted sum of the multi-scale filter outputs in Eq. 2.2 using a learned weight vector  $\alpha$ . For each embedding  $Z_{p,q}$ , we generate a scale-specific graph reconstruction.

$$\mathbf{Z}_{p,q}^{(s)} = \mathbf{W}_{p,q} \mathrm{MLP}_s(\mathbf{X}) \tag{2.1}$$

$$\mathbf{Z}^{(s)} = \sum_{p=0,q=\tau}^{\tau,0} \alpha_{p,q}^{(s)} \mathbf{Z}_{p,q}^{(s)}$$
(2.2)

In our subsequent steps, we will compare our method with competing baselines in terms of detecting node or subgraph-level anomalies. If the competing methods are designed for supervised learning, we will adapt their loss functions to unsupervised ones to ensure a fair comparison. As datasets may vary, we also aim to investigate which filters are responsible for detecting anomalies at specific scales by analyzing the  $\alpha$  parameters.

# Chapter 3

# Global Counterfactual Explainer for Graph Neural Networks

# 3.1 Introduction

Graph Neural Networks (GNNs) [13, 14, 73, 74, 75, 76] are being used in many domains such as drug discovery [77], chip design [78], combinatorial optimization [79], physical simulations [80, 81] and event prediction [82, 83, 84]. Taking the graph(s) as input, GNNs are trained to perform various downstream tasks that form the core of many real-world applications. For example, graph classification has been applied to predict whether a drug would exhibit the desired chemical activity [77]. Similarly, node prediction is used to predict the functionality of proteins in protein-protein interaction networks [85] and categorize users into roles on social networks [86].

Despite the impressive success of GNNs on predictive tasks, GNNs are *black-box* machine learning models. It is non-trivial to explain or reason why a particular prediction is made by a GNN. Explainability of a prediction model is important to understand its shortcomings and identify areas for improvement. In addition, the ability

to explain a model is critical towards making it trustworthy. Owing to this limitation of GNNs, there has been significant efforts in recent times towards explanation approaches.

Existing work on explaining GNN predictions can be categorized mainly in two directions: 1) factual reasoning [65, 87, 88, 89], and 2) counterfactual reasoning [90, 91, 92, 93]. Generally speaking, the methods in the first category aim to find an important subgraph that correlates most with the underlying GNN prediction. In contrast, the methods with counterfactual reasoning attempt to identify the smallest amount of perturbation on the input graph that changes the GNN's prediction, for example, removal/addition of edges or nodes.



Figure 3.1: Formaldehyde (a) is classified by a GNN to be an undesired mutagenic molecule with its important subgraph found by factual reasoning highlighted in red. Formic acid (b) is its non-mutagenic counterfactual example obtained by removing one edge and adding one node and two edges.

Compared to factual reasoning, counterfactual explainers have the additional advantage of providing the means for recourse [94]. For example, in the applications of drug discovery [77, 95], mutagenicity is an adverse property of a molecule that hampers its potential to become a marketable drug [96]. In Figure 3.1, formaldehyde is classified by a GNN to be mutagenic. Factual explainers can attribute the subgraph containing the carbon-hydrogen bond to the cause of mutagenicity, while counterfactual explainers provide an effective way (i.e., a recourse) to turn formaldehyde into formic acid, which is non-mutagenic, by replacing a hydrogen atom with a hydroxyl.

In this work, we focus on counterfactual explanations. Our work is based on the observation that existing counterfactual explainers [65, 87, 88, 89] for graphs take a *local* 

perspective, generating counterfactual examples for individual input graphs. However, this approach has two key limitations:

- ▷ Lack of global insights: It is desirable to offer insights that generalize across a multitude of data graphs. For example, instead of providing formic acid as a counterfactual example to formaldehyde, we can summarize global recourse rules such as "Given any molecule with a carbonyl group (carbon-oxygen double bond), it needs a hydroxy to be non-mutagenic". This focus on global counterfactual explanation promises to provide higher-level insights that are complementary to those obtained from local counterfactual explanations.
- ▷ Information overload: The primary motivation behind counterfactual analysis is to provide human-intelligible explanations. With this objective, consider real-world graph datasets that routinely contain thousands to millions of graphs. Owing to instance-specific counterfactual explanations, the number of counterfactual graphs grows linearly with the graph dataset size. Consequently, the sheer volume of counterfactual graphs overloads human cognitive ability to process this information. Hence, the initial motivation of providing human-intelligible insights is lost if one does not obtain a holistic view of the counterfactual graphs.

**Contributions:** In this paper, we study the problem of model-agnostic, *global* counterfactual explanations of GNNs for graph classification. More specifically, given a graph dataset, our goal is to counterfactually explain the largest number of input graphs with a small number of counterfactuals. As we will demonstrate later in our experiments, this formulation naturally forces us to remove redundancy from instance-specific counterfactual explanations and hence has higher information density. Algorithmically, the proposed problem introduces new challenges. We theoretically establish that the proposed problem is NP-hard. Furthermore, the space of all possible counterfactual graphs itself is exponential. Our work overcomes these challenges and makes the following contributions:

- ▷ Novel formulation: We formulate the novel problem of global counterfactual reasoning/explanation of GNNs for graph classification. In contrast to existing works on counterfactual reasoning that only generate instance-specific examples, we provide an explanation on the global behavior of the model.
- ▷ Algorithm design: While the problem is NP-hard, we propose GCFEXPLAINER, which organizes the exponential search space as an *edit map*. We then perform *vertex-reinforced random walks* on it to generate diverse, representative counterfactual candidates, which are *greedily summarized* as the global explanation.
- ▷ Experiments: We conduct extensive experiments on real-world datasets to validate the effectiveness of the proposed method. Results show that GCFEXPLAINER not only provides important high-level insights on the model behavior but also outperforms state-of-the-art baselines related to counterfactual reasoning in various recourse quality metrics.

## **3.2** Global Counterfactual Explanations

This section introduces the global counterfactual explanation (GCE) problem for graph classification. We start with the background on local counterfactual reasoning. Then, we propose a representation of the global recourse rule that provides a high-level counterfactual understanding of the classifier behavior. Finally, we introduce quality measures for recourse rules and formally define the GCE problem.

#### 3.2.1 Local Counterfactual

Consider a graph G = (V, E), where V and E are the sets of (labelled) nodes and edges respectively. A (binary) graph classifier (e.g., a GNN)  $\phi$  classifies G into either the undesired class ( $\phi(G) = 0$ ) or the desired one ( $\phi(G) = 1$ ). An explanation of  $\phi$  seeks to answer how these predictions are made. Those based on factual reasoning analyze what properties G possesses to be classified in the current class while those based on counterfactual reasoning find what properties G needs to be assigned to the opposite class.

Existing counterfactual explanation methods take a local perspective. Specifically, for each input graph G, they find a **counterfactual** (graph) C that is somewhat similar to G but is assigned to a different class. Without loss of generality, let G belong to the undesired class, i.e.,  $\phi(G) = 0$ , then the counterfactual C satisfies  $\phi(C) = 1$ . The similarity between C and G is quantified by a predefined distance metric d, for example, the number of added/removed edges [90, 91].

In our work, we consider the graph edit distance (GED) [97], a more general distance measure, as the distance function to account for other types of changes. Specifically,  $(G_1, G_2)$  counts the minimum number of "edits" to convert  $G_1$  to  $G_2$ . An "edit" can be the addition or removal of edges and nodes, or change of node labels (see Figure 3.2). Moreover, to account for graphs of different sizes, we normalize the GED by the sizes of graphs:  $(G_1, G_2) = (G_1, G_2)/(|V_1| + |V_2| + |E_1| + |E_2|)$ . Nonetheless, our method can be applied with other graph distance metrics, such as those based on graph kernels (e.g., RW [98], NSPDG [99], WL [100]).

The distance function measures the quality of the counterfactual found by the explanation model. Ideally, the counterfactual C should be very close to the input graph G while belonging to a different class. Formally, we define the counterfactuals that are within a certain distance  $\theta$  from the input graph as **close counterfactuals**.

**Definition 1** (Close Counterfactual). Given the GNN classifier  $\phi$ , distance parameter  $\theta$ , and an input graph G with undesired outcome, i.e.,  $\phi(G) = 0$ ; a counterfactual graph,



Figure 3.2: Edits between graphs.

C, is a close counterfactual of G when  $\phi(C) = 1$  and  $d(G, C) \leq \theta$ .

While the (close) counterfactual C found by existing methods explains the classifier behavior for the corresponding input graph G, it is hard to generalize to understand the global pattern. Next, we introduce the global recourse rule that provides a high-level summary of the classifier behavior across different input graphs.

#### 3.2.2 Global Recourse Representation

The global counterfactual explanation requires a global recourse rule r. Specifically, for any (undesired) input graph G with  $\phi(G) = 0$ , r provides a (close) counterfactual (i.e., a recourse) for G:  $\phi(r(G)) = 1$ . While both a recourse rule and a local counterfactual explainer find a counterfactual given an input graph, their goals are different. The goal of the local counterfactual explainer is to find the best (closest) counterfactual possible for each input graph, and therefore, r can be very complicated, e.g., in the form of an optimization algorithm [91, 93]. On the other hand, a recourse rule aims to provide an explanation of the classifier's global behavior, which requires a simpler form that is understandable for domain experts without prior knowledge of deep learning on graphs.

Existing global recourse rules for classifiers with feature vectors as input take the

form of short decision trees [101]. However, this is hard to be generalized to graph data with rich structure information. Instead, we propose the representation of a global recourse rule for a graph classifier to be a collection of counterfactual graphs  $\mathbb{C}$  in the desired class that are *diverse* and *representative* enough to capture its global behavior. This representation does not require any additional knowledge for domain experts to understand and draw insights from, similar to the local counterfactual examples. It is also easy to find the local counterfactual for a given input graph G based on  $\mathbb{C}$  by nominating the closest graph in  $\mathbb{C}$ :  $r(G) = \arg \min_{C \in \mathbb{C}} d(G, C)$ .

#### 3.2.3 Quantifying Recourse Quality

Given a graph classifier  $\phi$  and a set of n input graphs  $\mathbb{G}$  in the undesired class, we want to compare the quality of different recourse representations  $\mathbb{C}$ . Similar to the quality metrics introduced for vector data [101], we aim to account for the following factors:

1. Coverage: Like local counterfactual explainers, we want to ensure that counterfactuals found for individual input graphs are of high quality. Specifically, we introduce recourse coverage—the proportion of input graphs that have close counterfactuals from  $\mathbb{C}$  under a given distance threshold  $\theta$ :

$$(\mathbb{C}) = |\{G \in \mathbb{G} \mid \min_{C \in \mathbb{C}} \{d(G, C)\} \leqslant \theta\}| / |\mathbb{G}|$$

2. **Cost**: Another quality metric based on local counterfactual quality is the recourse cost (i.e., the distance between the input graph and its counterfactual) across the input graphs:

$$(\mathbb{C}) =_{G \in \mathbb{G}} \{ \min_{C \in \mathbb{C}} \{ d(G, C) \} \}$$

where is an aggregation function, e.g., mean or median.

3. **Interpretability**: Finally, the recourse rule should be easy (small) enough for human cognition. We quantify the interpretability as the size of recourse representation:

$$(\mathbb{C}) = |\mathbb{C}|$$

#### 3.2.4 Problem Formulation and Characterization

An ideal recourse representation should maximize the coverage while minimizing the cost and the size. Formally, we define the global counterfactual explanation problem as follows:

**Problem 1** (Global Counterfactual Explanation for Graph Classification (GCE)). Given a GNN graph classifier  $\phi$  that classifies n input graphs  $\mathbb{G}$  to the undesired class 0 and a budget  $k \ll n$ , our goal is to find the best recourse representation  $\mathbb{C}$  that maximizes the recourse coverage with size k:

$$\max_{\mathbb{C}}(\mathbb{C}) \ s.t. \ (\mathbb{C}) = k$$

We note that in our problem formulation only coverage and size are explicitly accounted for, whereas cost is absent. We make this design choice since cost and coverage are intrinsically opposing forces. Specifically, if we are willing to allow a high cost, coverage increases since we allow for higher individual distances between an input graph and its counterfactual. Therefore, we take the approach of binding the cost to the distance threshold  $\theta$  in the coverage definition. Nonetheless, an explicit analysis of all these metrics including cost is performed to quantify recourse quality during our empirical evaluation in Section 3.4. Below we discuss the hardness of GCE.

**Theorem 1** (NP-hardness). *The GCE problem is NP-hard.* 

#### Proof:

To establish NP-hardness of the proposed problem we reduce it from the classical *Maximum Coverage* problem.

**Definition 2** (Maximum Coverage). Given a budget k and a collection of subsets  $S = \{S_1, \dots, S_m\}$  from a universe of items  $U = \{u_1, \dots, u_n\}$ , find a subset  $S' \subseteq S$  of sets such that  $|S'| \leq k$  and the number of covered elements  $|\bigcup_{\forall S_i \in S'} S_i|$  is maximized.

We show that given any instance of a maximum coverage problem  $\langle S, U \rangle$ , it can be mapped to a GCE problem. For  $u_i$ , we construct a star graph with a center node with an empty label and n leaf nodes with n-1 empty labels and one label  $u_i$ . For  $S_i$ , we construct a similar star graph with a center node with a special label  $\gamma$  and n leaf nodes with  $|S_i|$  labeled with the elements in  $S_i$  and  $n - |S_i|$  with empty labels. The classifier  $\phi$ classifies a graph as a desired one if and only if it is a star graph with a  $\gamma$ -labeled central node and n leaf nodes with a set of labels among  $S = \{S_1, \dots, S_m\}$ . The allowed edit operations are either adding or deleting a set of labels (as a single edit), but not both together. So, each  $S_i$  corresponds to a counterfactual candidate  $C_i$  and  $d(G_j, C_i) \leq \theta = 1$ if and only if  $u_j \in S_i$ . With this construction, it is easy to see that an optimal solution for this instance of GCE is the optimal solution for the corresponding instance of the maximum coverage problem.

Owing to NP-hardness, it is not feasible to identify the optimal solution for the GCE problem in polynomial time unless NP = P. In the next section, we will introduce GCFEXPLAINER, an effective and efficient heuristic that solves the GCE problem.

# **3.3** Proposed Method: GCFExplainer

In this section, we propose GCFEXPLAINER, the first global counterfactual explainer for graph classification. The GCE problem requires us to find a collection of k counterfactual graphs that maximize the coverage of the input graphs. Intuitively, we want each individual counterfactual graph to be a close counterfactual to (i.e., "cover") as many input graphs as possible. Additionally, different counterfactual graphs should cover different sets of input graphs to maximize the overall coverage. These intuitions motivate the design of our algorithm GCFEXPLAINER, which has three major components:

- 1. Structuring the search space: The search space of counterfactual graphs consists of *all* graphs that are in the same domain as the input graphs and within a distance of  $\theta$ . In other words, any graph within a distance of  $\theta$  from an input graph may be a potential counterfactual candidate and therefore needs to be analyzed. The number of potential graphs within  $\theta$  increases exponentially with  $\theta$  since the space of graph edits is combinatorial [102, 103]. GCFEXPLAINER uses an *edit map* to organize these graphs as a meta-graph  $\mathcal{G}$ , where individual nodes are graphs that are created via a different number of edits from the input graphs and each edge represents a single edit.
- 2. Vertex-reinforced random walk: To search for good counterfactual candidates, GCFEXPLAINER leverages vertex-reinforced random walks (VRRW) [104] on the edit map  $\mathcal{G}$ . VRRW has the nice property of converging to a set of nodes that are both important (i.e., cover many input graphs) and diverse (i.e., non-overlapping coverage), which will form a small set of counterfactual candidates for further processing.
- 3. Iterative computation of the summary: After obtaining good counterfactual candidates from VRRW, GCFEXPLAINER creates the final set of the counterfactual graphs (i.e., the summary) as the recourse representation by iteratively adding the best candidate based on the maximal gain of the coverage given the already added candidates.

#### 3.3.1 Structuring the Search Space

The search space for counterfactual graphs in GCFEXPLAINER is organized via an edit map  $\mathcal{G}$ . The edit map is a meta-graph whose nodes are graphs in the same domain as the input graphs and edges connect graphs that differ by a single graph edit. As an example, each graph in Figure 3.2 represents a node in the edit map, and the arrows denote edges between graphs (nodes) that are one edit away. In the edit map, we only include connected graphs since real graphs of interest are often connected (e.g., molecules, proteins, etc.).

While all potential counterfactual candidates are included as its nodes, the edit map has an exponential size and it is computationally prohibitive to fully explore it. However, a key observation is that a counterfactual candidate can only be a few hops away from some input graph. Otherwise, the graph distance between the counterfactual and the input graph would be too large for the counterfactual to cover it. This observation motivates our exploration of the edit map to be focused on the union of close neighborhoods of the input graphs (see Section 3.3.2). Additionally, while we cannot compute the entire edit map, it is easy to chart the close neighborhoods by iteratively performing all possible edits from the input graphs. Next, we introduce the vertex-reinforced random walk to efficiently explore the edit map to find counterfactual candidates.

#### 3.3.2 Vertex-Reinforced Random Walk

Vertex-reinforced random walk (VRRW) [104] is a time-variant random walk. Different from other more widely applied random walk processes such as the simple random walk and the PageRank [105, 106, 107, 108], the transition probability p(u, v) of VRRW from node u to node v depends not only on the edge weight w(u, v) but also the number of previous visits in the walk to the target node v, which we denote using N(v). Specifically,

$$p(u,v) \propto w(u,v)N(v) \tag{3.1}$$

GCFEXPLAINER applies VRRW on the edit map and produces n most frequently visited nodes in the walk as the set of counterfactual candidates S. Next, we formalize VRRW in our setting and explain how it surfaces good counterfactual candidates for GCE.

#### Vertex-reinforcement

Our main motivation for using VRRW to explore the edit map instead of other random walk processes is that VRRW converges to a diverse and representative set of nodes [109, 110] in different regions of the edit map. In this way, the frequently visited nodes in instances of VRRW have the potential to be good counterfactual candidates as they would cover a diverse set of input graphs in the edit map. The reason behind the diversity of the highly visited nodes is the previous visit count N(v) in the transition probability. Specifically, nodes with larger visit counts tend to be visited more often later ("richer gets richer"), and thereby dominating all other nodes in their neighborhood. This leads to a bunch of highly visited nodes to "represent" each region of the edit map. We refer the readers to [109] for details on the mathematical basis and the theoretical correctness of this property. Moreover, as our goal is to find counterfactual candidates, we only reinforce (i.e., increase the visit counts of) graphs in the counterfactual class.

#### Importance function

While the vertex-reinforcement mechanism ensures diversity of the highly visited nodes, we still need to guide the walker to visit graphs that are good counterfactual candidates. We achieve this by assigning large edge weight w(u, v) to good counterfactual candidates via an importance function I(v):

$$w(u,v) = I(v) \tag{3.2}$$

The importance function I(v) should capture the quality of a graph v as a counterfactual candidate. It has the following components:

- 1. Counterfactual probability  $p_{\phi}(v)$ . The graph classifier  $\phi$  predicts a probability for v to be in the counterfactual class ( $\phi(v) = 1$ ). By using it as part of the importance function, the walker is encouraged to visit regions with rich counterfactual graphs.
- 2. Individual coverage ( $\{v\}$ ). The individual coverage of a graph v computes the proportion of input graphs that are close to v. This encourages the walker to visit graphs that cover a large number of input graphs.
- 3. Gain of coverage (v; S). Given a graph v and the current set of counterfactual candidates S (i.e., the n most frequently visited nodes), we can compute the gain between the current coverage and the coverage after adding v to S:

$$(v; \mathbb{S}) = (\mathbb{S} \cup \{v\}) - (\mathbb{S})$$

This guides the walker to find graphs that complement the current counterfactual candidates to cover additional input graphs.

The importance function is a combination of these components:

$$I(v) = p_{\phi}(v)(\alpha(\{v\}) + (1 - \alpha)(v; \mathbb{S}))$$
(3.3)

where  $\alpha$  is a hyperparameter between 0 and 1. With the above importance function, the
VRRW in GCFEXPLAINER converges to a set of diverse nodes that have high counterfactual probability and collectively cover a large number of input graphs.

#### Dynamic teleportation

The last component of VRRW, teleportation, is to help us manage the exponential search space of the edit map. Since our goal is to find close counterfactuals to the input graphs, the walker only needs to explore the nearby regions of the input graphs. Therefore, we start the walk from the input graphs, and also at each step, let the walker teleport back (i.e. transit) to a random input graph with probability  $\tau$ .

To decide which input graph to teleport to, we adopt a dynamic probability distribution based on the current counterfactual candidate set S. Specifically, let  $g(G) = |\{v \in S \mid d(v,G) \leq \theta \text{ and } \phi(v) = 1\}|$  be the number of close counterfactuals in S covering an input graph G. Then the probability to teleport to G is

$$p_{\tau}(G) = \frac{\exp(-g(G))}{\sum_{G' \in \mathbb{G}} \exp(-g(G'))}$$
(3.4)

This dynamic teleportation favors input graphs that are not well covered by the current solution set and encourages the walker to explore nearby counterfactuals to cover them after teleportation.

#### 3.3.3 Iterative Computation of the Summary

We have applied VRRW to generate a good set of n counterfactual candidates S. In the last step of GCFEXPLAINER, we aim to further refine the candidate set and create the final recourse representation (i.e., the summary) with k counterfactual graphs. This summarization problem is also NP-hard and we propose to build  $\mathbb{C}$  in an iterative and greedy manner from S. Specifically, we start with an empty solution set  $\mathbb{C}_0$ . Then, for each iteration t, we add the graph v to  $\mathbb{C}_t$  with the maximal gain of coverage  $(v; \mathbb{C}_t)$ . This is repeated k times to get the final recourse representation  $\mathbb{C}$  with k graphs. It is easy to show that the summarization problem is submodular and therefore, our greedy algorithm provides (1-1/e)-approximation.

Notice that the greedy algorithm can also be applied to the local counterfactuals found by existing methods to generate a GCE solution. Here, we highlight three advantages of GCFEXPLAINER:

- Existing local counterfactual explainers [90, 91, 92, 93] are only able to generate counterfactuals based on one type of graph edits—edge removal, while GCFEx-PLAINER incorporates all types of edits to include a richer set of counterfactual candidates.
- 2. The set of counterfactual candidates from GCFEXPLAINER is generated with the GCE objective in mind, while the local counterfactuals from existing methods are optimized for individual input graphs. Therefore, they may not be good candidates to capture the global behavior of the classifier.
- 3. It is easy to incorporate domain constraints (e.g., the valence of chemical bonds) into GCFEXPLAINER by pruning the neighborhood of the edit map, while existing methods based on optimization require non-trivial efforts to customize.

We will empirically demonstrate the superiority of GCFEXPLAINER to this two-stage approach with state-of-the-art local counterfactual explanation methods in our experiments in Section 3.4.2.

**Pseudocode and complexity:** The pseudocode of GCFEXPLAINER is presented in 2. Line 1-16 summarizes the VRRW component of GCFEXPLAINER. Specifically, Line 3-10 Algorithm 2 GCFEXPLAINER( $\phi$ ,  $\mathbb{G}$ )

1:  $G \leftarrow$  random input graph from  $\mathbb{G}$ ,  $N(G) \leftarrow 1$ ,  $\mathbb{S} = \{G\}$ 2: for  $i \in 1 : M$  do Let  $\varepsilon \sim Bernoulli(\tau)$ 3: 4: if  $\varepsilon = 0$  then 5:for  $v \in Neighbors(G)$  do Compute I(v) based on Equation 3.3 6: 7: Compute p(G, v) based on Equation 3.1 8: end for 9:  $v \leftarrow$  random neighbor of G based on p(G, v)10: else  $v \leftarrow$  random input graph from  $\mathbb{G}$  based on Equation 3.4 11: 12:end if if  $\phi(v) = 1$  then 13:if  $v \in \mathbb{S}$  then 14:15: $N(v) \leftarrow N(v) + 1$ 16:else  $\mathbb{S} \leftarrow \mathbb{S} + \{v\}, N(v) \leftarrow 1$ 17:end if 18: end if 19:20:  $G \leftarrow v$ 21: end for 22:  $\mathbb{S} \leftarrow \text{top } n$  frequently visited counterfactuals in  $\mathbb{S}$ 23:  $\mathbb{C} \leftarrow \emptyset$ 24: for  $t \in 1 : k$  do 25: $v \leftarrow \arg \max_{v \in \mathbb{S}} (v; \mathbb{C})$  $\mathbb{C} \leftarrow \mathbb{C} + \{v\}$ 26:27: end for 28: return  $\mathbb{C} = 0$ 

determines the next graph to visit based on VRRW transition probabilities and dynamic teleportation, and Line 11-16 update the visit counts and the set of counterfactual candidates. The iterative computation of the counterfactual summary is described in Line 17-21. The overall complexity of GCFEXPLAINER is O(Mhn + kn), where M is the number of iterations for the VRRW, h is the average node degree in the meta-graph, n is the number of input graphs, and k is the size of the global counterfactual representation. In practice, we store the computed transition probabilities with a space-saving algorithm [111] to improve the running time of GCFEXPLAINER.

# 3.4 Experiments

We provide empirical results for the proposed GCFExplanier along with baselines on commonly used graph classification datasets. Our code is available at https:// github.com/mertkosan/GCFExplainer.

	NCI1	Mutagenicity	AIDS	Proteins
#Graphs	3978	4308	1837	1113
#Nodes	118714	130719	28905	43471
#Edges	128663	132707	29985	81044
#Node Labels	10	10	9	3

Table $3.1$ :	The	statistics	of the	datasets.
---------------	-----	------------	--------	-----------

## 3.4.1 Experimental Settings

#### Datasets

We use four different real-world datasets for graph classification benchmark with their statistics in Table 3.1. Specifically, NCI1 [112], Mutagenicity [113, 96], and AIDS [113] are collections of molecules with nodes representing different atoms and edges representing chemical bonds between them. The molecules are classified by whether they are anticancer, mutagenic, and active against HIV, respectively. Proteins [85, 114] is a collection of proteins classified into enzymes and non-enzymes, with nodes representing secondary structure elements and edges representing structural proximity. For all datasets, we filter out graphs containing rare nodes with label frequencies smaller than 50.

#### Graph classifier

We follow [88] and train a GNN with 3 convolution layers [13] of embedding dimension 20, a max pooling layer, and a fully connected layer for classification. The

model is trained with the Adam optimizer [115] and a learning rate of 0.001 for 1000 epochs. The datasets are split into 80%/10%/10% for training/validation/testing with the model accuracy shown in Table 3.2.

	NCI1	Mutagenicity	AIDS	Proteins
Training	0.8439	0.8825	0.9980	0.7800
Validation	0.8161	0.8302	0.9727	0.8198
Testing	0.7809	0.8000	0.9781	0.7297

Table 3.2: Accuracy of the GNN graph classifier.

#### Baselines

To the best of our knowledge, GCFEXPLAINER is the first global counterfactual explainer. To validate its effectiveness, we compare it against state-of-the-art local counterfactual explainers combined with the greedy summarization algorithm described in Section 3.3.3. The following local counterfactual generation methods are included in our experiments.

- ▷ GROUND-TRUTH: Using graphs belonging to the desired class from the original dataset as local counterfactuals.
- ▷ RCEXPLAINER [91]: Local counterfactual explainer based on the modeling of implicit decision regions of GNNs.
- ▷ CFF [93]: Local counterfactual explainer based on joint modeling of factual and counterfactual reasoning.

#### Explainer settings

We use a distance threshold  $\theta$  of 0.05 for training all explainers. Since computing the exact graph edit distance is NP-hard, we apply a state-of-the-art neural approximation algorithm [103]. For GCFEXPLAINER, we set the teleportation probability  $\tau = 0.1$  and

tune  $\alpha$ , the weight between individual coverage and gain of coverage, from  $\{0, 0.5, 1\}$ . A sensitivity analysis is presented in Section 3.4.6. The number of VRRW iterations M is set to 50000, which is enough for convergence as shown in Section 3.4.5. For baselines, we tune their hyperparameters to achieve the best local counterfactual rates while maintaining an average distance to input graphs that is smaller than the distance threshold  $\theta$ .

	NC	I1	Mutage	enicity	AII	DS	Prot	eins
	Coverage	$\operatorname{Cost}$	Coverage	Cost	Coverage	$\operatorname{Cost}$	Coverage	$\operatorname{Cost}$
GROUND-TRUTH	16.54%	0.1326	28.96%	0.1275	0.41%	0.2012	8.47%	0.2155
RCEXPLAINER	15.22%	0.1370	31.99%	0.1290	8.96%	0.1531	8.74%	0.2283
CFF	17.61%	0.1331	30.43%	0.1327	3.39%	0.1669	3.83%	0.2557
GCFEXPLAINER	27.85%	0.1281	37.08%	0.1135	14.66%	0.1516	10.93%	0.1856

Table 3.3: Recourse coverage ( $\theta = 0.1$ ) and median recourse cost comparison between GCFEXPLAINER and baselines for a 10-graph global explanation. GCFEXPLAINER consistently and significantly outperforms all baselines across different datasets.

#### 3.4.2 Recourse Quality

We start by comparing the recourse quality between GCFEXPLAINER and baselines. Table 3.3 shows the recourse coverage with  $\theta = 0.1$  and median recourse cost of the top 10 counterfactual graphs (i.e., k = 10). We first notice that the two state-of-the-art local counterfactual explainers have similar performance as GROUND-TRUTH, consistent with our claim that local counterfactual examples from existing methods are not good candidates for a global explanation. The proposed GCFEXPLAINER, on the other hand, achieves significantly better performance for global recourse quality. Compared to the best baseline, RCEXPLAINER, GCFEXPLAINER realizes a 46.9% gain in recourse coverage and a 9.5% reduction in recourse cost.

Next, we show the recourse coverage and cost for different sizes of counterfactual summary in Figure 3.3. As expected, adding more graphs to the recourse representation



Figure 3.3: Coverage and cost performance comparison between GCFEXPLAINER and baselines based on different counterfactual summary sizes. GCFEXPLAINER consistently outperforms the baselines across different sizes.

increases recourse coverage while decreasing recourse cost, at the cost of interpretability. And GCFEXPLAINER maintains a constant edge over the baselines.

We also compare the recourse coverage based on different distance thresholds  $\theta$ , with results shown in Figure 3.4. While coverage increases for all methods as the threshold increases, GCFEXPLAINER consistently outperforms the baselines across different thresholds.



Figure 3.4: Recourse coverage comparison between GCFEXPLAINER and baselines based on different distance threshold values ( $\theta$ ). GCFEXPLAINER consistently outperforms the baselines across different  $\theta$ .

## 3.4.3 Global Counterfactual Insight

We have demonstrated the superiority of GCFEXPLAINER based on various quality metrics for global recourse. Here, we show how GCFEXPLAINER provides global insights compared to local counterfactual examples. Figure 3.5 illustrates (a) four input undesired graphs with a similar structure from the AIDS dataset, (b) corresponding local counterfactual examples (based on RCEXPLAINER and CFF), and (c) the representative global counterfactual graph from GCFEXPLAINER covering the input graphs. Our goal is to understand why the input graphs are inactive against AIDS (undesired) and how to obtain the desired property with minimal changes.

The local counterfactuals in (b) attribute the classification results to different edges in individual graphs (shown as red dotted lines) and recommend their removal to make input graphs active against HIV. Note that while only two edits are proposed for each individual graph, they appear at different locations, which are hard to generalize for a global view of the model behavior. In contrast, the global counterfactual graph from GCFEXPLAINER presents a high-level recourse rule. Specifically, the carbon atom with the carbon-oxygen bond is connected to *two* other carbon atoms in the input graphs, making them ketones (with a C=O bond) or ethers (with a C-O bond). On the other hand, the global counterfactual graph highlights a different functional group, aldehyde (shown in blue), to be the key for combating AIDS. In aldehydes, the carbon atom with a carbon-oxygen bond is only connected to *one* other carbon atom, leading to different chemical properties compared to ketones and ethers. Indeed, aldehydes have been shown to be effective HIV protease inhibitors [116].

Finally, this case study also demonstrates that counterfactual candidates found by GCFEXPLAINER are better for global explanation than local counterfactuals. We note that while the graph edit distance between the local counterfactuals and their corresponding input graphs is only 2, they do not cover other similarly structured input graphs (with distance > 5). Meanwhile, our global counterfactual graph covers all input graphs (with distance  $\leq 4$ ).



Figure 3.5: Illustration of global and local counterfactual explanations for the AIDS dataset. The global counterfactual graph (c) presents a high-level recourse rule—changing ketones and ethers into aldehydes (shown in blue)—to combat HIV, while the edge removals (shown in red) recommended by local counterfactual examples (b) are hard to generalize.

# 3.4.4 Ablation Study

We then conduct an ablation study to investigate the effectiveness of GCFEx-PLAINER components. We consider three alternatives:

- $\triangleright$  GCFEXPLAINER-NVR: no vertex-reinforcement (N(v) = 1)
- $\triangleright$  GCFEXPLAINER-NIF: no importance function (I(v) = 1)

 $\triangleright$  GCFEXPLAINER-NDT: no dynamic teleportation  $(p_{\tau}(G) = 1/|\mathbb{G}|)$ 

The coverage results are shown in Table 3.4. We observe decreased performance when any of GCFEXPLAINER components is absent.

	NCI1	Mutagenicity	AIDS	Proteins
GCFExplainer-NVR	24.56%	35.44%	11.33%	8.56%
GCFExplainer-NIF	13.29%	29.16%	4.54%	6.83%
GCFExplainer-NDT	27.34%	36.35%	14.05%	9.28%
GCFEXPLAINER	$\mathbf{27.85\%}$	$\mathbf{37.08\%}$	14.66%	10.93%

Table 3.4: Ablation study results based on recourse coverage.

## 3.4.5 Convergence Analysis

In this subsection, we show the empirical convergence of VRRW based on the Mutagenicity dataset in Figure 3.6. We observe that the coverage performance for different summary sizes starts to converge after 15000 iterations and fully converges after 50000 iterations, which is the number we applied in our experiments.



Figure 3.6: Convergence of VRRW for the Mutagenicity dataset based on recourse coverage with different summary sizes. VRRW fully converges after M = 50000 iterations.

	NCI1	Mutagenicity	AIDS	Proteins
$\alpha = 0.0$	27.85%	36.87%	12.83%	10.11%
$\alpha = 0.5$	27.50%	36.59%	14.66%	10.38%
$\alpha = 1.0$	22.27%	$\mathbf{37.08\%}$	13.99%	10.93%

Table 3.5: Sensitivity analysis on  $\alpha$ , the weight between individual coverage and gain of coverage in the importance function.

#### 3.4.6 Sensitivity Analysis

The only hyperparameter of GCFEXPLAINER we tune is  $\alpha$  in Equation 3.3 that weights the individual coverage and gain of coverage for the importance function. Table 3.5 shows the results based on different  $\alpha$ . While GCFEXPLAINER outperforms baselines with all different  $\alpha$ , we observe that individual coverage works better for NCI1 and gain of cumulative coverage works better for other datasets.

	NCI1	Mutagenicity	AIDS	Proteins
RCEXPLAINER	30454	52549	29047	8444
$\operatorname{CFF}$	22794	31749	21296	$\boldsymbol{6412}$
GCFExplainer	19817	24006	2615	19246
GCFExplainer-S	19365	18798	2539	<u>7429</u>

Table 3.6: Counterfactual candidates generation time comparison. GCFEXPLAINER (-S) has competitive running time albeit exploring more counterfactual graphs.

# 3.4.7 Running Time

Table 3.6 summarizes the running times of generating counterfactual candidates based on different methods. GCFEXPLAINER has a competitive running time albeit exploring more counterfactual graphs in the process. We also include results for GCFEXPLAINER-S which samples a maximum of 10000 neighbors for computing the importance at each step. It achieves better running time at a negligible cost of 3.3% performance loss on average. Finally, summarizing the counterfactual candidates takes less than a second for all methods.

# 3.5 Related Work

Explanations for Graph Neural Networks. There is much research [65, 87, 88, 89] on explaining graph neural networks (GNNs). The first proposed method, GNNExplainer [65], finds the explanatory subgraph and sub-features by maximizing the mutual information between the original prediction and the prediction based on the subgraph and sub-features. Later, PGExplainer [87] provides an inductive framework that extracts GNN node embeddings and learns to map embedding pairs to the probability of edge existence in the explanatory weighted subgraph. PGMExplainer [88] builds a probabilistic explanation model that learns new predictions from perturbed node features, performs variable selection using Markov blanket of variables, and then produces a Bayesian network via structure learning. In XGNN [89], the authors find model-level explanations by a graph generation module that outputs a sequence of edges using reinforcement learning. These explanation methods focus on *factual* reasoning while the goal of our work is to provide a global *counterfactual* explanation for GNNs.

Counterfactual Explanations. Recently, there are several attempts to have explanations of graph neural networks (GNNs) via counterfactual reasoning [90, 91, 92, 93]. One of the earlier methods, CF-GNNExplainer [90], provides counterfactual explanations in terms of a learnable perturbed adjacency matrix that leads to the flipping of classifier prediction for a node. On the other hand, RCExplainer [91] aims to find a robust subset of edges whose removal changes the prediction of the remaining graph by modeling the implicit decision regions based on GNN graph embeddings. In [92], the authors investigate counterfactual explanations for a more specific class of graphs—the brain networks—that share the same set of nodes by greedily adding or removing edges using a heuristic. More recently, the authors of CFF [93] argue that a good explanation for GNNs should consider both factual and counterfactual reasoning and they explicitly incorporate those objective functions when searching for the best explanatory subgraphs and sub-features. Counterfactual reasoning has also been applied for link prediction [117]. All the above methods produce *local* counterfactual examples while our work aims to provide a *global* explanation in terms of a summary of representative counterfactual graphs.

# 3.6 Conclusions

We have proposed GCFEXPLAINER, the first global counterfactual explainer for graph classification. Compared to local explainers, GCFEXPLAINER provides a highlevel picture of the model behavior and effective global recourse rules. We hope that our work will not only deepen our understanding of graph neural networks but also build a bridge for experts from other domains to leverage deep learning models for high-stakes decision-making.

**Future Works:** We have identified two weaknesses in our framework. Firstly, GCFEX-PLAINER occasionally generates invalid molecules in the summary (in terms of valence), rendering the summary graphs inapplicable in certain domains. To address this issue, we propose incorporating fragment-based editing during the random walk process. This approach aims to enhance the generation of valid molecules in the summaries, ensuring their relevance and applicability.

Secondly, our algorithm is currently transductive, meaning that when a new graph emerges, the entire algorithm needs to be rerun. To mitigate this limitation, we plan to leverage molecule generation models from existing literature. By incorporating such models, we anticipate the ability to generate new valid graphs on the fly, eliminating the need for rerunning the entire algorithm when faced with new data.

To address the issue of molecular validity in our framework, we propose two approaches. The first approach, referred to as **Post-checking**, involves checking the validity of each generated molecule after the summary generation process. However, it should be noted that this method may result in a decrease in the number of counterfactual graphs present in the summary set.

Alternatively, we can modify the importance function used during the random walk to ensure that only valid graphs are traversed. The updated importance function, referred to as **In-checking**, is defined as follows:

$$I(v) = \begin{cases} p(\phi(v))(\alpha \text{ coverage}(v) + (1 - \alpha) \text{ gain}(v)) & f(v) = 1\\ 0 & f(v) = 0 \end{cases}$$
(3.5)

Here, f(v) represents the validity of the molecule, where f(v) = 1 indicates that v is a legal molecule. By incorporating the validity function within the importance function, the random walk will prioritize valid graphs, ensuring that only valid molecules are included in the summary.

During our experiments, we observed that only a limited number of editing operations resulted in new valid molecules, while most operations led to valence violations. To enhance search efficiency, we propose transitioning from atom-based editing to fragmentbased editing, which focuses exclusively on adding or deleting legal fragments.

To construct a fragment vocabulary, we follow the approach outlined in [118], which involves breaking every single bond in the molecules within the dataset and considering the resulting smaller arms as separate fragments. During the random walk, we implement fragment addition and removal for the current node. Additionally, we enumerate node positions to insert the fragment and test all fragments present in the vocabulary. For removal, we break single bonds and delete the smaller fragment.

By incorporating fragment-based editing, we strive to enhance the search efficiency of our framework while prioritizing the generation of valid molecules and avoiding valence violations. Table 3.7 presents the results obtained from the **Post-checking**, **Inchecking**, and **Fragment-based** methods on the Mutagenicity dataset. Comparing these methods to the original GCFExplainer, we observe a degradation in coverage and cost performance, while the validity of generated molecules improves. Notably, the Mutagenicity dataset exhibits a small fragment vocabulary size, indicating a prevalence of shared patterns among the graphs. Consequently, fragments extracted from one graph are likely to be valuable for generating valid molecules in another graph. As a result, the fragment-based method proves to be more efficient compared to the other two checking methods.

	Mutagenicity (vocab=408)			
	$\overrightarrow{\text{Coverage}} \uparrow$	$\mathrm{Cost}\downarrow$	#Valid / #Total	Time (')
GCFExplainer	38.52%	11.37	/	/
Post-checking	32.36%	12.84	883/2438	258
In-checking	34.95%	12.23	2438/2438	242
Fragment-based	35.32%	12.15	2438/2438	229

Table 3.7: Fragment-based editing enhances the search efficiency and promotes valid molecule generation. Despite slightly lower coverage and cost, the validity of molecules improves due to shared patterns and the efficiency of fragment utilization. Time is calculated based on minutes.

To address the limitation of the fragment-based method in generating a summary set for unseen graphs, we aim to make our framework more inductive. One approach is to explore molecule generation methods, such as the one proposed by Kong et al. [119], and optimize them using our objective function, specifically the importance function. As our objective function is non-differentiable, we are investigating solutions from reinforcement learning to optimize our framework.

Our proposed approach involves developing an end-to-end framework that initially generates molecules for input graphs. These generated molecules will then be optimized using the same importance function, which considers factors such as being counterfactual to the input graphs, high coverage, and/or low cost. By integrating reinforcement learning techniques, we aim to enhance the overall performance of our framework, enabling it to generate optimized molecules that align with our defined objectives.

Furthermore, we aim to incorporate a human-in-the-loop mechanism to leverage expert feedback in order to enhance the quality of the generated explanations. This is crucial because the effectiveness of the explanations heavily relies on the accuracy of the underlying graph classifier, which may not always be optimal. By obtaining additional feedback from domain experts, we can iteratively improve the quality and relevance of the explanations, thereby enhancing the overall performance of our framework.

# Chapter 4

# Robust Ante-hoc Graph Explainer using Bilevel Optimization

# 4.1 Introduction



Figure 4.1: Explanations generated by our approach (RAGE) in two case studies: PLANTED CLIQUE (graphs with and without cliques) and SUNGLASSES (headshots with and without sunglasses). RAGE explanations identify edges in the clique and in the region around the sunglasses for both difficulties. For PLANTED CLIQUE (a), the heatmap and sizes show node and edge influences, and the nodes with a thick border are members of the planted clique. For SUNGLASSES (b-c), the red dots show the influential pixel connections to the detection.

A critical problem in machine learning on graphs is understanding predictions made by graph-based models in high-stakes applications. This has motivated the study of graph explainers, which aim to identify subgraphs that are both compact and correlated with model decisions. However, there is no consensus on what constitutes a good explanation—i.e. correlation metric. Recent papers [65, 87, 91] have proposed different alternative notions of explainability that do not take the user into consideration and instead are validated using examples. On the other hand, other approaches have applied labeled explanations to learn an explainer directly from data [120]. However, such labeled explanations are hardly available.

Explainers can be divided into *post-hoc* and *ante-hoc* (or intrinsic) [121]. Post-hoc explainers treat the prediction model as a black box and learn explanations by modifying the input of a pre-trained model [122]. On the other hand, ante-hoc explainers learn explanations as part of the model. The key advantage of post-hoc explainers is flexibility since they make no assumption about the prediction model to be explained or the training algorithm applied to learn the model. However, these explanations have two major limitations: (1) they are not sufficiently informative to enable the user to reproduce the behavior of the model, and (2) they are often based on a model that was trained without taking explainability into account.

The first limitation is based on the intuitive assumption that a good explanation should enable the user to approximately reproduce the decisions of the model for new input. That is simply because the predictions will often depend on parts of the input that are not part of the explanation. The second limitation is based on the fact that for models with a large number of parameters, such as neural networks, there are likely multiple parameter settings that achieve similar values of the loss function. However, only some of such models might be explainable [123, 124]. While these limitations do not necessarily depend on a specific model, this paper addresses them in the context of Graph Neural Networks for graph-level tasks (classification and regression).

We propose RAGE—a novel ante-hoc explainer for graphs—that aims to find compact explanations while maximizing the graph classification/regression accuracy using bilevel optimization. Figure 4.2 compares the post-hoc and ante-hoc approaches in the context of graph classification. RAGE explanations are given as input to the GNN, which guarantees that no information outside of the explanation is used for prediction. This enables the user to select an appropriate trade-off between the compactness of the explanations and their discrimination power. We show that RAGE explanations are more robust to noise in the input graph than existing (post-hoc and ante-hoc) alternatives. Moreover, our explanations are learned jointly with the GNN, which enables RAGE to learn GNNs that are accurate and explainable. In fact, we show that RAGE's explainability objective produces an inductive bias that often improves the accuracy of the learned GNN compared to the base model. We emphasize that while RAGE is an ante-hoc model, it is general enough to be applied to a broad class of GNNs.

Figure 4.1 shows examples of RAGE explanations in two case studies. In 4.1a, we show an explanation from a synthetic dataset (PLANTED CLIQUE), where the goal is to classify whether the graph has a planted clique or not based on examples. As expected, the edge influences learned by RAGE match with the planted clique. In 4.1b, we show an explanation for a real dataset (SUNGLASSES) with graphs representing headshots (images), where the goal is to classify whether the person in the corresponding headshot is wearing sunglasses. We notice that edge influences highlight pixels around the sunglasses. We will evaluate our approach quantitatively in terms of accuracy, reproducibility, and robustness. Our results show that RAGE often outperforms several baselines. Our main contributions can be summarized as follows:

▷ We highlight and empirically demonstrate two important limitations of post-hoc graph explainers. They do not provide enough information to enable reproducing the be-



Figure 4.2: (a) Post-hoc models generate explanations for a pre-trained GNN classifier using its predictions. (b) Ante-hoc models, as our approach, learn GNNs and explanations jointly. This enables ante-hoc models to identify GNNs that are both explainable and accurate.

havior of the predictor and are based on fixed models that might be accurate but not explainable.

- ▷ We propose RAGE, a novel GNN and flexible explainer for graph classification and regression tasks. RAGE applies bilevel optimization, learning GNNs in the inner problem and an edge influence function in the outer loop. Our approach is flexible enough to be applied to a broad class of GNNs.
- ▷ We will compare RAGE against state-of-the-art graph classification and GNN explainer baselines using six datasets—including five real-world ones. Our initial preliminary results show that RAGE generates explanations that are robust, and enable reproducing the behavior of the predictor. We also provide case studies showing that our method improves interpretability by highlighting essential parts of the input data.

# 4.2 Related Work

**Graph classification with GNNs:** Graph Neural Networks (GNNs) have gained prominence in graph classification due to their ability to learn features directly from data [13, 47, 125, 126]. GNN-based graph classifiers aggregate node-level representations via pooling operators to represent the entire graph. The design of effective graph pooling operators is key for effective graph classification [55, 17, 56, 127]. However, simple pool-

ing operators that disregard the graph structure, such as mean and max, remain popular and have been shown to have comparable performance to more sophisticated alternatives [128]. Recently, [129] proposed a multi-head attention pooling layer to capture structural dependencies between nodes. In this paper, we focus on graph classification and regression tasks.

**Explainability of GNNs:** Explainability has become a key requirement for the application of machine learning in many settings (e.g., healthcare, court decisions) [130]. Several post-hoc explainers have been proposed for explaining Graph Neural Networks' predictions using subgraphs [65, 87, 131, 91, 93, 132]. GNNExplainer [65] applies a meanfield approximation to identify subgraphs that maximize the mutual information with GNN predictions. PGExplainer [87] applies a similar objective, but samples subgraphs using the reparametrization trick. RCExplainer [91] identifies decision regions based on graph embeddings that generate a subgraph explanation such that removing it changes the prediction of the remaining graph (i.e., counterfactual). While post-hoc explainers treat a trained GNN as a black box — i.e., it only relies on predictions made by the GNN ante-hoc explainers are model-dependent. GIB [133] applies the *bottleneck principle* and bilevel optimization to learn subgraphs relevant for classification but different from the corresponding input graph. ProtGNN [124] learns prototypes (interpretable subgraphs) for each class and makes predictions by matching input graphs and class prototypes. Bilevel optimization and prototypes help in the generalizability of explanations. Our preliminary results show that RAGE explanations are more meaningful and robust than alternatives and can reproduce the model behavior better than existing post-hoc and ante-hoc explainers.

**Bilevel optimization:** Bilevel optimization is a class of optimization problems where two objective functions are nested within each other [134]. Although the problem is known to be NP-hard, recent algorithms have enabled the solution of large-scale problems in machine learning, such as automatic hyperparameter optimization and meta-learning [135]. Bilevel optimization has recently also been applied to graph problems, including graph meta-learning [136] and transductive graph sparsification scheme [137]. Like RAGE, GIB [133] also applies bilevel optimization to identify discriminative subgraphs inductively. However, we show that our approach consistently outperforms GIB in terms of discriminative power, reproducibility, and robustness.

**Graph structure learning:** Graph structure learning (GSL) aims to enhance (e.g., complete, de-noise) graph information to improve the performance of downstream tasks [138]. LDS-GNN [139] applies bilevel optimization to learn the graph structure that optimizes node classification. VIB-GSL [140] advances GIB [133] by applying a variational information bottleneck on the entire graph instead of only edges. We notice that GSL mainly focuses on learning the entire graph, whereas we only sparsify the graph, which reduces the search space and is more interpretable than possibly adding new edges. Furthermore, learning the entire graph is not scalable in large graph settings.

# 4.3 Methodology

#### 4.3.1 Problem Formulation

We formulate our problem as a supervised graph classification (or regression). Given a graph set  $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$  and continuous or discrete labels  $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ for each graph respectively, our goal is to learn a function  $\hat{f} : \mathcal{G} \to \mathcal{Y}$  that approximates the labels of unseen graphs.



Figure 4.3: Illustration of an edge-based ante-hoc explainer that uses bilevel optimization. Explainer generates an explanation graph from the input graph by assigning an influence value to each edge. Edge influences are incorporated to edge weights on the explanation graph, the input of GNN Classifier. The inner problem optimizes GNN Classifier with T iterations, while the outer problem updates Explainer using gradients from inner iterations. The dotted edges in the explanation graph show that they do not influence the classification, while others have different degrees of influence.

## 4.3.2 RAGE: Robust Ante-hoc Graph Explainer

We introduce RAGE, an ante-hoc explainer that generates robust explanations using bilevel optimization. RAGE performs compact and discriminative subgraph learning as part of the GNN training that optimizes the prediction of class labels in graph classification or regression tasks.

RAGE is based on a general scheme for an edge-based approach for learning ante-hoc explanations using bilevel optimization, as illustrated in Figure 4.3. The explainer will assign an influence value to each edge, which will be incorporated into the original graph. The GNN classifier is trained with this new graph over T inner iterations. Gradients from inner iterations are kept to update the explainer in the outer loop. The outer iterations minimize a loss function that induces explanations to be compact (sparse) and discriminative (accurate). We will now describe our approach (RAGE) in more detail.

#### Explainer - Subgraph Learning

RAGE is an edge-based subgraph learner. It learns edge representations from the node representations/features. Surprisingly, most edge-based explainers for undirected graphs are not permutation invariant when calculating edge representations. Shuffling nodes could change their performance drastically since the edge representations would differ. We calculate permutation invariant edge representations  $h_{ij}$  given two node representations  $h_i$  and  $h_j$  as follows:  $h_{ij} = [\max(h_i, h_j); \min(h_i, h_j)]$ , where **max** and **min** are pairwise for each dimension and  $[\cdot; \cdot]$  is the concatenation operator.

Edge influences are learned via an MLP with sigmoid activation based on edge representations:  $z_{ij} = MLP(h_{ij})$ . This generates an edge influence matrix  $Z \in [0, 1]^{nxn}$ . We denote our explainer function as  $g_{\Phi}$  with trainable parameters  $\Phi$ .

#### Influence-weighted Graph Neural Networks

Any GNN architecture can be made sensitive to edge influences Z via a transformation of the adjacency matrix of the input graphs. As our model does not rely on a specific architecture, we will refer to it generically as GNN(A, X), where A and X are the adjacency and attribute matrices, respectively. We rescale the adjacency matrix with edge influences Z as follows:  $A_Z = Z \odot A$ .

The GNN treats  $A_Z$  in the same way as the original matrix:  $H = GNN(A_Z, X)$ 

We generate a graph representation h from the node representation matrix H via a max pooling operator. The graph representation h is then given as input to a classifier that will predict graph labels y. Here, we use an MLP as our classifier.

#### **Bilevel Optimization**

In order to perform both GNN training and estimate the influence of edges jointly, we formulate graph classification as a bilevel optimization problem. In the inner problem (Equation 4.2), we learn the GNN parameters  $\theta^* \in \mathbb{R}^h$  given edges influences  $Z^* \in$  $[0,1]^{nxn}$  based on a training loss  $\ell^{tr}$  and training data  $(D^{tr}, y^{tr})$ . We use the symbol C to refer to any GNN architecture. In the outer problem (Equation 4.1), we learn edge influences  $Z^*$  by minimizing the loss  $\ell^{sup}$  using support data  $(D^{sup}, y^{sup})$ . The loss functions for the inner and outer problem,  $f_{Z^*}$  and F, also apply regularization functions,  $\Theta_{inner}$  and  $\Theta_{outer}$ , respectively.

$$Z^* = \underset{Z}{\operatorname{arg\,min}} F(\theta^*, Z) = \ell^{sup}(C(\theta^*, Z, D^{sup}), y^{sup}) + \Theta_{outer}$$
(4.1)

$$\theta^* = \underset{\theta}{\operatorname{arg\,min}} f_{Z^*}(\theta) = \ell^{tr}(C(\theta, Z^*, D^{tr}), y^{tr}) + \Theta_{inner}$$
(4.2)

RAGE can be understood through the lens of meta-learning. The outer problem performs *meta-training* and edge influences are learned by a *meta-learner* based on support data. The inner problem solves multiple *tasks* representing different training splits sharing the same influence weights.

At this point, it is crucial to justify the use of bilevel optimization to compute edge influences Z. A simpler alternative would be computing influences as edge attention weights using standard gradient-based algorithms (i.e., single-level). However, we argue that bilevel optimization is a more robust approach to our problem. More specifically, we decouple the learning of edge influences from the GNN parameters and share the same edge influences in multiple training splits. Consequently, these influences are more likely to generalize to unseen data. We validate this hypothesis empirically using different datasets in our experiments (Sec. 4.6.1).

#### Loss Functions

RAGE loss functions have two main terms: a prediction loss and a regularization term. As prediction losses, we apply cross-entropy or mean-square-error, depending on whether the problem is classification or regression. The regularization for the inner problem  $\Theta_{inner}$  is a standard  $L_2$  penalty over the GNN weights  $\theta$ . For the outer problem  $\Theta_{outer}$ , we also apply an  $L_1$  penalty to enforce the sparsity of Z. Finally, we also add an  $L_2$  penalty on the weights of  $g_{\Phi}$ .

#### 4.3.3 Bilevel Optimization Training

The main steps performed by our model (RAGE) are given in Algorithm 3. For each outer iteration (lines 1-14), we split the training data into two sets—training and support—(line 2). First, we use training data to calculate  $Z^{tr}$ , which is used for GNNtraining in the inner loop (lines 5-10). Then, we apply the gradients from the inner problem to optimize the outer problem using support data (lines 11-13). Note that we reinitialize GNN and MLP parameters (line 4) before starting inner iterations to remove undesirable information [141] and improve data generalization [142]. We further discuss the significance and the impact of this operation in Sec. 4.6.2. The main output of our algorithm is the explainer  $g_{\Phi_{\kappa}}$ . Moreover, the last trained  $GNN_{\theta_T}$  can also be used for the classification of unseen data, or a new GNN can be trained based on Z. In both cases, the GNN will be trained with the same input graphs, which guarantees the behavior of the model  $GNN_{\theta_T}$  can be reproduced using explanations from  $g_{\Phi_{\kappa}}$ .

For gradient calculation, we follow the gradient-based approach described in [143]. The critical challenge of training our model is how to compute gradients of our outer objective with respect to edge influences Z. By the chain rule, such gradients depend on the gradient of the classification/regression training loss with respect to Z. We will,

again, use the connection between RAGE and meta-learning to describe the training algorithm.

#### Training (Inner Loop)

At inner loop iterations, we keep gradients while optimizing model parameters  $\theta$ .

$$\theta_{t+1} = inner-opt_t \left(\theta_t, \nabla_{\theta_t} \ell^{tr}(\theta_t, Z_\tau)\right)$$

After T iterations, we compute  $\theta^*$ , which is a function of  $\theta_1, \ldots, \theta_T$  and  $Z_{\tau}$ , where  $\tau$  is the number of iterations for meta-training. Here, *inner-opt*<sub>t</sub> is the inner optimization process that updates  $\theta_t$  at step t. If we use SGD as an optimizer, *inner-opt*<sub>t</sub> will be written as follows with a learning rate  $\eta$ :

inner-opt<sub>t</sub> 
$$(\theta_t, \nabla_{\theta_t} \ell^{tr}(\theta_t, Z_\tau)) \coloneqq \theta_t - \eta \cdot \nabla_{\theta_t} \ell^{tr}(\theta_t, Z_\tau)$$

#### Meta-training (Outer Loop)

After T inner iterations, the gradient trajectory saved to  $\theta^*$  will be used to optimize  $\Phi$ . We denote *outer-opt*<sub>\tau</sub> as outer optimization that updates  $\Phi_{\tau}$  at step  $\tau$ . The meta-training step is written as:

$$\Phi_{\tau+1} = outer-opt_{\tau} \left( \Phi_{\tau}, \nabla_{\Phi_{\tau}} \ell^{sup}(\theta^*) \right)$$
  
=  $outer-opt_{\tau} \left( \Phi_{\tau}, \nabla_{\Phi_{\tau}} \ell^{sup}(inner-opt_T(\theta_T, \nabla_{\theta_T} \ell^{tr}(\theta_T, z_{\tau}))) \right)$ 

After each meta optimization step, we calculate edge influences  $Z_{\tau+1}$  using  $g_{\Phi_{\tau+1}}(.)$ . Notice that our training algorithm is more computationally-intensive than training a simple GNN architecture. Therefore we set T and  $\kappa$  to small values. Thus, RAGE can

#### Algorithm 3 RAGE

**Require:** Graphs  $A_{1:n}$ , node attributes  $X_{1:n}$ , labels  $y_{1:n}$ , explainer  $g_{\Phi_0}$ , outer/inner loops  $\kappa$  and T

**Ensure:** Trained  $q_{\Phi_r}$ 1: for  $\tau \in [0, \kappa - 1]$  do  $A^{tr}, A^{sup}, X^{tr}, X^{sup}, y^{tr}, y^{sup} \leftarrow \operatorname{split}(A_{1:n}, X_{1:n}, y_{1:n})$ 2:  $Z^{tr} \leftarrow g_{\Phi_{\tau}}(A^{tr}, X^{tr})$ 3: (re)initialize  $GNN_{\theta_0}$ ,  $MLP_{\theta_0}$ 4: for  $t \in [0, T - 1]$  do 5:  $H^{tr} \leftarrow GNN_{\theta_t}(Z^{tr} \odot A^{tr}, X^{tr})$ 6:  $h^{tr} \leftarrow POOL_{max}(H^{tr})$ 7:  $p^{tr} \leftarrow MLP_{\theta_t}(h^{tr})$ 8:  $GNN_{\theta_{t+1}}, MLP_{\theta_{t+1}} \leftarrow inner-opt \ f_{Z^{tr}}(p^{tr}, y^{tr})$ 9: end for 10:  $Z^{sup} \leftarrow q_{\Phi_{\tau}}(A^{sup}, X^{sup})$ 11:  $p^{sup} \leftarrow MLP_{\theta_T}(POOL_{max}(GNN_{\theta_T}(Z^{sup} \odot A^{sup}, X^{sup})))$ 12: $g_{\Phi_{\tau+1}} \leftarrow outer\text{-}opt \ F_{\theta_{\tau}}(p^{sup}, y^{sup})$ 13:14: **end for** 15: return  $g_{\Phi_{\kappa}} = 0$ 

also be efficiently applied at training and testing times.

# 4.4 Experiments

We evaluate RAGE on several datasets, outperforming both post-hoc and ante-hoc explainers in terms of metrics, including discriminative power and robustness. A key advantage of our approach is being able to efficiently search for a GNN that is both explainable and accurate. Our case studies show the effectiveness of explanations generated by RAGE over the baselines. We also provide more results and analysis on RAGE. Our implementation of RAGE is available at https://anonymous.4open.science/r/RAGE/.

#### 4.4.1 Experimental Settings

**Datasets:** We consider six graph classification (regression) datasets in our experiments. Table 4.1 shows the main statistics of them. MUTAGENICITY [113, 96], PRO-TEINS [85, 114], IMDB-B [144] are graph classification datasets. We assign one-hot node degrees as features for IMDB-BINARY, which originally does not have node features. SUNGLASSES [145] is a dataset created from the CMU face dataset, where nodes are pixels, edges connect nearby pixels, node attributes are pixel (gray-scale) colors, and labels indicate whether the person in the picture wears sunglasses. TREE OF LIFE [146] is a collection of Protein-Protein Interaction (PPI) networks with amino-acid sequence embeddings [147] as node features and (real) evolution scores as graph labels. PLANTED CLIQUE are Erdős–Rényi (ER) graphs (edge probability of 0.1) and add a planted clique of size k (class 1) to half of them (others assigned to class 0). We select the value of k to be larger than the size of the largest clique in the ER graphs. We also consider a noisy version of MUTAGENICITY, which we call MUTAGENICITYNOISYX, where the noise  $X \in \{1, 2, 4, 8, 16, 32\}$  corresponds to the number of i.i.d. random edges added to each graph.

Table 4.1: The statistics of the datasets.					
#Graphs	#Nodes	#Edges	#Features		
4337	131488	133447	14		
1113	43471	81044	32		
1000	19773	96531	136		
624	2396160	9353760	1		
1245	944888	5634922	64		
100	10000	49860	64		
	The statistic:           #Graphs           4337           1113           1000           624           1245           100	$\begin{array}{c c} \hline \  \  \  \  \  \  \  \  \  \  \  \  \$	The statistics of the datasets.#Graphs#Nodes#Edges4337131488133447111343471 $81044$ 100019773 $96531$ 6242396160 $9353760$ 1245 $944888$ $5634922$ 10010000 $49860$		

**Baselines:** We consider several classical baselines for graph classification and regression including GCN [13], GAT [47], GIN [126], and SortPool [17]). We also compare RAGE against ExpertPool [56], which learns attention weights for node pooling, and

against state-of-the-art GNN-based methods such as DropGNN [148], GMT [129], GIB [133], ProtGNN [124], and those that apply graph structure learning including LDS-GNN [139] and VIB-GSL [140]. For methods that use node classification, we add an additional pooling layer to adapt to the graph classification setting. Finally, we consider inductive GNN explainers: PGExplainer [87], RCExplainer [91].

**Evaluation metrics:** We compare the methods in terms of accuracy using AUC for classification and also MSE for regression. Moreover, we compare explanations in terms of *stability* [149], and *reproducibility*. Stability (or robustness) quantifies the correlation between explanations generated for the original dataset and its noisy variants. Reproducibility assesses the accuracy of a GNN trained solely using the explanations as a dataset.

Machine Settings: We run and test our experiments on a machine with NVIDIA GeForce RTX 2080 GPU (8GB of RAM) and 32 Intel Xeon CPUs (2.10GHz and 128GB of RAM).

Hyperparameters: We tune the hyperparameters of our methods and baselines with a grid search. Adam optimizer with a learning rate of 0.001 works well in practice for our inner and outer optimization. T and  $\kappa$  (number of inner and outer epochs) are set to 20 and 100 with early stopping, respectively. We choose 0.001 for regularization weights for inner and outer problem regularization. If the focus is only on graph classification performance, sparsity regularization weight can be set to 0.

Model selection: We model g as a combination of a 3-layer GCN and a 1-layer MLP with sigmoid activation function. For the inner GNN, we use a 3-layer influenceweighted GCN (see Section 4.3.2) and a 1-layer MLP in our experiments. The size of the embeddings is fixed at 20. For all graph classification and explainer baselines, we usually set the GCN layers with the same settings as ours in order to have a fair comparison (unless there are special circumstances of the baseline methods). We changed the baselines' loss functions to mean squared error for Tree of Life datasets and removed the sigmoid activation function MLP.

**Dataset splits:** We use 80% training, 10% validation, and 10% testing splits in our experiments. For our method, we further split the training set into 50% training and 50% support sets. We run each method 20 times and report the average accuracy and standard deviation.

# 4.5 Preliminary Results

Classification (AUC $\%$ )						Regression (MSE)
	MUTAGENICITY	Proteins	IMDB-B	Sunglasses	Planted Clique	TREE OF LIFE
GCN [13]	$86.82\pm0.39$	$82.71 \pm 1.08$	$81.49 \pm 1.16$	$79.66 \pm 13.12$	$52.89 \pm 15.87$	$0.222\pm0.010$
GAT [47]	$86.05\pm0.59$	$82.31 \pm 1.70$	$80.66 \pm 1.56$	$67.80 \pm 15.37$	$51.78 \pm 22.36$	$0.179 \pm 0.018$
GIN [126]	$88.15 \pm 0.37$	$82.65 \pm 0.90$	$84.14 \pm 1.20$	$55.25 \pm 4.62$	$87.78 \pm 12.36$	$0.751 \pm 0.584$
ExpertPool [56]	$\overline{86.81\pm0.47}$	$81.33 \pm 1.21$	$83.20\pm0.48$	$\underline{93.68 \pm 1.15}$	$\overline{80.00\pm19.12}$	$0.100\pm0.015$
SortPool [17]	$85.34 \pm 0.64$	$82.76 \pm 1.10$	$80.62 \pm 1.43$	$\overline{93.26\pm2.52}$	$54.44 \pm 26.97$	$\overline{0.098\pm0.014}$
DropGNN [148]	$84.86 \pm 2.11$	$\overline{82.59 \pm 4.13}$	$\textbf{84.73} \pm \textbf{2.00}$	$54.74 \pm 2.30$	$69.66 \pm 12.70$	$2.690 \pm 1.298$
GMT [129]	$86.06 \pm 1.17$	$82.19 \pm 3.13$	$80.82 \pm 1.38$	$52.32 \pm 1.21$	$56.39 \pm 26.64$	$\underline{0.087 \pm 0.004}$
GIB [133]	$85.53 \pm 0.99$	$82.71 \pm 0.95$	$82.21 \pm 2.04$	$61.30 \pm 7.26$	$53.33 \pm 16.33$	$\overline{0.305\pm0.046}$
ProtGNN [124]	$86.72\pm0.62$	$81.21 \pm 2.07$	$82.53 \pm 2.37$	N/S	$57.78 \pm 13.33$	N/A
LDS-GNN [139]	$86.12 \pm 1.50$	$81.73 \pm 1.32$	$81.12 \pm 1.30$	OOM	$54.12 \pm 15.32$	OOM
VIB-GSL [140]	$84.19 \pm 1.10$	$\mathbf{EG}$	$81.11 \pm 1.21$	OOM	$54.44 \pm 12.96$	OOM
RAGE	$89.52 \pm 0.36$	$\textbf{85.20} \pm \textbf{0.93}$	$\underline{84.16\pm0.32}$	$\textbf{99.36} \pm \textbf{0.44}$	$\textbf{97.78} \pm \textbf{4.44}$	$0.073 \pm 0.007$

EG: Exploding Gradient OOM: Out Of Memory N/S: Not Scalable N/A: Not Applicable

Table 4.2: Test scores for graph classification/regression. Best and second-best values are in bold and underlined for each dataset. RAGE achieves the best results on average, outperforming the baselines.

## 4.5.1 Graph Classification and Regression

Table 4.2 shows the graph classification (regression) results in terms of AUC (MSE) for RAGE and the baselines using five real-world and one synthetic dataset. RAGE outperforms the competing approaches in five datasets and has comparable results for IMDB-B.

Each of its baselines has its own drawbacks and performs poorly on (at least) one dataset with the exception of ExpertPool, which has consistent performance across datasets. Still, RAGE outperforms ExpertPool for every dataset by 10.75% on average and by up to 27.2% on Tree of Life. Surprisingly, most baselines achieve poor results for the Sunglasses and Planted Clique datasets, with the best baselines achieving 93.68% (ExpertPool) and 87.78% (GIN) AUC, respectively. This is evidence that existing approaches, including GIB, are not able to effectively identify compact discriminative subgraphs. We also notice that, comparatively, RAGE achieves the best results for real datasets with large graphs (Sunglasses and Tree of Life). Intuitively, these are datasets for which identifying discriminative subgraphs has the highest impact on performance. Additionally, baselines using graph structure learning have poor performance and are not able to scale to large datasets.



Figure 4.4: Reproducibility comparison between methods for different explanation sizes (in percentage) using four datasets. RAGE outperforms or has comparable results to the baselines across different sizes and datasets.

## 4.5.2 Reproducibility

Reproducibility measures how well explanations alone can predict class labels. It is a key property as it allows the user to correlate explanations and predictions without neglecting potentially relevant information from the input. In our evaluation, we vary



Figure 4.5: Pearson correlation (higher is better) between explanations for Mutagenicity and MutagenicityNoisyX generated by RAGE and the baselines. RAGE is significantly more stable than the baselines, with GIB as the best baseline.

the size of the explanations by thresholding edges based on their values. We then train a GNN using only the explanations and labels. We compare RAGE against post-hoc and ante-hoc explainers and the resulting accuracies are shown in Figure 4.4.

## 4.5.3 Robustness

Effective explanations should be robust to noise in the data. We evaluate the robustness of RAGE and the baselines using MUTAGENICITYNOISYX—i.e. noisy versions of MUTAGENICITY with random edges. We discuss results in terms of stability.

Figure 4.5 presents a comparison of explanations obtained from MUTAGENICITY dataset and its noisy variants, evaluated based on Pearson correlation. Our results demonstrate that RAGE outperforms other graph explainers. Furthermore, GIB is identified as a competitive baseline, which is consistent with the reproducibility metric. These findings provide further evidence that RAGE's approach to generalizability

through bilevel optimization contributes to the robustness of explanations. Furthermore, post-hoc explainers, PGExplainer and RCExplainer, are more sensitive to noise.



(a) Original Image





(c) GIB



(b) PGExplainer

(d) RAGE



Figure 4.6: Two examples with different levels of difficulty due to poses from the Sunglasses dataset. RAGE is able to detect the edges between the sunglasses and the human faces, which are the most relevant for the prediction task (i.e., whether the human is wearing sunglasses) for both examples.

## 4.5.4 Case Study

We now provide a case study illustrating the explanations generated by RAGE and compare them against those generated by post-hoc and ante-hoc baselines. The Sunglasses contain ground truth explanations that are quite intuitive, are applied in our case study. Figure 4.6 shows the results for the Sunglasses dataset. Notice that the examples have different levels of difficulty due to the poses in the pictures. RAGE detects edges at the border between the sunglasses and the faces of the humans in both pictures. These are the edges most relevant for the prediction task—i.e. detecting whether the human is wearing sunglasses. Intuitively, the highlighted edges are the most influential ones for the learned GNN to achieve the high accuracy results from Table 4.2. For the easier pose (Figure 4.6b), PGExplainer identifies one of the glasses as an explanation together with other dark pixels in the image. However, for Figure 4.6e, PGExplainer misses the sunglasses completely. GIB explanations do not match the sunglasses in both examples, which explains its poor performance in this dataset (see Table 4.2). We were not able to apply RCExplainer to this dataset due to an out-of-memory error.

# 4.6 Ablation Study

## 4.6.1 Single-level Optimization

We create a single-level version of RAGE (RAGE-single) to test the effectiveness of our bilevel optimization scheme. More specifically, RAGE-single optimizes the explainer and GNN classifier parts in an end-to-end fashion with a single loss function. Table 4.3 shows that RAGE consistently outperforms RAGE-single. The performance gap is more prominent for large (e.g., Tree of Life) datasets. This is expected since large and noisy datasets are more prone to overfitting.

	RAGE	<b>RAGE-single</b>
Mutagenicity	$89.52 \pm 0.36$	$88.79 \pm 0.82$
Proteins	$85.20 \pm 0.93$	$84.05 \pm 1.26$
IMDB-B	$84.16\pm0.32$	$82.26 \pm 1.42$
Sunglasses	$99.36 \pm 0.44$	$95.23 \pm 1.51$
Planted Clique	$97.78 \pm 4.44$	$86.67 \pm 12.96$
Tree of Life (MSE)	$0.0725 \pm 0.0068$	$0.1002 \pm 0.0101$

Table 4.3: Test scores (AUC and MSE) of RAGE and its single-level variant RAGE-single for all datasets. RAGE outperforms RAGE-single in all datasets. Nevertheless, RAGE-single still has consistent performance across datasets.

	RAGE	RAGE-keep
Mutagenicity	$89.52 \pm 0.36$	$85.76\pm0.77$
Proteins	$85.20\pm0.93$	$84.47\pm0.68$
IMDB-B	$84.16\pm0.32$	$83.46 \pm 1.20$
Sunglasses	$99.36 \pm 0.44$	$70.66 \pm 6.88$
Planted Clique	$97.78 \pm 4.44$	$97.78 \pm 4.44$
Tree of Life (MSE)	$0.0725 \pm 0.0068$	$0.0912 \pm 0.0056$

Table 4.4: Test scores (AUC and MSE) of RAGE and its without reinitialization variant RAGE-keep for all datasets. RAGE outperforms RAGE-keep in all datasets except PLANTED CLIQUE which has the same performance. The usefulness of reinitialization is more obvious in larger datasets.

## 4.6.2 Keeping Base Model

RAGE incorporates reinitialization of the base GNN parameters before each inner loop iteration to eliminate unnecessary training trajectory from previously found explanations. This technique is known to reinforce effective features that may work under different conditions, as suggested by Zhou et al. [141]. Since the objective of RAGE is to combine sparsity and accuracy, there is no incentive to retain subgraphs that are unlikely to be used by the GNNs learned during the inner loop for making predictions. Therefore, restarting the entire GNN process after each iteration of RAGE is an effective approach to obtain a good representation of the subgraphs that will be applied by any accurate GNN trained using our method. Additionally, reinitializing neural network weights has been found to improve generalization on data, as discussed by Alabdulmohsin et al. [142].

To demonstrate the effect of reinitialization, we created a version of RAGE (RAGEkeep) without reinitializing the base GNN. The results of Table 4.4 indicate that reinitialization significantly improved the classifier quality. Notably, for larger datasets like SUNGLASSES, the difference was more pronounced.


Figure 4.7: Training and validation loss over iterations of bilevel optimization training of Mutagenicity dataset. Even though our training process is not unstable, there is still room for improvement.

### 4.7 Training Stability

We provide the change in training and validation loss over epochs. Antoniou et.al. [150] discuss the drawback of bilevel optimization algorithms in terms of stability and proposes a few modifications. Here, we provide the same analysis done in [150] for bilevel optimization training. The results show that RAGE is not significantly affected by the training instability of bilevel optimization. It could be because of the selection of inner and outer-loop epochs. However, Figure 4.7 shows that there is still room for improvement. The solutions provided in [150] to make the training process more stable are also applicable to our method, RAGE.

## 4.8 Conclusions

We investigate the problem of generating explanations for GNN-based graph-level tasks (classification and regression) and propose RAGE, a novel ante-hoc GNN explainer based on bilevel optimization. RAGE inductively learns compact and accurate explanations by optimizing the GNN and explanations jointly. Moreover, different from several baselines, RAGE explanations do not omit any information used by the model, thus enabling the model behavior to be reproduced based on the explanations. We will compare RAGE against state-of-the-art graph classification methods and GNN explainers using synthetic and real datasets. Our preliminary results demonstrate that RAGE consistently outperforms the baseline models across various evaluation metrics, such as reproducibility and robustness to noise. Moreover, our initial case studies (Figure 4.1) vividly illustrate RAGE's ability to accurately identify pertinent components within the graphs, leading to compelling explanations and insights.

**Future Work:** Our work will be finalized by conducting more experiments such as evaluating the robustness of our method by measuring various metrics, including the quality of explanations across different runs or base models. To enhance our proposed approach, we plan to explore the potential of a sampling-based extension, which can enable the discovery of multiple plausible yet independent explanations for predictions. Furthermore, we intend to investigate the application of our method in the semi-supervised setting, where a limited number of labeled explanations are available, and where human evaluation can be incorporated in an interactive manner.

**Limitations:** RAGE still has room for improvement on its training stability (Sec. 4.7) and training running time due to the nature of bilevel optimization. We discuss potential improvements for training stability. For running time, gradient estimations or momentum-based optimization can be applied [151] for faster training.

**Broader Impacts:** We do not anticipate any direct negative societal impact. However, it is possible for third parties to use our method and declare that it generates good explanations and accuracy without proper evaluation on the domain.

# Chapter 5

# Link Prediction without Graph Neural Networks

# 5.1 Introduction

Machine learning on graphs supports various structured-data applications including social network analysis [152, 153, 154], recommender systems [155, 156, 157], natural language processing [158, 159, 160], and physics modeling [161, 162, 163]. Among the graph-related tasks, one could argue that link prediction [164, 165] is the most fundamental one. This is because link prediction not only has many concrete applications [166, 167, 168] but can also be considered an (implicit or explicit) step of the graphbased machine learning pipeline [169, 170, 171]—as the observed graph is usually noisy and/or incomplete.

In recent years, Graph Neural Networks (GNNs) [13, 14, 73] have emerged as the predominant paradigm for machine learning on graphs. Similar to their great success in node classification [106, 172, 173] and graph classification [55, 17, 174], GNNs have been shown to achieve state-of-the-art link prediction performance [175, 176, 177].



Figure 5.1: GNN incorporates topology into attributes via message-passing, which is effective for tasks **on** the topology. Link prediction, however, is a task **for** the topology, which motivates the design of Gelato—a novel framework that leverages graph learning to incorporate attributes into topology.

Compared to classical approaches that rely on expert-designed heuristics to extract topological information (e.g., Common Neighbors [178], Adamic-Adar [179], Preferential Attachment [180]), GNNs have the potential to discover new heuristics via supervised learning and the natural advantage of incorporating node attributes.

However, there is little understanding of what factors contribute to the success of GNNs in link prediction, and whether simpler alternatives can achieve comparable performance—as recently found for node classification [181]. GNN-based methods approach link prediction as a binary classification problem. Yet different from other classification problems, link prediction deals with extremely class-imbalanced data due to the sparsity of real-world graphs. We argue that class imbalance should be accounted for in both training and evaluation of link prediction. In addition, GNNs combine topological and attribute information by learning topology-smoothened attributes (embeddings) via message-passing [182]. This attribute-centric mechanism has been proven effective for tasks on the topology such as node classification [183], but link prediction is a task for the topology, which naturally motivates topology-centric paradigms (see Figure 5.1).

The goal of this paper is to address the key issues raised above. We first show that the evaluation of GNN-based link prediction pictures an overly optimistic view of model performance compared to the (more realistic) imbalanced setting. Class imbalance also prevents the generalization of these models due to bias in their training. Instead, we propose the use of the N-pair loss with an unbiased set of training edges to account for class imbalance. Moreover, we present *Gelato*, a novel framework that combines topological and attribute information for link prediction. As a simpler alternative to GNNs, our model applies topology-centric graph learning to incorporate node attributes directly into the graph structure, which is given as input to a topological heuristic, Autocovariance, for link prediction. Extensive experiments demonstrate that our model significantly outperforms state-of-the-art GNN-based methods in both accuracy and scalability.

To summarize, our contributions are:

- ▷ We scrutinize the training and evaluation of supervised link prediction methods and identify their limitations in handling class imbalance.
- ▷ We propose a simple, effective, and efficient framework to combine topological and attribute information for link prediction without using GNNs.
- ▷ We introduce an N-pair link prediction loss combined with an unbiased set of training edges that we show to be more effective at addressing class imbalance.

# 5.2 Limitations in supervised link prediction evaluation and training

Supervised link prediction is often formulated as a binary classification problem, where the positive (or negative) class includes node pairs connected (or not connected) by a link. A key difference between link prediction and typical classification problems (e.g., node classification) is that the two classes in link prediction are *extremely* imbalanced, since most real-world graphs of interest are sparse (see Table 5.1). However, we find that class imbalance is not properly addressed in both evaluation and training of existing supervised link prediction approaches, as discussed below.

Link prediction evaluation. Area Under the Receiver Operating Characteristic Curve (AUC) and Average Precision (AP) are the two most popular evaluation metrics for supervised link prediction [184, 175, 185, 186, 187, 188, 189, 190, 177]. We first argue that, as in other imbalanced classification problems [191, 192], AUC is not an effective evaluation metric for link prediction as it is biased towards the majority class (nonedges). On the other hand, AP and other rank-based metrics such as Hits@k—used in Open Graph Benchmark (OGB) [193]—are effective for imbalanced classification if evaluated on a test set that follows the original class distribution. Yet, existing link prediction methods [184, 175, 187, 189, 177] compute AP on a test set that contains all positive test pairs and only an equal number of random negative pairs. Similarly, OGB computes Hits@k against a very small subset of random negative pairs. We term these approaches biased testing as they highly overestimate the ratio of positive pairs in the graph. Evaluation metrics based on these biased test sets provide an overly optimistic measurement of the actual performance in *unbiased testing*, where every negative pair is included in the test set. In fact, in real applications where test positive edges are not known a priori, it is impossible to construct those biased test sets to begin with. Below, we also present an illustrative example of the misleading performance evaluation based on *biased* testing.

*Example*: Consider a graph with 10k nodes, 100k edges, and 99.9M disconnected (or negative) pairs. A (bad) model that ranks 1M false positives higher than the true edges achieves 0.99 AUC and 0.95 in AP under *biased testing* with equal negative samples. (Detailed computation in Section 5.4.8.)

The above discussion motivates a more representative evaluation setting for supervised link prediction. Specifically, we argue for the use of rank-based evaluation metrics—AP, Precision@k [164], and Hits@k [194]—with *unbiased testing*, where positive edges are ranked against all negative pairs. These metrics have been widely applied in related problems, such as unsupervised link prediction [164, 195, 196, 197], knowledge graph completion [194, 198, 199], and information retrieval [200], where class imbalance is also significant. In our experiments, we will illustrate how these evaluation metrics combined with *unbiased testing* provide a drastically different and more informative performance evaluation compared to existing approaches.

Link prediction training. Following the formulation of supervised link prediction as binary classification, most existing models adopt the binary cross entropy loss to optimize their parameters [184, 175, 185, 186, 188, 176, 189, 190]. To deal with class imbalance, these approaches downsample the negative pairs to match the number of positive pairs in the training set (*biased training*). We highlight two drawbacks of *biased training*: (1) it induces the model to overestimate the probability of positive pairs, and (2) it discards potentially useful evidence from most negative pairs. Notice that the first drawback is often hidden by *biased testing*. Instead, this paper proposes the use of *unbiased training*, where the ratio of negative pairs in the training set is the same as in the input graph. To train our model in this highly imbalanced setting, we apply the N-pair loss for link prediction instead of the cross entropy loss (Section 5.3.3).

### 5.3 Method

Notation and problem. Consider an attributed graph G = (V, E, X), where V is the set of n nodes, E is the set of m edges (links), and  $X = (x_1, ..., x_n)^T \in \mathbb{R}^{n \times r}$  collects r-dimensional node attributes. The topological (structural) information of the graph is represented by its adjacency matrix  $A \in \mathbb{R}^{n \times n}$ , with  $A_{uv} > 0$  if an edge of weight  $A_{uv}$ connects nodes u and v and  $A_{uv} = 0$ , otherwise. The (weighted) degree of node u is given as  $d_u = \sum_v A_{uv}$  and the corresponding degree vector (matrix) is denoted as  $d \in \mathbb{R}^n$  $(D \in \mathbb{R}^{n \times n})$ . The volume of the graph is  $(G) = \sum_u d_u$ . Our goal is to infer missing links



in G based on its topological and attribute information, A and X.

Figure 5.2: Gelato applies graph learning to incorporate attribute information into the topology via an MLP. The learned graph is given to a topological heuristic that predicts edges between node pairs with high Autocovariance similarity. The parameters of the MLP are optimized end-to-end using the N-pair loss. Experiments show that Gelato outperforms state-of-the-art GNN-based link prediction methods.

**Model overview.** Figure 5.2 provides an overview of our link prediction model. It starts with a topology-centric graph learning phase that incorporates node attribute information directly into the graph structure via a Multi-layer Perceptron (MLP). We then apply a topological heuristic, Autocovariance (AC), to the attribute-enhanced graph to obtain a pairwise score matrix. Node pairs with the highest scores are predicted as (positive) links. The scores for training pairs are collected to compute an N-pair loss. Finally, the loss is used to train the MLP parameters in an end-to-end manner. We named our model Gelato (Graph enhancement for link prediction with <u>autocovariance</u>). Gelato represents a paradigm shift in supervised link prediction by combining a graph encoding of attributes with a topological heuristic instead of relying on increasingly popular GNN-based embeddings.

#### 5.3.1 Graph learning

The goal of graph learning is to generate an enhanced graph that incorporates node attribute information into the topology. This can be considered as the "dual" operation of message-passing in GNNs, which incorporates topological information into attributes (embeddings). We argue that graph learning is the more suitable scheme to combine attributes and topology for link prediction, since link prediction is a task for the topology itself (as opposed to other applications such as node classification).

Specifically, our first step of graph learning is to augment the original edges with a set of node pairs based on their (untrained) attribute similarity (i.e., adding an  $\varepsilon$ neighborhood graph):

$$\widetilde{E} = E + \{(u, v) \mid s(x_u, x_v) > \varepsilon_\eta\}$$
(5.1)

where  $s(\cdot)$  can be any similarity function (we use cosine in our experiments) and  $\varepsilon_{\eta}$  is a threshold that determines the number of added pairs as a ratio  $\eta$  of the original number of edges m.

A simple MLP then maps the pairwise node attributes into a trained edge weight for every edge in  $\widetilde{E}$ :

$$w_{uv} = ([x_u; x_v]; \theta) \tag{5.2}$$

where  $[x_u; x_v]$  denotes the concatenation of  $x_u$  and  $x_v$  and  $\theta$  contains the trainable parameters. For undirected graphs, we instead use the following permutation invariant operator [201]:

$$w_{uv} = ([x_u + x_v; |x_u - x_v|]; \theta)$$
(5.3)

The final edge weights of the enhanced graph are a weighted combination of the topological weights, the untrained weights, and the trained weights:

$$\widetilde{A}_{uv} = \alpha A_{uv} + (1 - \alpha)(\beta w_{uv} + (1 - \beta)s(x_u, x_v))$$
(5.4)

where  $\alpha$  and  $\beta$  are hyperparameters. The enhanced adjacency matrix  $\widetilde{A}$  is then fed into a topological heuristic for link prediction introduced in the next section. Note that the MLP is not trained directly to predict the links, but instead trained end-to-end to enhance the input graph given to the topological heuristic. Also note that the MLP can be easily replaced by a more powerful model such as a GNN, but the goal of this paper is to demonstrate the general effectiveness of our framework and we will show that even a simple MLP leads to significant improvement over the base heuristic.

#### 5.3.2 Topological heuristic

Assuming that the learned adjacency matrix  $\tilde{A}$  incorporates structural and attribute information, Gelato applies a topological heuristic to  $\tilde{A}$ . Specifically, we adopt Autocovariance, which has been shown to achieve state-of-the-art link prediction results for non-attributed graphs [197].

Autocovariance is a random-walk based similarity metric. Intuitively, it measures the difference between the co-visiting probabilities for a pair of nodes in a truncated walk and in an infinitely long walk. Given the enhanced graph  $\widetilde{G}$ , the Autocovariance similarity matrix  $R \in \mathbb{R}^{n \times n}$  is given as

$$R = \frac{\widetilde{D}}{(\widetilde{G})} (\widetilde{D}^{-1} \widetilde{A})^t - \frac{\widetilde{d}\widetilde{d}^T}{{}^2(\widetilde{G})}$$
(5.5)

where  $t \in \mathbb{N}_0$  is the scaling parameter of the truncated walk. Each entry  $R_{uv}$  represents a similarity score for node pair (u, v) and top similarity pairs are predicted as links. Note that  $R_{uv}$  only depends on the t-hop enclosing subgraph of (u, v) and can be easily differentiated with respect to the edge weights in the subgraph. In fact, Gelato could be applied with any differentiable topological heuristic or even a combination of them. In our experiments (Section 5.4.2), we will show that Autocovariance alone enables stateof-the-art link prediction performance.

Next, we introduce how to train our model parameters with supervised information.

#### 5.3.3 N-pair loss and unbiased training

As we have mentioned in Section 5.2, current supervised link prediction methods rely on *biased training* and the cross entropy loss (CE) to optimize model parameters. Instead, Gelato applies the N-pair loss [202] that is inspired by the metric learning and learningto-rank literature [203, 204, 205, 206] to train the parameters of our graph learning model (see Section 5.3.1) from highly imbalanced *unbiased training* data.

The N-pair loss (NP) contrasts each positive training edge (u, v) against a set of negative pairs N(u, v). It is computed as follows:

$$L(\theta) = -\sum_{(u,v)\in E} \log \frac{\exp(R_{uv})}{\exp(R_{uv}) + \sum_{(p,q)\in N(u,v)} \exp(R_{pq})}$$
(5.6)

Intuitively,  $L(\theta)$  is minimized when each positive edge (u, v) has a much higher similarity than its contrasted negative pairs:  $R_{uv} \gg R_{pq}, \forall (p,q) \in N(u,v)$ . Compared to CE, NP is more sensitive to negative pairs that have comparable similarities to those of positive pairs—they are more likely to be false positives. While NP achieves good performance in our experiments, alternative losses from the learning-to-rank literature [207, 208, 209] could also be applied.

Gelato generates negative samples N(u, v) using unbiased training. This means that N(u, v) is a random subset of all disconnected pairs in the training graph, and |N(u, v)| is proportional to the ratio of negative pairs over positive ones. In this way, we leverage more information contained in negative pairs compared to *biased training*. Note that, similar to unbiased training, (unsupervised) topological heuristics implicitly use information from all edges and non-edges. Also, unbiased training can be combined with adversarial negative sampling methods [210, 211] from the knowledge graph embedding literature to increase the quality of contrasted negative pairs.

Complexity analysis. The only trainable component in our model is the graph learning MLP with  $O(rh + lh^2)$  parameters—where r is the number of node features, l is the number of hidden layers, and h is the number of neurons per layer. Notice that the number of parameters is independent of the graph size. Constructing the  $\varepsilon$ -neighborhood graph based on cosine similarity can be done efficiently using hashing and pruning [212, 213]. Computing the enhanced adjacency matrix with the MLP takes  $O((1 + \eta)mr)$  time per epoch—where m = |E| and  $\eta$  is the ratio of edges added to E from the  $\varepsilon$ -neighborhood graph. We apply sparse matrix multiplication to compute k entries of the t-step AC in  $O(\max(k, (1+\eta)mt))$  time. Note that unlike recent GNN-based approaches [175, 214, 177] that generate distinctive subgraphs for each link (e.g., via the labeling trick), enclosing subgraphs for links in Gelato share the same information (i.e., learned edge weights), which significantly reduces the computational cost. Our experiments will demonstrate Gelato's efficiency in training and inference.

### 5.4 Experiments

We provide empirical evidence for our claims regarding supervised link prediction and demonstrate the accuracy and efficiency of Gelato. Our implementation is anonymously available at https://anonymous.4open.science/r/Gelato/.

#### 5.4.1 Experiment settings

**Datasets.** Our method is evaluated on five attributed graphs commonly used as link prediction benchmark [185, 186, 188, 189, 190, 177]:

▷ CORA [72] and CITESEER [215] are citation networks where nodes represent scientific publications (classified into seven and six classes, respectively) and edges represent the citations between them. Attributes of each node is a binary word vector indicating the absence/presence of the corresponding word from a dictionary.

- ▷ PUBMED [216] is a citation network where nodes represent scientific publications (classified into three classes) and edges represent the citations between them. Attributes of each node is a TF/IDF weighted word vector.
- ▷ PHOTO and COMPUTERS are subgraphs of the Amazon co-purchase graph [217] where nodes represent products (classified into eight and ten classes, respectively) and edges imply that two products are frequently bought together. Attributes of each node is a bag-of-word vector encoding the product review.

We use the publicly available version of the datasets from the pytorch-geometric library [218] (under the MIT licence) curated by [219] and [220]. Table 5.1 shows dataset statistics. Table 5.1)

	#Nodes	#Edges	# Attrs	Avg. degree	Density
Cora	2,708	$5,\!278$	$1,\!433$	3.90	0.14%
CITESEER	3,327	4,552	3,703	2.74	0.08%
PubMed	19,717	$44,\!324$	500	4.50	0.02%
Рното	$7,\!650$	$119,\!081$	745	31.13	0.41%
Computers	13,752	$245,\!861$	767	35.76	0.26%

Table 5.1: A summary of dataset statistics.

**Baselines.** For GNN-based link prediction, we include six state-of-the-art methods published in the past two years: LGCN [186], TLC-GNN [188], Neo-GNN [176], NBFNet [189], BScNets [190], and WalkPool [177], as well as three pioneering works—GAE [184], SEAL [175], and HGCN [185]. For topological link prediction heuristics, we consider Common Neighbors (CN) [178], Adamic Adar (AA) [179], Resource Allocation (RA) [221], and Autocovariance (AC) [197]—the base heuristic in our model. To demonstrate the superiority of the proposed end-to-end model, we also include an MLP trained directly for link prediction, the cosine similarity (Cos) between node attributes, and AC on top of the respective weighted/augmented graphs (i.e., two-stage approaches where the MLP

is trained separately for link prediction rather than trained end-to-end) as baselines.

We list link prediction baselines and their reference repositories we use in our experiments in Table 5.2. Note that we had to implement the batched training and testing for several baselines as their original implementations do not scale to *unbiased training* and *unbiased testing* without downsampling.

Baseline	Repository
GAE [13]	https://github.com/zfjsail/gae-pytorch
SEAL [175]	https://github.com/facebookresearch/SEAL_OGB
HGCN [185]	https://github.com/HazyResearch/hgcn
LGCN [186]	https://github.com/ydzhang-stormstout/LGCN/
TLC-GNN [188]	https://github.com/pkuyzy/TLC-GNN/
Neo-GNN [176]	https://github.com/seongjunyun/Neo-GNNs
NBFNet $[189]$	https://github.com/DeepGraphLearning/NBFNet
BScNets $[190]$	https://github.com/BScNets/BScNets
WalkPool [177]	https://github.com/DaDaCheng/WalkPooling
AC [197]	https://github.com/zexihuang/random-walk-embedding

Table 5.2: Reference of baseline code repositories.

Hyperparameters. For Gelato, we tune the proportion of added edges  $\eta$  from {0.0, 0.25, 0.5, 0.75, 1.0}, the topological weight  $\alpha$  from {0.0, 0.25, 0.5, 0.75}, and the trained weight  $\beta$  from {0.25, 0.5, 0.75, 1.0}, with a sensitivity analysis included in Section 5.4.6. All other settings are fixed across datasets: MLP with one hidden layer of 128 neurons, AC scaling parameter t = 3, Adam optimizer [115] with a learning rate of 0.001, a dropout rate of 0.5, and *unbiased training* without downsampling. For baselines, we use the same hyperparameters as in their papers.

Data splits for unbiased training and unbiased testing. Following [184, 175, 185, 186, 190, 177], we adopt 85%/5%/10% ratios for training, validation, and testing. Specifically, for *unbiased training* and *testing*, we first randomly (with seed 0) divide the (positive) edges E of the original graph into  $E_{train}^+$ ,  $E_{valid}^+$ , and  $E_{test}^+$  for training, validation, and testing based on the selected ratios. Then, we set the negative pairs

in these three sets as (1)  $E_{train}^- = E^- + E_{valid}^+ + E_{test}^+$ , (2)  $E_{valid}^- = E^- + E_{test}^+$ , and (3)  $E_{test}^- = E^-$ , where  $E^-$  is the set of all negative pairs (excluding self-loops) in the original graph. Notice that the validation and testing *positive* edges are included in the *negative* training set, and the testing *positive* edges are included in the *negative* validation set. These inclusions simulate the real-world scenario where the testing edges (and the validation edges) are unobserved during validation (training).

**Evaluation metrics.** We adopt Precision@k~(prec@k)—proportion of positive edges among the top k of all testing pairs, Hits@k~(hits@k)—ratio of positive edges individually ranked above kth place against all negative pairs, and Average Precision (AP)—area under the precision-recall curve, as evaluation metrics. We report results from 10 runs with random seeds ranging from 1 to 10.

**Positive masking.** For unbiased training, a trick similar to negative injection [175] in biased training is needed to guarantee model generalizability. Specifically, we divide the training positive edges into batches and during the training with each batch  $E_b$ , we feed in only the residual edges  $E - E_b$  as the structural information to the model. This setting simulates the testing phase, where the model is expected to predict edges without using their own connectivity information. We term this trick positive masking.

**Other implementation details.** We add self-loops to the enhanced adjacency matrix to ensure that each node has a valid transition probability distribution that is used in computing Autocovariance. The self-loops are added to all isolated nodes in the training graph for PUBMED, PHOTO, and COMPUTERS, and to all nodes for CORA and CITESEER. Following the postprocessing of the Autocovariance matrix for embedding in [197], we standardize Gelato similarity scores before computing the loss. For training with the cross entropy loss, we further add a linear layer with the sigmoid activation function to map our prediction score to a probability. We optimize our model with gradient descent via autograd in pytorch [222]. We find that the gradients are

sometimes invalid when training our model (especially with the cross entropy loss), and we address this by skipping the parameter updates for batches leading to invalid gradients. Finally, we use *prec*@100% on the (unbiased) validation set as the criteria for selecting the best model from all training epochs. The maximum number of epochs for CORA/CITESEER/PUBMED and PHOTO/COMPUTERS are set to be 100 and 250, respectively.

**Experiment environment.** We run our experiments in an *a2-highgpu-1g* node of the Google Cloud Compute Engine. It has one NVIDIA A100 GPU with 40GB HBM2 GPU memory and 12 Intel Xeon Scalable Processor (Cascade Lake) 2nd Generation vCPUs with 85GB memory.

Number of trainable parameters. The only trainable component in Gelato is the graph learning MLP, which for Photo has 208,130 parameters. By comparison, the best performing GNN-based method, Neo-GNN, has more than twice the number of parameters (455,200).

#### 5.4.2 Link prediction performance

Table 5.3 summarizes the link prediction performance in terms of the mean and standard deviation of Average Precision (AP) for all methods. Figure 5.3 and Figure 5.4 show results based on prec@k (k as a ratio of test edges) and hits@k (k as the rank) for varying k.

First, we want to highlight the drastically different performance of GNN-based methods compared to those found in the original papers [186, 188, 176, 189, 190, 177] and reproduced in Section 5.4.9. While they achieve AUC/AP scores of often higher than 90% under *biased testing*, here we see most of them underperform even the simplest topological heuristics such as Common Neighbors under *unbiased testing*. These results

		Cora	CITESEER	PubMed	Рното	Computers
	GAE	$0.27\pm0.02$	$0.66\pm0.11$	$0.26\pm0.03$	$0.28 \pm 0.02$	$0.30\pm0.02$
	SEAL	$1.89 \pm 0.74$	$0.91 \pm 0.66$	***	$10.49 \pm 0.86$	$6.84^{*}$
	HGCN	$0.82\pm0.03$	$0.74 \pm 0.10$	$0.35 \pm 0.01$	$2.11\pm0.10$	$2.30 \pm 0.14$
	LGCN	$1.14 \pm 0.04$	$0.86\pm0.09$	$0.44 \pm 0.01$	$3.53\pm0.05$	$1.96\pm0.03$
GNN	TLC-GNN	$0.29\pm0.09$	$0.35\pm0.18$	OOM	$1.77\pm0.11$	OOM
	Neo-GNN	$2.05 \pm 0.61$	$1.61\pm0.36$	$1.21 \pm 0.14$	$10.83 \pm 1.53$	$6.75^*$
	NBFNet	$1.36\pm0.17$	$0.77\pm0.22$	***	$11.99 \pm 1.60$	***
	BScNets	$0.32\pm0.08$	$0.20\pm0.06$	$0.22\pm0.08$	$2.47 \pm 0.18$	$1.45 \pm 0.10$
	WalkPool	$2.04\pm0.07$	$1.39\pm0.11$	$1.31^{*}$	OOM	OOM
	CN	$1.10\pm0.00$	$0.74\pm0.00$	$0.36\pm0.00$	$7.73\pm0.00$	$5.09\pm0.00$
Topological	AA	$2.07\pm0.00$	$1.24 \pm 0.00$	$0.45 \pm 0.00$	$9.67\pm0.00$	$6.52\pm0.00$
Heuristics	RA	$2.02\pm0.00$	$1.19\pm0.00$	$0.33\pm0.00$	$10.77 \pm 0.00$	$7.71\pm0.00$
ficulistics	AC	$2.43\pm0.00$	$2.65\pm0.00$	$2.50\pm0.00$	$16.63\pm0.00$	$11.64 \pm 0.00$
	MLP	$0.30\pm0.05$	$0.44 \pm 0.09$	$0.14\pm0.06$	$1.01\pm0.26$	$0.41 \pm 0.23$
Attributes	$\cos$	$0.42 \pm 0.00$	$1.89\pm0.00$	$0.07\pm0.00$	$0.11\pm0.00$	$0.07\pm0.00$
+	MLP+AC	$3.24\pm0.03$	$1.95\pm0.05$	$2.61\pm0.06$	$15.99 \pm 0.21$	$11.25\pm0.13$
Topology	Cos+AC	$3.60\pm0.00$	$4.46\pm0.00$	$0.51\pm0.00$	$10.01 \pm 0.00$	$5.20\pm0.00$
1010089	MLP+Cos+AC	$3.39\pm0.06$	$4.15\pm0.14$	$0.55\pm0.03$	$10.88\pm0.09$	$5.75\pm0.11$
Gelato		$3.90\pm0.03$	$4.55\pm0.02$	$2.88\pm0.09$	$25.68 \pm 0.53$	$18.77 \pm 0.19$

 $^{*}$  Run only once as each run takes ~100 hrs;  $^{***}$  Each run takes >1000 hrs; OOM: Out Of Memory.

Table 5.3: Link prediction performance comparison (mean  $\pm$  std AP). Gelato consistently outperforms GNN-based methods, topological heuristics, and two-stage approaches combining attributes and topology.



Figure 5.3: Link prediction performance in terms of prec@k for varying values of k (as percentages of test edges). With few exceptions, Gelato outperforms the baselines across different values of k.



Figure 5.4: Link prediction performance in terms of hits@k for varying values of k. With few exceptions, Gelato outperforms the baselines across different values of k.

support our arguments from Section 5.2 that evaluation metrics based on *biased testing* can produce misleading results compared to *unbiased testing*. The overall best performing GNN model is Neo-GNN, which directly generalizes the pairwise topological heuristics. Yet still, it consistently underperforms AC, a random-walk based heuristic that needs neither node attributes nor supervision/training.

We then look at two-stage combinations of AC and models for attribute information. We observe noticeable performance gains from combining attribute cosine similarity and AC in CORA and CITESEER but not for other datasets. Other two-stage approaches achieve similar or worse performance.

Finally, Gelato significantly outperforms the best GNN-based model with an average relative gain of 145.2% and AC with a gain of 52.6% in terms of AP—similar results were obtained for *prec*@k and *hits*@k. This validates our hypothesis that a simple MLP can effectively incorporate node attribute information into the topology when our model is trained end-to-end. Next, we will provide insights behind these improvements and demonstrate the efficiency of Gelato on training and inference.

#### 5.4.3 Visualizing Gelato predictions

To better understand the performance of Gelato, we visualize its learned graph, prediction scores, and predicted edges in comparison with AC and the best GNN-based baseline (Neo-GNN) in Figure 5.5.

5.5a shows the input adjacency matrix for the subgraph of PHOTO containing the top 160 nodes belonging to the first class sorted in decreasing order of their (within-class) degree. 5.5b illustrates the enhanced adjacency matrix learned by Gelato's MLP. Comparing it with the Euclidean distance between node attributes in 5.5c, we see that our enhanced adjacency matrix effectively incorporates attribute information. More specifically, we notice the down-weighting of the edges connecting the high-degree nodes with larger attribute distances (matrix entries 0-40 and especially 0-10) and the up-weighting of those connecting medium-degree nodes with smaller attribute distances (40-80). In 5.5d and 5.5e, we see the corresponding AC scores based on the input and the enhanced adjacency matrix (Gelato). Since AC captures the degree distribution of nodes [197], the vanilla AC scores greatly favor high-degree nodes (0-40). By comparison, thanks to the down-weighting, Gelato assigns relatively lower scores to edges connecting them to lowdegree nodes (60-160), while still capturing the edge density between high-degree nodes (0-40). The immediate result of this is the significantly improved precision as shown in 5.5h compared to 5.5g. Gelato covers as many positive edges in the high-degree region as AC while making far fewer wrong predictions for connections involving low-degree nodes.

The prediction probabilities and predicted edges for Neo-GNN are shown in 5.5f and 5.5i, respectively. Note that while it predicts edges connecting high-degree node pairs (0-40) with high probability, similar values are assigned to many low-degree pairs (80-160) as well. Most of those predictions are wrong, both in the low-degree region of 5.5i and also in other low-degree parts of the graph that are not shown here. This analysis supports our claim that Gelato is more effective at combining node attributes and the graph topology, enabling state-of-the-art link prediction.



Figure 5.5: Illustration of the link prediction process of Gelato, AC, and the best GNN-based approach, Neo-GNN, on a subgraph of PHOTO. Gelato effectively incorporates node attributes into the graph structure and leverages topological heuristics to enable state-of-the-art link prediction.

#### 5.4.4 Loss and training setting

In this section, we demonstrate the advantages of the proposed N-pair loss and *unbiased training* for supervised link prediction. Figure 5.6 shows the training and validation losses and *prec*@100% (our validation metric) in training Gelato based on the cross entropy (CE) and N-pair (NP) losses under *biased* and *unbiased training* respectively. The final test AP scores are shown in the titles.



Figure 5.6: Training of Gelato based on different losses and training settings for Photo with test AP (mean  $\pm$  std) shown in the titles. Compared with the cross entropy loss, the N-pair loss with *unbiased training* is a more consistent proxy for *unbiased testing* metrics and leads to better peak performance.

In the first column (CE with *biased training*), different from the training, both loss and precision for (unbiased) validation decrease. This leads to even worse test performance compared to the untrained base model (i.e., AC). In other words, albeit being the most popular loss function for supervised link prediction, CE with *biased training* does not generalize to *unbiased testing*. On the contrary, as shown in the second column, the proposed NP loss with *biased training*—equivalent to the pairwise logistic loss [223]—is a more effective proxy for *unbiased testing* metrics.

The right two columns show results with *unbiased training*, which is better for CE

as more negative pairs are present in the training set (with the original class ratio). On the other hand, NP is more consistent with unbiased evaluation metrics, leading to 8.5% better performance. This is because, unlike CE, which optimizes positive and negative pairs independently, NP contrasts negative pairs against positive ones, giving higher importance to negative pairs that are more likely to be false positives.

#### 5.4.5 Ablation study

We have demonstrated the superiority of Gelato over its individual components and two-stage approaches in Table 5.3 and analyzed the effect of losses and training settings in Section 5.4.4. Here, we collect the results with the same hyperparameter setting as Gelato and present a comprehensive ablation study in Table 5.4.5. Specifically, *Gelato-MLP* (AC) represents Gelato without the MLP (Autocovariance) component, i.e., only using Autocovariance (MLP) for link prediction. *Gelato-NP* (UT) replaces the proposed Npair loss (*unbiased training*) with the cross entropy loss (*biased training*) applied by the baselines. Finally, *Gelato-NP+UT* replaces both the loss and the training setting.

	Cora	CITESEER	PubMed	Рното	Computers
Gelato-MLP	$2.43\pm0.00$	$2.65\pm0.00$	$2.50\pm0.00$	$16.63\pm0.00$	$11.64 \pm 0.00$
Gelato-AC	$1.94\pm0.18$	$3.91\pm0.37$	$0.83 \pm 0.05$	$7.45\pm0.44$	$4.09 \pm 0.16$
Gelato-NP+UT	$2.98\pm0.20$	$1.96\pm0.11$	$2.35\pm0.24$	$14.87 \pm 1.41$	$9.77 \pm 2.67$
Gelato-NP	$1.96\pm0.01$	$1.77\pm0.20$	$2.32\pm0.16$	$19.63\pm0.38$	$9.84 \pm 4.42$
Gelato-UT	$3.07\pm0.01$	$1.95\pm0.05$	$2.52\pm0.09$	$23.66 \pm 1.01$	$11.59 \pm 0.35$
Gelato	$3.90\pm0.03$	$4.55\pm0.02$	$2.88\pm0.09$	$25.68\pm0.53$	$18.77\pm0.19$

Table 5.4: Results of the ablation study based on AP scores. Each component of Gelato plays an important role in enabling state-of-the-art link prediction performance.

We observe that removing either MLP or Autocovariance leads to inferior performance, as the corresponding attribute or topology information would be missing. Further, to address the class imbalance problem of link prediction, both the N-pair loss and unbiased training are crucial for effective training of Gelato.

While all supervised baselines originally adopt *biased training*, we also implement the same *unbiased training* (and N-pair loss) as Gelato for those that are scalable in Section 5.4.10—results are consistent with the ones discussed in Section 5.4.2.

#### 5.4.6 Sensitivity analysis

The selected hyperparameters of Gelato for each dataset are recorded in Table 5.5, and a sensitivity analysis of  $\eta$ ,  $\alpha$ , and  $\beta$  are shown in Figure 5.7 and Figure 5.8 respectively for Photo and Cora.

	Cora	CITESEER	PubMed	Рното	Computers
$\eta$	0.5	0.75	0.0	0.0	0.0
$\alpha$	0.5	0.5	0.0	0.0	0.0
$\beta$	0.25	0.5	1.0	1.0	1.0

Table 5.5: Selected hyperparameters of Gelato.



Figure 5.7: Performance of Gelato with different values of  $\eta$ .

For most datasets, we find that simply setting  $\beta = 1.0$  and  $\eta = \alpha = 0.0$  leads to the best performance, corresponding to the scenario where no edges based on cosine



Figure 5.8: Performance of Gelato with different  $\alpha$  and  $\beta$ .

similarity are added and the edge weights are completely learned by the MLP. For CORA and CITESEER, however, we first notice that adding edges based on untrained cosine similarity alone leads to improved performance (see Table 5.3), which motivates us to set  $\eta = 0.5/0.75$ . In addition, we find that a large trainable weight  $\beta$  leads to overfitting of the model as the number of node attributes is large while the number of (positive) edges is small for CORA and CITESEER (see Table 5.1). We address this by decreasing the relative importance of trained edge weights ( $\beta = 0.25/0.5$ ) and increasing that of the topological edge weights ( $\alpha = 0.5$ ), which leads to better generalization and improved performance. Based on our experiments, these hyperparameters can be easily tuned using simple hyperparameter search techniques, such as line search, using a small validation set.

	GAE	SEAL	HGCN	LGCN	TLC-GNN	Neo-GNN	NBFNet	BScNets	MLP	Gelato
Training	1,022	11,493	92	56	42,440	14,807	30,896	115	232	1,265
Testing	0.031	380	0.093	0.099	5.722	346	76,737	0.394	1.801	0.057

Table 5.6: Training and inference time comparison between supervised link prediction methods for PHOTO. Gelato has competitive training time (even under *unbiased training*) and is significantly faster than most baselines in terms of inference, especially compared to the best GNN model, Neo-GNN.

#### 5.4.7 Running time

We compare Gelato with other supervised link prediction methods in terms of running time for PHOTO in Table 5.4.6. As the only method that applies *unbiased training* with more negative samples—Gelato shows a competitive training speed that is  $11 \times$ faster than the best performing GNN-based method, Neo-GNN. In terms of inference time, Gelato is much faster than most baselines with a 6,000× speedup compared to Neo-GNN. We further observe more significant efficiency gains for Gelato over Neo-GNN for larger datasets—e.g.,  $14 \times$  (training) and  $25,000 \times$  (testing) for COMPUTERS.

# 5.4.8 Analysis of link prediction evaluation metrics with different test settings

In Section 5.2, we present an example scenario where a bad link prediction model that ranks 1M false positives higher than the 100k true edges achieves good AUC/AP with *biased testing*. Here, we provide the detailed computation steps and compare the results with those based on *unbiased testing*.

5.9a and 5.9b show the receiver operating characteristic (ROC) and precision-recall (PR) curves for the model under *biased testing* with equal negative samples. Due to the downsampling, only 100k (out of 99.9M) negative pairs are included in the test set, among which only  $100k/99.9M \times 1M \approx 1k$  pairs are ranked higher than the positive edges. In the ROC curve, this means that once the false positive rate reaches 1k/100k = 0.01, the true positive rate would reach 1.0, leading to an AUC score of 0.99. Similarly, in the PR curve, when the recall reaches 1.0, the precision is  $100k/(1k + 100k) \approx 0.99$ , leading to an overall AP score of ~0.95.

By comparison, as shown in 5.9c, when the recall reaches 1.0, the precision under *unbiased testing* is only  $100k/(1M + 100k) \approx 0.09$ , leading to an AP score of ~0.05. This demonstrates that evaluation metrics based on *biased testing* provide an overly optimistic measurement of link prediction model performance compared to the more realistic *unbiased testing* setting.



Figure 5.9: Receiver operating characteristic and precision-recall curves for the bad link prediction model that ranks 1M false positives higher than the 100k true edges. The model achieves 0.99 in AUC and 0.95 AP under *biased testing*, while the more informative performance evaluation metric, Average Precision (AP) under *unbiased testing*, is only 0.05.

#### 5.4.9 Results based on AUC scores

As we have argued in Section 5.2, AUC is not an effective evaluation metric for link prediction (even under *unbiased testing*) as it is biased towards the majority class. In Table 5.7, we report the AUC scores for different methods under *unbiased testing*. These results, while being consistent with those found in the link prediction literature, disagree with those obtained using the rank-based evaluation metrics under *unbiased testing*.

		Cora	CITESEER	PubMed	Рното	Computers
	GAE	$87.30\pm0.22$	$87.48\pm0.39$	$94.10\pm0.22$	$77.59\pm0.73$	$79.36\pm0.37$
	SEAL	$91.82\pm1.08$	$90.37\pm0.91$	***	$98.85 \pm 0.04$	$98.7^{*}$
	HGCN	$92.60\pm0.29$	$92.39\pm0.61$	$94.40 \pm 0.14$	$96.08\pm0.08$	$97.86 \pm 0.10$
	LGCN	$91.60\pm0.23$	$93.07\pm0.77$	$95.80\pm0.03$	$98.36 \pm 0.01$	$97.81 \pm 0.01$
GNN	TLC-GNN	$91.57\pm0.95$	$91.18\pm0.78$	OOM	$98.20\pm0.08$	OOM
	Neo-GNN	$91.77\pm0.84$	$90.25 \pm 0.80$	$90.43 \pm 1.37$	$98.74 \pm 0.55$	$98.34^*$
	NBFNet	$86.06 \pm 0.59$	$85.10\pm0.32$	***	$98.29\pm0.35$	***
	BScNets	$91.59 \pm 0.47$	$89.62 \pm 1.05$	$97.48 \pm 0.07$	$98.68\pm0.06$	$98.41\pm0.05$
	WalkPool	$92.33\pm0.30$	$90.73\pm0.42$	$98.66^*$	OOM	OOM
	CN	$71.15\pm0.00$	$66.91\pm0.00$	$64.09\pm0.00$	$96.73\pm0.00$	$96.15 \pm 0.00$
Topological	AA	$71.22\pm0.00$	$66.92\pm0.00$	$64.09 \pm 0.00$	$97.02\pm0.00$	$96.58 \pm 0.00$
Heuristics	$\mathbf{R}\mathbf{A}$	$71.22\pm0.00$	$66.93\pm0.00$	$64.09 \pm 0.00$	$97.20\pm0.00$	$96.82\pm0.00$
moundered	AC	$75.41\pm0.00$	$72.41\pm0.00$	$67.46 \pm 0.00$	$98.24 \pm 0.00$	$96.86\pm0.00$
	MLP	$85.43\pm0.36$	$87.89 \pm 2.05$	$87.89 \pm 2.05$	$91.24 \pm 1.61$	$88.84 \pm 2.58$
Attributes	$\cos$	$79.06\pm0.00$	$89.86 \pm 0.00$	$89.14 \pm 0.00$	$71.46 \pm 0.00$	$71.68 \pm 0.00$
+ Topology	MLP+AC	$79.77 \pm 0.03$	$82.26 \pm 0.29$	$66.31 \pm 0.74$	$98.02 \pm 0.05$	$96.69 \pm 0.05$
	$\cos + AC$	$86.34 \pm 0.00$	$89.29\pm0.00$	$75.56 \pm 0.00$	$97.28\pm0.00$	$96.26 \pm 0.00$
	MLP+Cos+AC	$86.90 \pm 0.14$	$89.42\pm0.09$	$75.78\pm0.27$	$97.27\pm0.01$	$96.24\pm0.01$
Gelato		$83.12 \pm 0.06$	$88.38\pm0.02$	$64.93\pm0.06$	$98.01\pm0.03$	$96.72 \pm 0.04$

<sup>\*</sup> Run only once as each run takes ~100 hrs; <sup>\*\*\*</sup> Each run takes >1000 hrs; OOM: Out Of Memory.

Table 5.7: Link prediction performance comparison (mean  $\pm$  std AUC). AUC results conflict with other evaluation metrics, presenting a misleading view of the model performance for link prediction.

#### 5.4.10 Baseline results with unbiased training

Table 5.8 summarizes the link prediction performance in terms of mean and standard deviation of AP for Gelato and the baselines using *unbiased training* without downsampling and the cross entropy loss, and Figure 5.10 and Figure 5.11 show results based on

		Cora	CiteSeer	PubMed	Рното	Computers
	GAE SEAL	$0.33 \pm 0.21 \\ 2.24^*$	$0.69 \pm 0.18 \\ 1.11^*$	$0.63^{*}_{***}$	$1.36 \pm 3.38$	$7.91^{*}_{***}$
	HGCN	$0.54\pm0.23$	$1.02\pm0.05$	$0.41^{*}$	$3.27 \pm 2.97$	$2.60^{*}$
	LGCN	$1.53 \pm 0.08$	$1.45 \pm 0.09$	$0.55^{*}$	$2.90 \pm 0.26$	$1.13^{*}$
GNN	TLC-GNN	$0.68\pm0.16$	$0.61\pm0.19$	OOM	$2.95 \pm 1.50$	OOM
	Neo-GNN	$2.76 \pm 0.36$	$1.80 \pm 0.22$	***	***	***
	NBFNet	***	***	***	***	***
	BScNets	$1.13 \pm 0.25$	$1.27 \pm 0.20$	$0.47^{*}$	$8.54 \pm 1.55$	$4.40^{*}$
	WalkPool	***	***	***	OOM	OOM
	$_{\rm CN}$	$1.10\pm0.00$	$0.74 \pm 0.00$	$0.36\pm0.00$	$7.73\pm0.00$	$5.09\pm0.00$
Topological	AA	$2.07\pm0.00$	$1.24 \pm 0.00$	$0.45 \pm 0.00$	$9.67 \pm 0.00$	$6.52 \pm 0.00$
Heuristics	RA	$2.02\pm0.00$	$1.19\pm0.00$	$0.33\pm0.00$	$10.77 \pm 0.00$	$7.71\pm0.00$
liounstics	AC	$2.43\pm0.00$	$2.65\pm0.00$	$2.50\pm0.00$	$16.63\pm0.00$	$11.64\pm0.00$
	MLP	$0.22\pm0.27$	$1.17 \pm 0.63$	$0.44 \pm 0.12$	$1.15\pm0.40$	$1.19\pm0.46$
Attributes + Topology	$\cos$	$0.42\pm0.00$	$1.89\pm0.00$	$0.07\pm0.00$	$0.11\pm0.00$	$0.07\pm0.00$
	MLP+AC	$0.63\pm0.32$	$1.00\pm0.43$	$1.17\pm0.44$	$11.88\pm0.83$	$8.73 \pm 0.45$
	Cos+AC	$3.60 \pm 0.00$	$4.46 \pm 0.00$	$0.51 \pm 0.00$	$10.01 \pm 0.00$	$5.20 \pm 0.00$
	MLP+Cos+AC	$3.80\pm0.01$	$3.94\pm0.03$	$0.77\pm0.01$	$12.80\pm0.03$	$7.57\pm0.02$
(	Gelato	$\overline{3.90\pm0.03}$	$4.55\pm0.02$	$2.88\pm0.09$	$25.68\pm0.53$	$18.77\pm0.19$

prec@k and hits@k for varying k values.

<sup>\*</sup> Run only once as each run takes ~100 hrs; <sup>\*\*\*</sup> Each run takes >1000 hrs; OOM: Out Of Memory.

Table 5.8: Link prediction performance comparison (mean  $\pm$  std AP) with supervised link prediction methods using *unbiased training*. While we observe noticeable improvement for some baselines (e.g., BScNets), Gelato still consistently and significantly outperform the baselines.

We first note that *unbiased training* without downsampling brings serious scalability challenges to most GNN-based approaches, making scaling up to larger datasets intractable. For the scalable baselines, *unbiased training* leads to marginal (e.g., Neo-GNN) to significant (e.g., BScNets) gains in performance. However, all of them still underperform the topological heuristic, Autocovariance, in most cases, and achieve much worse performance compared to Gelato. This supports our claim that the topologycentric graph learning mechanism in Gelato is more effective than the attribute-centric message-passing in GNNs for link prediction, leading to state-of-the-art performance.

As for the GNN-based baselines using both *unbiased training* and our proposed Npair loss, we have attempted to modify the training functions of the reference repositories



Figure 5.10: Link prediction performance in terms of prec@k (in percentage) for varying values of k with baselines using *unbiased training*. While we observe noticeable improvement for some baselines (e.g., BScNets), Gelato still consistently and significantly outperform the baselines.



Figure 5.11: Link prediction performance in terms of hits@k (in percentage) for varying values of k with baselines using *unbiased training*. While we observe noticeable improvement for some baselines (e.g., BScNets), Gelato still consistently and significantly outperform the baselines.

of the baselines and managed to train SEAL, LGCN, Neo-GNN, and BScNet. However, despite our best efforts, we are unable to obtain better link prediction performance using the N-pair loss. We defer the further investigation of the incompatibility of our loss and the baselines to future research. On the other hand, we have observed significantly better performance for MLP with the N-pair loss compared to the cross entropy loss under *unbiased training*. This can be seen by comparing the MLP performance in Table 5.8 here and the *Gelato-AC* performance in Table 5.4.5 in the ablation study. The improvement shows the potential benefit of applying our training setting and loss to other supervised link prediction methods.

We also show the training time comparison between different supervised link predic-

tion methods using *unbiased training* without downsampling in Table 5.4.10. Gelato is the second fastest model, only slower than the vanilla MLP. This further demonstrates that Gelato is a more efficient alternative compared to GNNs.

	GAE	SEAL	HGCN	LGCN	TLC-GNN	Neo-GNN	NBFNet	$\operatorname{BScNets}$	MLP	Gelato
Training	6,361	>450,000	$1,\!668$	1,401	$53,\!304$	>450,000	>450,000	2,323	744	1,285

Table 5.9: Training time comparison between supervised link prediction methods for PHOTO under *unbiased training*. Gelato, while achieving the best performance, is also the second most efficient method in terms of total training time, slower only than the vanilla MLP.

## 5.5 Related work

**Topological heuristics for link prediction.** The early link prediction literature focuses on topology-based heuristics. This includes approaches based on local (e.g., Common Neighbors [178], Adamic Adar [179], and Resource Allocation [221]) and higher-order (e.g., Katz [224], PageRank [225], and SimRank [226]) information. More recently, random-walk based graph embedding methods, which learn vector representations for nodes [105, 227, 197], have achieved promising results in graph machine learning tasks. Popular embedding approaches, such as DeepWalk [105] and node2vec [227], have been shown to implicitly approximate the Pointwise Mutual Information similarity [228], which can also be used as a link prediction heuristic. This has motivated the investigation of other similarity metrics such as Autocovariance [229, 197, 230]. However, these heuristics are unsupervised and cannot take advantage of data beyond the topology.

**Graph Neural Networks for link prediction.** GNN-based link prediction addresses the limitations of topological heuristics by training a neural network to combine topological and attribute information and potentially learn new heuristics. GAE [184] combines a graph convolution network [13] and an inner product decoder based on node embeddings for link prediction. SEAL [175] models link prediction as a binary subgraph classification problem (edge/non-edge), and follow-up work (e.g., SHHF [214], WalkPool [177]) investigates different pooling strategies. Other recent approaches for GNN-based link prediction include learning representations in hyperbolic space (e.g., HGCN [185], LGCN [186]), generalizing topological heuristics (e.g., Neo-GNN [176], NBFNet [189]), and incorporating additional topological features (e.g., TLC-GNN [188], BScNets [190]). Motivated by the growing popularity of GNNs for link prediction, this work investigates key questions regarding their training, evaluation, and ability to learn effective topological heuristics directly from data. We propose Gelato, which is simpler, more accurate, and faster than the state-of-the-art GNN-based link prediction methods. Gelato learns a graph that combines topological and attribute Graph learning. information. Our goal differs from generative models [231, 232, 233], which learn to sample from a distribution over graphs. Graph learning also enables the application of GNNs when the graph is unavailable, noisy, or incomplete (for a recent survey, see [234]). LDS [139] and GAug [235] jointly learn a probability distribution over edges and GNN parameters. IDGL [236] and EGLN [237] alternate between optimizing the graph and embeddings for node/graph classification and collaborative filtering. [238] proposes two-stage link prediction by augmenting the graph as a preprocessing step. In comparison, Gelato effectively learns a graph in an end-to-end manner by minimizing the loss of a topological heuristic.

## 5.6 Conclusion

This work sheds light on key limitations in how GNN-based link prediction methods handle the intrinsic class imbalance of the problem and presents more effective and efficient ways to combine attributes and topology. Our findings might open new research directions on machine learning for graph data, including a better understanding of the advantages of increasingly popular and sophisticated deep learning models compared to more traditional and simpler graph-based solutions.

**Future Works:** We have identified several potential extensions and improvements for our framework and experiments:

- ▷ Scalability: Initially, Gelato had a space complexity of  $O(V^2)$ , which limited its applicability to large datasets, such as those found in OGB [193]. To address this limitation, we plan to develop a version of Gelato that incorporates graph partitioning techniques, enabling scalability to handle large datasets more efficiently.
- ▷ Theoretical justification: We will focus on providing a theoretical explanation for why existing GNN-based approaches appear to struggle with learning autocovariance based on their expressive power. This investigation aims to deepen our understanding of the limitations and challenges associated with autocovariance learning in GNNs.
- ▷ Gelato+GNN: As an additional enhancement, we intend to implement a variant of Gelato that employs a GNN instead of an MLP to learn the underlying graph. By leveraging the increased expressive power of GNNs, we anticipate improved performance and capabilities in the Gelato framework.

# Chapter 6

# Feature-based Individual Fairness in k-Clustering

# 6.1 Introduction

Machine learning systems are increasingly being used in various societal decisionmaking, including predicting recidivism [239, 240], deciding interest rates [241], and even allocating healthcare resources [242]. However, beginning with the report on bias in recidivism risk prediction [239], it has been known that such systems are often biased against certain groups of people. In recent years, various methods and definitions have been proposed for ensuring fairness in supervised learning settings, with efforts ranging from debiasing datasets [243] to explicitly encoding the fairness constraints during the training of a classifier [244].

Our paper focuses on fairness in unsupervised learning, particularly clustering. There are two major reasons why clustering should be fair with respect to different subgroups. First, clustering is often a pre-processing step for generating new data representations for downstream tasks. Since we want the downstream decisions to be fair, the clustering step needs to be unbiased [245]. Second, clustering is also used in various resource allocation problems, e.g., in *facility location* [246]. Since it is desirable that no group is disproportionately affected by such decisions, there has been an increasing interest in designing clustering algorithms that are fair with respect to different subgroups [247, 248, 249, 250]. Such group-fair clustering algorithms ensure that each protected group has an approximately equal presence in each cluster.

Compared to group fairness, individually fair clustering has received less attention. Individually fair clustering is motivated by the *facility location problem* where the goal is to open k facilities while minimizing the total transportation cost between individuals and their nearest facility. If we choose k facilities (or centers) uniformly at random, then each point x could expect one of its nearest n/k neighbors to be one of such facilities. This led a few studies [246, 251, 252] to consider the following notion of individual fairness. For a point x, let r(x) be the radius such that the ball of radius r(x) centered at x has at least n/k points. An individually fair clustering guarantees that, for every x, a cluster center is chosen from the r(x)-neighborhood of x.

Although individually fair clustering [246] provides guarantees for each point, it does not exactly reflect the original premise of individual fairness suggested by [253], which requires that similar individuals should receive similar decisions. In the context of clustering, this means that two points x and y that are similar (in terms of features) should be clustered similarly. However, the definition proposed by [246] does not provide such a guarantee, as points similar to a point x could be different from the points within a radius of r(x) from x.

**Proposed Definition of Individual Fairness.** In order to address the drawback above, we propose a new notion of individual fairness in clustering. First, motivated by the original definition of individual fairness in supervised learning [253], we introduce a

feature-based notion of individual fairness. We say that two individuals are similar if their features match significantly (parameterized by  $\gamma$  in Definition 3). Now, for each individual v, let C(v) denote the cluster v is assigned to. Then our feature-based individually fair clustering requires that C(v) also contains at least  $m_v$  individuals that are similar to v. The variable  $m_v$  is a parameter to encode the degree of fairness. More specifically, it encodes the amount of similarity an individual seeks inside their own cluster. This guarantees that a point v is not isolated in its own cluster but that the cluster has a desired representation (or participation) from points similar to it. Note that, the features that are used to compute similarity for individual fairness might not necessarily be used for clustering. In fact, these two sets of features might be disjoint.

Our notion of individual fairness guarantees that similar individuals (in terms of possibly sensitive features) often share similar clusters. Consider the following motivating example. Ad networks collect user behavior data (e.g., browsing history, location) as well as possibly sensitive attributes (e.g., race and gender) to cluster users into several categories [254]. These categories are directly used for targeted recommendations, including jobs and healthcare. In this context, the cost function of the clustering algorithm should be based on user behavior while an individual's notion of fairness should be based on sensitive attributes such as race and gender. In this case, similarity in terms of sensitive features can be seen as a relaxation of a protected group membership.

**Contributions.** Our main contributions are as follows:

- ▷ Novel Formulation: We propose a new definition of individual fairness in clustering based on how individuals are similar in terms of their features. Our definition guarantees that each individual has a desired level of representation of similar individuals in their own cluster.
- ▷ **Problem Characterization:** We show that minimizing the clustering cost subject

to the new notion of individual fairness is NP-hard, and also cannot be approximated within a factor  $\delta$  for any  $\delta > 0$ .

- ▷ Algorithm: We design a randomized algorithm providing an additive approximation cost while guaranteeing fairness within a multiplicative factor with high probability.
- ▷ Experiments: Our experiments on several standard datasets show that our approach produces by 34.5% less cost on average in clustering than the best-competing method while ensuring individual fairness for more than 95.5% points on average.

**Related Work.** [247] first introduced the problem of fair clustering with disparate impact constraints and their goal was to ensure that all the protected groups have approximately equal representation in every cluster. Several works [255, 256] studied different generalizations of the fair clustering problem. Furthermore, several papers [248, 257, 258] proposed procedures to scale fair clustering to a large number of points. Although we consider individual fairness, our work is related to [249], which shows that a  $\rho$ -approximation to the vanilla clustering problem can be converted to a ( $\rho + 2$ )-approximate solution to fair clustering with bounded (and often negligible) violation of fairness constraints.

Our paper is focused on individual fairness, which was first defined by [253] in the context classification, and requires similar individuals to be treated similarly. For clustering problems, such a notion of individual fairness was first defined by [246]. They studied individual fairness in terms of the guarantee a randomly chosen set of k points must satisfy. Informally, an individually fair clustering guarantees that for each point x, a cluster center is chosen from a certain neighborhood of x. [251] designed a bicriteria approximation algorithm for individually fair k-means and k-median problems. Their algorithms guarantee that not only the fairness constraints are approximately satisfied, but also the objective is approximately maximized. Later, [252] proposed an algorithm that has theoretical fairness guarantees comparable with [251], and empirically, obtains
noticeably fairer solutions. Recently, [259] designed improved bicriteria algorithms for general  $\ell_p$ -norm costs. Another recent study [260] defined a coreset for individually fair clustering problem using the generalized fair radius notion, and [261] used per-point fairness and aggregate fairness constraints for the k-center problem. Later they incorporated the price of fairness notion to combine these two constraints into one algorithm. [262] study stochastic pairwise constraints between points. However, unlike ours, these constraints are insufficient to model the individual fairness constraints.

The definition of individual fairness in [246] was mainly motivated by fairness in the facility location problem. Recently, [263] considered a different notion of individual fairness in clustering, where the goal is to ensure that each point, is closer to the points in its own cluster than the points in any other cluster. Our proposed definition can be seen as a way to capture these two notions, as we consider feature-based similarity, as well as guaranteed representation for each point.

Here, we focus on the  $\ell_p$ -norm cost for clustering, which is just the sum of  $\ell_p$ -distances of each point from its corresponding cluster center. [261] also considers  $\ell_p$ -norm objectives in their individual fairness formulation. However, several papers did consider other objectives in the context of group-fair clustering [250, 264, 265]. Finally, our focus is on fair clustering algorithms, and there is extensive literature on fair algorithms for unsupervised [266, 267] and supervised learning more broadly [244, 268, 269]. The coverage of these algorithms is out of the scope of the paper, and we refer the interested reader to the following excellent surveys: [270, 271] and [272].

# 6.2 Preliminaries

We first introduce some necessary notations. Let V be a set of n points  $V = \{1, 2, ..., n\}$ . We denote  $\{S_1, S_2, ..., S_q\}$  as a set of q features, where  $S_i$  is the set of val-

ues for the *i*-th feature. We denote the tuple of q features of the point *i* by  $X_i = (X_i^t)_{t \in [q]}$ . We write  $C = (C_i)_{i \in [k]}$  to denote a clustering (i.e. partition) of the set V and  $(c_i)_{i \in [k]}$  to denote the corresponding cluster centers. Given a clustering C and a point v, let  $\phi(v, C)$ be the cluster center assigned to the point v. When the clustering C is clear from the context, we use  $\phi(v)$  to denote the cluster center assigned to the vertex v. We are also given a distance function  $d: V \times V \to \mathbb{R}$  that measures the distance between any pair of points. Given a clustering C, we can measure its cost through the distance metric d. In particular, we will be interested in measuring the sum of the p-th powers of distances from each point to its cluster center for  $p \in \mathbb{N} \cup \{0\}$ :

$$\operatorname{Cost}(C) = \sum_{v \in V} d(v, \phi(C, v))^p.$$
(6.1)

We assume that the distance function d depends on some features of the points but don't assume any relationship between those features and the ones used for fairness.

#### 6.2.1 Similarity

In order to define the feature-based notion of individual fairness, we first define a similarity measure based on the features. To the best of our knowledge, all the existing notions of individual fairness in clustering only depend on the distance-based neighborhood of each point. In contrast, our definition of individual fairness builds upon the features of individual points that are not necessarily used for clustering. In order to define the feature-based notion of fairness, we first define a similarity measure based on the features.

In real-world settings, the fairness features can be both continuous and discrete. To handle both cases, we convert the discrete variables (features) into one-hot encoding vectors. The continuous variables are also normalized to be within the range [0, 1]. After these conversions, we can now define the distance (or similarity) between two vectors. We convert the distance to similarity as follows:

$$s(X_i, X_j) = e^{-d(X_i, X_j)}$$
 (6.2)

where s is the similarity between  $X_i$  and  $X_j$  and d is a distance function (e.g., Euclidean). This operation guarantees that s will always generate a value between 0 and 1. We say that  $X_i$  and  $X_j$  are gamma similar if  $s > \gamma$ .

**Definition 3** ( $\gamma$ -similarity). For a parameter  $\gamma \in [0, 1]$ , we say two points  $i, j \in V, i \neq j$ are  $\gamma$ -similar if  $s(X_i, X_j) > \gamma$  where  $s(X_i, X_j) = e^{-d'(X_i, X_j)}$ . We assume that a point is not  $\gamma$ -matched with itself.

Our definition of similarity is flexible enough to support diverse applications. For any point v, we use  $\Gamma(v)$  to denote the set of points in V that are  $\gamma$ -similar to v. Next, we introduce our definition of individually fair clustering.

#### 6.2.2 Individual Fairness

**Definition 4** (Individual Fairness in Clustering). Given a set V of n points along with a q-length feature vector  $X_v = (x_v^1, \ldots, x_v^q)$  for every point  $v \in V$ , a similarity parameter  $\gamma \in [0, 1]$ , an integer tuple  $(m_v)_{v \in V}$ , and an integer k, we say that a clustering  $(C_i)_{i \in [\ell]}$  $(\ell \leq k)$  is  $(m_v)_{v \in V}$ -individually fair if it satisfies the following constraint for every point  $v \in V$ :

$$|\{ u : u \in \Gamma(v) \text{ and } \phi(u) = \phi(v) \}| \ge m_v \tag{6.3}$$

The fairness constraint (6.3) says that at least  $m_v$  points that are  $\gamma$ -similar to point v must belong to the cluster of v. Our goal is to cluster V into  $\ell (\leq k)$  clusters,  $(C_i)_{i \in [\ell]}$ ,

with corresponding centers (or facilities<sup>1</sup>)  $(c_i)_{i \in [\ell]}$ , such that clusters are individually fair for every point and minimize the clustering cost (e.g. sum of the *p*-th powers of distances from cluster centers for some  $p \in \mathbb{N} \cup \{0\}$ ). Formally, our INDIVIDUALLY FAIR CLUSTERING problem is defined as follows.

**Definition 5** (INDIVIDUALLY FAIR CLUSTERING (IFC)). The input is a set V of n points with a q-length feature vector  $X_v = (x_v^1, \ldots, x_v^q)$  for each  $v \in V$ , a similarity parameter  $\gamma \in [0, 1]$ , an integer tuple  $(m_v)_{v \in V}$ , a set F of potential facilities. The objective is to open a subset  $S \subseteq F$  of at most k facilities, and find an assignment  $\phi : V \longrightarrow S$  to minimize Cost(C) satisfying the fairness constraints (eq., 6.3).

The classical clustering problem, which we call VANILLA CLUSTERING, is the same as the IFC problem except for the fairness requirements from Equation 6.3.

# 6.3 Results

We present our main technical results in two directions. First, we provide several hardness results to show that the general INDIVIDUALLY FAIR CLUSTERING (IFC) problem is hard even if one considers approximation. Then we contrast the hardness results by developing randomized approximation algorithms for various special cases of the IFC problem.

#### 6.3.1 Hardness Results

In order to prove hardness results, we consider the decision version of the IFC problem, where the goal is to find a clustering whose cost is below a certain threshold. Note that, it is always possible to find a (trivial) individually fair clustering by one cluster containing

<sup>&</sup>lt;sup>1</sup>We use cluster center and facility interchangeably.

all the points. However, the cost of such a fair clustering could be high, and we ask whether it is possible to beat the cost of such trivially fair clustering. As there can be multiple trivial fair clustering (depending on the cluster center chosen), we naturally pick the one minimizing the cost as the benchmark.

**Definition 6** (TRIVIALLY FAIR CLUSTERING). Given a set V of n points along with qlength feature vector  $X_v = (x_v^1, \ldots, x_v^q)$  for every point  $v \in V$ , the trivially fair clustering puts all points in one cluster and picks the point as cluster center which minimizes the cost:

$$\min_{f \in F} \sum_{v \in V} d(v, f)^p$$

We show that it is NP-complete to compute if there exists a clustering better than TRIVIALLY FAIR CLUSTERING by providing a reduction from SATISFACTORY-PARTITION, which is known to be NP-complete [273].

**Definition 7** (SATISFACTORY-PARTITION). Given a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  and an integer  $\lambda_v$  for every vertex  $v \in \mathcal{V}$ , compute if there exists a partition  $(\mathcal{V}_1, \mathcal{V}_2)$  of  $\mathcal{V}$  such that

i) 
$$\mathcal{V}_1, \mathcal{V}_2 \neq \emptyset$$

ii) For every  $i \in [2]$  and every  $v \in \mathcal{V}_i$ , the number of neighbors of v in  $\mathcal{V}_i$  is at least  $\lambda_v$ .

We denote an arbitrary instance of SATISFACTORY-PARTITION by  $(\mathcal{G}, (\lambda_v)_{v \in \mathcal{G}})$ .

**Theorem 1.** It is NP-complete to decide whether an instance of INDIVIDUALLY FAIR CLUSTERING admits a clustering of cost less than the TRIVIALLY FAIR CLUSTERING even when there are only 2 facilities.

*Proof:* The problem clearly belongs to NP. We now exhibit a reduction from SATISFACTORY-PARTITION. Let  $(\mathcal{G} = (\mathcal{V} = \{v_1, v_2, \dots, v_n\}, \mathcal{E}), (\lambda_v)_{v \in \mathcal{V}})$  be an arbitrary

instance of SATISFACTORY-PARTITION. In our INDIVIDUALLY FAIR CLUSTERING instance, we have two facilities l and r and the set of points  $U = \{u_1, u_2, \ldots, u_n\}$ . We define the distances as

$$d(u_i, l) = \begin{cases} \left( \left\lceil \frac{n}{2} \right\rceil + \beta \right)^{1/p} & 1 \leqslant i \leqslant \left\lceil \frac{n}{2} \right\rceil + 1 \\ \beta^{1/p} & \left\lceil \frac{n}{2} \right\rceil + 2 \leqslant i \leqslant n \end{cases}$$
$$d(u_i, r) = \begin{cases} \beta^{1/p} & 1 \leqslant i \leqslant \left\lfloor \frac{n}{2} \right\rfloor \\ \left( \left\lceil \frac{n}{2} \right\rceil + \beta + 1 \right)^{1/p} & \left\lfloor \frac{n}{2} \right\rfloor + 1 \leqslant i \leqslant n \end{cases}$$

for any  $\beta \ge \frac{\lceil \frac{n}{2} \rceil + 1}{2}$ . We can easily verify that the following properties are satisfied:

i) The distances satisfy the triangle inequality.

ii) 
$$\sum_{v \in V} d(v, l)^p = \sum_{v \in V} d(v, r)^p = A$$
 (say).

iii) For any  $X \,\subset\, V, X \neq \emptyset, X \neq V$ , we have  $\sum_{v \in X} d(v, l)^p \neq \sum_{v \in X} d(v, r)^p$ . To see this, let  $s = |X \cap \{u_1, u_2, \dots, u_{\lceil \frac{n}{2} \rceil + 1}\}|$  and  $t = |X \cap \{u_{\lfloor \frac{n}{2} \rfloor + 1}, u_{\lfloor \frac{n}{2} \rfloor + 2}, \dots, u_n\}|$ so that  $s \leqslant \lceil \frac{n}{2} \rceil + 1$  and  $t \leqslant \lceil \frac{n}{2} \rceil$ . Then  $\sum_{v \in X} d(v, l)^p = s \lceil \frac{n}{2} \rceil + \beta |X|$  and  $\sum_{v \in X} d(v, r)^p = t(\lceil \frac{n}{2} \rceil + 1) + \beta |X|$ . Thus  $\sum_{v \in X} d(v, l)^p = \sum_{v \in X} d(v, r)^p$  would imply  $s \lceil \frac{n}{2} \rceil = t(\lceil \frac{n}{2} \rceil + 1)$  and therefore either s = t = 0 or  $s = \lceil \frac{n}{2} \rceil + 1$  and  $t = \lceil \frac{n}{2} \rceil$ . In the former case  $X = \emptyset$  while in the latter case X = V, a contradiction.

We now describe the feature vector. For every edge  $e \in \mathcal{E}$ , we have a feature  $\theta_e$ . A point  $u_i, i \in [n]$  has value 1 for  $\theta_e$  if the edge e is incident on the vertex  $v_i$  in  $\mathcal{G}$ ; otherwise has value 0 for  $\theta_e$ . We define the distance function d' on feature space as the number of

features that two points differ. Finally, we set the similarity parameter  $\gamma = \frac{e^{-m} + e^{-(m-1)}}{2}$ . We observe that two points  $u_i, u_j, i, j \in [n]$  are  $\gamma$ -similar if and only if there is an edge between  $v_i$  and  $v_j$  in  $\mathcal{G}$ . Finally, we define  $m_{v_i} = \lambda_{v_i}$  for every  $i \in [n]$ .

We claim that the SATISFACTORY-PARTITION instance is a yes instance if and only if there exists a fair clustering of U with cost < A. The "if" part follows directly, since any fair clustering of U with cost < A must be non-trivial and fairness ensures that the corresponding partition of  $\mathcal{G}$  is satisfactory.

For the "only if" part, let  $(X, \overline{X})$  be a non-trivial satisfactory partition of  $\mathcal{G}$ . Let  $\phi_1$  denote the assignment that assigns all corresponding vertices in X to l and all corresponding vertices in  $\overline{X}$  to r, and  $\phi_2$  denote the assignment that assigns all vertices in X to r and all vertices in  $\overline{X}$  to l. Thus,

$$\operatorname{Cost}(\phi_1) = \sum_{v \in X} d(v, l)^p + \sum_{v \in \bar{X}} d(v, r)^p$$
$$\operatorname{Cost}(\phi_2) = \sum_{v \in X} d(v, r)^p + \sum_{v \in \bar{X}} d(v, l)^p$$

Thus,  $\operatorname{Cost}(\phi_1) + \operatorname{Cost}(\phi_2) = 2A$ . Now, it cannot be the case that  $\operatorname{Cost}(\phi_1) = \operatorname{Cost}(\phi_2) = A$ , which would imply that one of the assignments  $\phi_1$  or  $\phi_2$  must have  $\operatorname{cost} < A$ . Suppose to the contrary that  $\operatorname{Cost}(\phi_1) = A$ . Thus,  $\sum_{v \in X} d(v, l)^p + \sum_{v \in \bar{X}} d(v, r)^p = \sum_{v \in X} d(v, r)^p + \sum_{v \in \bar{X}} d(v, r)^p$  and therefore  $\sum_{v \in X} d(v, l)^p = \sum_{v \in X} d(v, r)^p$ , a contradiction.

Given the NP-completeness result, we explore the possibility of approximation for the INDIVIDUALLY FAIR CLUSTERING (IFC) problem. However, the next theorem shows that IFC is inapproximable within factor  $\delta$  for any  $\delta > 0$ .

**Theorem 2.** Distinguishing between instances of the IFC problem having zero and nonzero optimal costs is NP-complete even when there are 2 facilities. Hence, for any computable function  $\delta$ , there does not exist a  $\delta$ -approximation algorithm for IFC unless P=NP.

*Proof:* Let  $\mathcal{A}$  be a deterministic polynomial time algorithm that distinguishes between instances of IFC with 0 and non-zero optimal costs. We use  $\mathcal{A}$  to build an algorithm for SATISFACTORY-PARTITION. Let  $(\mathcal{G} = (\mathcal{V} = \{v_1, v_2, \ldots, v_n\}, \mathcal{E}), (\lambda_v)_{v \in \mathcal{V}})$ be an instance of SATISFACTORY-PARTITION. We create n - 1 instances  $I_1, I_2, \ldots, I_{n-1}$ of IFC as follows: the set of points is  $U = \{u_1, \ldots, u_n\}$  and  $m_{v_i} = \lambda_{v_i}$  for every  $i \in [n]$ for all the instances; for instance  $I_i, i \in [n]$ , we introduce 2 facilities l and r and define distances as follows:

$$d(u_j, l) = \begin{cases} 1 & j = 1 \\ 0 & j \in \{2, 3, \dots, n\} \end{cases}$$
$$d(u_j, r) = \begin{cases} 1 & j = i+1 \\ 0 & j \in \{1, 2, \dots, n\} \setminus \{i+1\} \end{cases}$$

We define feature vectors of every point similar to Theorem 1 to realize the above distances. We now run  $\mathcal{A}$  on each of these instances. If  $\mathcal{A}$  finds any instance in  $\{I_1, \ldots, I_{n-1}\}$ to have zero optimal cost, then we return yes for the SATISFACTORY-PARTITION instance; otherwise, we return no for the SATISFACTORY-PARTITION instance.

Clearly, the above algorithm runs in polynomial time. For proof of correctness; if  $\mathcal{G}$  does not have a non-trivial satisfactory partition, then clearly every instance in  $I_1, \ldots, I_{n-1}$  has an optimal cost of one. On the other hand, if  $\mathcal{G}$  has a non-trivial satisfactory partition, say  $(X, \bar{X})$ , then we claim that at least one of  $I_1, \ldots, I_{n-1}$  has optimal cost 0. Without loss of generality, we assume that we have  $v_1 \in X$ . Let  $v_j \in \bar{X}$ , for some  $j \in \{2, 3, \ldots, n\}$ . Then clearly  $OPT(I_{j-1}) = 0$  (assigning all the corresponding vertices in X to the cluster center r and all other vertices to the cluster center l).

The distances in the above reduction do not satisfy the triangle inequality. If we insist

triangle inequality of distances, then we have the following (weaker than Theorem 2) inapproximability result.

**Theorem 3.** There does not exist any FPTAS for the IFC problem when the distances in the input satisfy triangle inequality unless P = NP.

*Proof:* Let  $\mathcal{A}$  be an FPTAS for IFC. Similar to the proof of Theorem 2, we create n-1 instances of SATISFACTORY-PARTITION where instance  $I_i$  is as follows: the set of points is  $U = \{u_1, \ldots, u_n\}$  and  $m_{v_i} = \lambda_{v_i}$  for every  $i \in [n]$  for all the instances; for instance  $I_i, i \in [n]$ , we introduce 2 facilities l and r and define distances as follows:

$$d(u_j, l) = \begin{cases} (1+\beta)^{1/p} & j = 1\\ \beta^{1/p} & j \in \{2, 3, \dots, n\} \end{cases}$$

$$d(u_j, r) = \begin{cases} (1+\beta)^{1/p} & j = i+1\\\\ \beta^{1/p} & j \in \{1, 2, \dots, n\} \setminus \{i+1\} \end{cases}$$

where  $\beta$  is any constant  $\geq 1/2$ . The algorithm runs  $\mathcal{B}$  on each of the above instances with approximation parameter  $\varepsilon = \frac{1}{2n\beta}$ . If  $\mathcal{B}$  returns a solution of cost less than  $1 + n\beta$ on any instance, return yes for the SATISFACTORY-PARTITION instance; otherwise, we return no for the SATISFACTORY-PARTITION instance.

Clearly, the cost of the trivial partition is  $1 + n\beta$ . Thus, if  $\mathcal{G}$  does not have a nontrivial satisfactory partition, then  $\mathcal{B}$  must always return the trivial assignment of cost  $1 + n\beta$  for all instances. If  $\mathcal{G}$  has a satisfactory partition  $(X, \bar{X})$ , then as in Theorem 2, there exists an instance with optimal cost  $n\beta$ . Thus, the solution returned by  $\mathcal{B}$  will have cost at most  $n\beta(1 + \frac{1}{2n\beta}) < 1 + n\beta$ . Hence, the algorithm is correct.

#### 6.3.2 Algorithmic Results

Given the strong inapproximability results in the previous section, we aim to develop approximation algorithms for INDIVIDUALLY FAIR CLUSTERING under suitable conditions. First, we develop an approximation algorithm for INDIVIDUALLY FAIR ASSIGN-MENT (Theorem 4). Next, we show how to obtain an algorithm for INDIVIDUALLY FAIR CLUSTERING of similar guarantee (Theorem 5). Bera et al. [249] designed an approximation algorithm for group fair clustering from an algorithm for group fair assignment. We follow a similar approach to individual fairness.

One of the main ingredients of our technical results is the INDIVIDUALLY FAIR AS-SIGNMENT problem, which, given a set of k potential cluster centers, determines an assignment of the points i.e. which point should be assigned to which cluster center. Formally, it is defined as follows:

**Definition 8** (INDIVIDUALLY FAIR ASSIGNMENT (IFA)). Given a set V of n points along with a q-length feature vector  $X_v = (x_v^1, \ldots, x_v^q)$  for every point  $v \in V$ , a similarity parameter  $\gamma \in [0, 1]$ , an integer tuple  $(m_v)_{v \in V}$ , and a set  $F = \{f_1, \ldots, f_k\}$  of k facilities, an  $(m_v)_{v \in V}$ -fair assignment finds the optimal cost-minimizing assignment satisfying the fairness constraints (6.3).

Our Algorithm, LP-FAIR: Algorithm 4 describes our randomized approximation algorithm for IFA. The linear program (LP) in Inequality (6.4) is a relaxation of IFA problem and has a variable  $x_{v,f_k}$  for each vertex v and facility  $f_k$ . In an (integral) "solution" the variable  $x_{v,f_k}$  takes value 1 if and only if the point v is assigned to the facility  $f_k$ . After solving the LP, Algorithm 4 determines the assignment  $\phi$  by assigning point v to  $f_k$  with probability  $x_{v,f_k}^*$ . Finally, the above procedure is repeated  $\log_{1+\delta} n$ times and the assignment with the lowest cost is returned to boost the success probability.

The next theorem presents probabilistic approximation guarantees provided by Algo-

Algorithm 4 LP-FAIR, Algorithm for IFA

**Require:**  $(V, (X_v)_{v \in V}, \gamma, (m_v)_{v \in V}, k)$ , and  $\delta$ .

0: for  $t = 1, 2, ..., T = \log_{1+\delta} n$  do

0: Solve the following LP to get solution  $x_t^{\star}$ .

$$\min_{x} \sum_{v \in V} \sum_{f_k \in F} d(v, f_k)^p \cdot x_{v, f_k}$$
s.t. 
$$\sum_{u \in \Gamma(v)} x_{u, f_k} \ge m_v \cdot x_{v, f_k} \quad \forall v \in V, f_k \in F$$

$$\sum_{f_k \in F} x_{v, f_k} = 1 \quad \forall v \in V$$

$$x_{v, f_k} \ge 0 \quad \forall v \in V, f_k \in F$$
(6.4)

0: for each  $v \in V$  do

0: Set  $\phi_t(v) = f_k$  with probability  $x_{t,v,f_k}^{\star}$ .

0: return Assignment  $\phi^*$  with the minimum cost. =0

rithm 4.

**Theorem 4.** For any  $\varepsilon, \delta > 0$ , there is a randomized algorithm for IFA running polynomial time in n and  $\frac{1}{\delta}$ , that outputs a solution of cost at most  $(1 + \delta)OPT$  where each vertex v has at least  $\frac{m_v}{k}(1 - \varepsilon) \gamma$ -similar points assigned to the same facility with high probability if  $m_v = \Omega(\frac{k \log n}{\varepsilon^2}), \forall v \in V$ .

*Proof:* Let  $x^*$  be a solution to the linear program 6.4. Algorithm 4 assigns point v to  $f_k$  with probability  $x^*_{v,f_k}$ . We now prove the quality of this solution. Let X be the random variable denoting the number of points in  $\Gamma(v)$  that are assigned to the same facility as a given point v. For every  $u \in \Gamma(v)$ , let  $X_u$  be the indicator random variable indicating whether u and v are assigned to the same facility. Thus,

$$E[X_u] = \sum_{f_k \in F} x_{v,f_k}^* x_{u,f_k}^*$$

So we have,

$$E[X] = \sum_{u \in \Gamma(v)} E[X_u] = \sum_{f_k \in F} (x_{v, f_k}^* \sum_{u \in \Gamma(v)} x_{u, f_k}^*) \ge m_v \sum_{f_k \in F} x_{v, f_k}^{*^2} \ge m_v/k$$

Now using Chernoff bound,  $\Pr[v \text{ has at most } \frac{m_v}{k}(1-\varepsilon) \gamma \text{-similar points}] \leq e^{-\frac{\varepsilon^2}{2}\frac{m_v}{k}} \leq \frac{1}{n^2}$ 

And using union bound,  $\Pr[\exists \text{ a vertex that has at most } \frac{m_v}{k}(1-\varepsilon) \gamma$ -similar points]  $\leq \frac{1}{n}$ 

Also, clearly, the expected cost of the computed solution is at most OPT. Hence, using Markov's inequality,  $\Pr[\text{cost of computed solution is } \ge (1 + \delta)\text{OPT}] \le \frac{1}{1+\delta}$ 

As we repeat the above algorithm  $T = \frac{\log n}{\log(1+\delta)}$  times and output the solution with minimum cost, we have  $\Pr[\text{cost of the computed solution is } \ge (1+\delta)\text{OPT}] \le \frac{1}{n}$ 

Also, by union bound, the probability that there exist a vertex having at most  $\frac{m_v}{k}(1-\varepsilon)$   $\gamma$ -similar points in one of the T solutions is at most  $\frac{T}{n}$ .

# Algorithm 5 Algorithm for INDIVIDUALLY FAIR CLUSTERING Require: $(V, (X_v)_{v \in V}, \gamma, (m_v)_{v \in V}, k)$

- 0: Solve clustering problem  $(V, (X_v)_{v \in V}, \gamma, (m_v)_{v \in V}, k)$  using any vanilla algorithm ignoring fairness constraints. Let  $((C_i)_{i \in [\ell]}, (c_i)_{i \in [\ell]})$  be the output of the clustering algorithm.
- 0:  $F \leftarrow \{c_1, \ldots, c_\ell\}$
- 0: Run algorithm for INDIVIDUALLY FAIR ASSIGNMENT on  $(V, (X_v)_{v \in V}, \gamma, (m_v)_{v \in V}, F)$ . Let  $\phi$  be the output.
- 0: return  $(\phi^{-1}(c_i))_{i \in [\ell]}$  (ignore  $\phi^{-1}(c_j)$  if  $\phi^{-1}(c_j)$  is the empty set for some  $j \in [\ell] = 0$

We next show a method to obtain an approximation algorithm for INDIVIDUALLY FAIR CLUSTERING from an approximation algorithm for INDIVIDUALLY FAIR ASSIGN-MENT in a black box fashion.

**Theorem 5.** If the distances satisfy the triangle inequality, then the existence of an  $\rho$ -approximation algorithm for VANILLA CLUSTERING and an  $\alpha$ -approximation algorithm

for INDIVIDUALLY FAIR ASSIGNMENT with  $\lambda$ -multiplicative violations for some  $\lambda \ge 1$ implies the existence of an  $\alpha(\rho + 2)$ -approximation algorithm for INDIVIDUALLY FAIR CLUSTERING with  $\lambda$ -multiplicative violation.

*Proof:* Let  $S^*$  be the optimal set of facilities opened and  $\phi^*$  be the optimal assignment in the input INDIVIDUALLY FAIR CLUSTERING instance. Let S be the set of facilities returned by the vanilla k-clustering problem and  $\phi$  the assignment. For each  $f^* \in S^*$ , let us define  $\operatorname{nrst}(f^*) = \arg\min_{f \in S} d(f, f^*)$ . Consider the assignment  $\phi'$  over the set of facilities S that assigns each vertex v to  $\operatorname{nrst}(\phi^*(v))$ . We claim that  $\phi'$  is a fair assignment (please see Claim 1).

For any vertex v, let  $\phi(v) = f$ ,  $\phi'(v) = f'$  and  $\phi^*(v) = f^*$ . Thus  $d(v, f') \leq d(v, f^*) + d(f^*, \operatorname{nrst}(f^*)) \leq d(v, f^*) + d(f^*, f) \leq 2d(v, f^*) + d(v, f)$ . Since  $l_p$  is a monotone norm,  $l_p(S, \phi') \leq 2l_p(S^*, \phi^*) + l_p(S, \phi) \leq (\rho + 2)OPT$ . Now, since we have an  $\alpha$ -approximation algorithm for INDIVIDUALLY FAIR ASSIGNMENT, the solution returned by the algorithm will have cost at most  $\alpha \cdot l_p(S, \phi') \leq \alpha(\rho + 2)OPT$ .

Note that the proof requires that  $\phi'$  is an individually fair assignment. We prove the following claim:

#### Claim 1. $\phi'$ is an individually fair assignment.

*Proof:* For  $v \in V$ , let  $T_v$  denote the set of vertices assigned to  $\phi^*(v)$ . Since  $\phi^*$  is an individually fair assignment, the number of  $\gamma$ -similar points of v in  $T_v$  is at least  $m_v$ . Now all vertices in  $T_v$  are assigned to the facility  $nrst(\phi^*(v))$ . Thus, the number of  $\gamma$ -similar points of v in the assignment  $\phi'$  is at least  $m_v$ . Hence,  $\phi'$  is an individually fair assignment.

# 6.4 Experiments

Here, we conduct an experimental evaluation of our LP-based algorithm and eight baselines using the K-means cost function (unless specified otherwise) on three datasets. Our assessment includes performance on various metrics (Sec. 6.4.1), the impact of varying cluster numbers (Sec. 6.4.2), and cluster quality (Sec. 6.4.3). Additionally, we provide running time information (Sec. 6.5), and extra experiments (Sec. 6.6). Our implementation is available online at https://github.com/mertkosan/lp-fair.

**Datasets:** We use three datasets from the UCI repository (https://archive.ics.uci.edu/ml/datasets) in our experiments. These datasets have also been used by previous work [249, 251, 247]. We consider the following attributes for distance and  $\gamma$ -similarity (fairness):

- ▷ Adult [274]: Cluster labels determine whether a person makes over 50K a year. Distance features are "educationnum" and "age". The  $\gamma$ -similarity features are "salary" and "hoursperweek".
- ▷ Bank [275]: Data from customers of a bank. Distance features are "duration", and "age",  $\gamma$ -similarity features are "education" and "balance".
- ▷ **Diabetes:** Data from diabetes patients from 130 hospitals in the USA from 1999 to 2008. The distance features are "age" and "number-emergency". The  $\gamma$ -similarity features are "time\_in\_hospital", "num\_lab\_procedures".

We also provide experiments with randomly selected features.

**Algorithms:** We evaluate the following nine algorithms.

- ▷ Our LP-based approach (LP-FAIR): Provides probabilistic approximation guarantees (Algorithm 4). Our implementation is based on Algorithm 5 (Section 6.3.2).
- $\triangleright$  PBS [262]: Utilizes stochastic pairwise constraints in optimizing clustering while

incorporating individual fairness. The paper has two methods: **k-means-PBS** [262] and **k-center-PBS-CC** [262].

- FairCenter [246]: Ensures fairness based on the existence of a cluster center nearby. Notice that this algorithm has different fairness criteria than ours.
- ▷ P-PoF-Alg [261]: Incorporates the Price of Fairness notion and combines the constraints from Alg-PP [261] and Alg-AG [261], which optimizes the clusters based on a per-point and an aggregate fairness metric, respectively.
- ▷ Hochbaum-Shmoys (H-S) [276]: Uses the triangle inequality to solve a k-center problem with a 2-approximation algorithm.
- ▷ Gonzalez [277]: Minimizes the maximum intercluster distance. H-S and Gonzalez are used as baselines in [261] for comparison.

**Performance measures:** We evaluate the algorithms described above using the following metrics:

▷ Normalized Cost: Clustering cost (Equation 6.1) normalized by the cost of trivially fair clustering (Definition 6). The normalization removes the effect of datasetdependent feature distributions and makes it easier to compare the results across datasets:

Normalized 
$$Cost(A) = \frac{Cost(A)}{Cost(TRIVIALLY FAIR CLUSTERING)}$$

- ▷ **Fairness**: This denotes the fraction of points that satisfy individual fairness.
- ▷ Macro Fairness: This denotes the average of the Fairness metric for each cluster.
- ▷ Cluster Imbalance: This measures the imbalance of the found clusters in terms of their sizes. It is a standard deviation of cluster sizes (i.e., the number of elements in the cluster). The lower value of imbalance means the clusters are more balanced.

Other experimental settings: We choose 200 points from each dataset randomly for all the experiments. The experiments are run five times, and averages and standard deviations are reported based on these repetitions. Let  $\Gamma_v$  denote the number of  $\gamma$ matched points of v in the entire dataset and k' be the number of clusters found by the algorithm. The initial  $m_v$  for each node v is set as  $\frac{\theta}{k} * |\Gamma_v|$ , then scaled by  $\frac{k}{k'}$  (after k' is decided) to make the fairness metric cluster balanced. Unless specified otherwise, we set  $\theta$  as 0.5 in all experiments.

FairnessMacro FairnessADULTBANKDIABETESADULTBANKDIABETESADULTBANKDIABETESADULTBANKDIABETESk-means-PBS $0.58 \pm 0.09$ $0.65 \pm 0.08$ $0.77 \pm 0.03$ $91.4 \pm 0.6$ $95.5 \pm 2.2$ $93.3 \pm 1.4$ $70.8 \pm 6.0$ $66.2 \pm 5.6$ $60.5 \pm 1.7$ k-center-PBS-CC $0.56 \pm 0.12$ $0.65 \pm 0.08$ $0.77 \pm 0.03$ $90.5 \pm 1.3$ $93.6 \pm 3.4$ $94.8 \pm 0.9$ $64.0 \pm 22.2$ $58.0 \pm 20.4$ $51.0 \pm 21.8$ FairCenter $0.54 \pm 0.11$ $0.53 \pm 0.13$ $0.34 \pm 0.03$ $90.0 \pm 5.5$ $96.0 \pm 2.0$ $94.1 \pm 5.0$ $64.0 \pm 15.0$ $76.0 \pm 8.0$ $64.0 \pm 8.0$ Alg-PP $0.63 \pm 0.10$ $0.52 \pm 0.05$ $0.42 \pm 0.09$ $86.8 \pm 10.9$ $81.4 \pm 7.0$ $70.0 \pm 24.5$ $76.9 \pm 13.2$ $60.0 \pm 20.0$ Alg-AG $0.62 \pm 0.11$ $0.56 \pm 0.18$ $0.65 \pm 0.27$ $86.8 \pm 10.9$ $83.4 \pm 5.5$ $87.2 \pm 7.9$ $70.0 \pm 24.5$ $56.9 \pm 13.2$ $60.0 \pm 20.0$ P-PoF-Alg $0.60 \pm 0.08$ $0.59 \pm 0.14$ $0.53 \pm 0.15$ $86.8 \pm 10.9$ $87.6 \pm 7.7$ $92.8 \pm 9.1$ $70.0 \pm 24.5$ $66.9 \pm 18.0$ $80.0 \pm 24.5$ H-S $0.27 \pm 0.05$ $0.25 \pm 0.02$ $0.11 \pm 0.03$ $90.8 \pm 4.3$ $97.7 \pm 1.8$ $91.1 \pm 7.1$ $53.0 \pm 13.3$ $70.0 \pm 8.4$ $66.0 \pm 21.5$ Gonzalez $0.33 \pm 0.08$ $0.33 \pm 0.02$ $0.09 \pm 0.03$ $88.6 \pm 3.5$ $90.4 \pm 3.9$ $89.9 \pm 4.0$ $56.0 \pm 15.0$ $60.0 \pm 0.0$ $56.0 \pm 15.0$ LP-FAIR $0.19 \pm $										
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $		Normalized Cost			Fairness			Macro Fairness		
k-means-PBS $0.58 \pm 0.09$ $0.65 \pm 0.08$ $0.77 \pm 0.03$ $91.4 \pm 0.6$ $95.5 \pm 2.2$ $93.3 \pm 1.4$ $70.8 \pm 6.0$ $66.2 \pm 5.6$ $60.5 \pm 1.7$ k-center-PBS-CC $0.56 \pm 0.12$ $0.65 \pm 0.08$ $0.77 \pm 0.03$ $90.5 \pm 1.3$ $93.6 \pm 3.4$ $94.8 \pm 0.9$ $64.0 \pm 22.2$ $58.0 \pm 20.4$ $51.0 \pm 21.8$ FairCenter $0.54 \pm 0.11$ $0.53 \pm 0.13$ $0.34 \pm 0.03$ $90.0 \pm 5.5$ $96.0 \pm 2.0$ $94.1 \pm 5.0$ $64.0 \pm 15.0$ $76.0 \pm 8.0$ $64.0 \pm 8.0$ Alg-PP $0.63 \pm 0.10$ $0.52 \pm 0.05$ $0.42 \pm 0.09$ $86.8 \pm 10.9$ $91.3 \pm 7.4$ $88.1 \pm 7.0$ $70.0 \pm 24.5$ $72.9 \pm 16.3$ $60.0 \pm 20.0$ Alg-AG $0.62 \pm 0.11$ $0.56 \pm 0.18$ $0.65 \pm 0.27$ $86.8 \pm 10.9$ $83.4 \pm 5.5$ $87.2 \pm 7.9$ $70.0 \pm 24.5$ $56.9 \pm 13.2$ $60.0 \pm 20.0$ P-PoF-Alg $0.60 \pm 0.08$ $0.59 \pm 0.14$ $0.53 \pm 0.15$ $86.8 \pm 10.9$ $87.6 \pm 7.7$ $92.8 \pm 9.1$ $70.0 \pm 24.5$ $66.9 \pm 18.0$ $80.0 \pm 24.5$ H-S $0.27 \pm 0.05$ $0.25 \pm 0.02$ $0.11 \pm 0.03$ $90.8 \pm 4.3$ $97.7 \pm 1.8$ $91.1 \pm 7.1$ $53.0 \pm 13.3$ $70.0 \pm 8.4$ $66.0 \pm 21.5$ Gonzalez $0.33 \pm 0.02$ $0.09 \pm 0.03$ $88.6 \pm 3.5$ $90.4 \pm 3.9$ $89.9 \pm 4.0$ $56.0 \pm 15.0$ $60.0 \pm 0.0$ $56.0 \pm 15.0$ LP-FAIR $0.19 \pm 0.03$ $0.18 \pm 0.02$ $0.06 \pm 0.03$ $92.3 \pm 0.7$ $96.3 \pm 4.6$ $97.9 \pm 2.8$ $80.0 \pm 0.0$ $92.0 \pm 9.8$ $92.0 \pm 9.8$		Adult	Bank	DIABETES	Adult	Bank	DIABETES	Adult	Bank	DIABETES
	k-means-PBS	$0.58\pm0.09$	$0.65\pm0.08$	$0.77\pm0.03$	$91.4\pm0.6$	$95.5\pm2.2$	$93.3 \pm 1.4$	$\underline{70.8\pm6.0}$	$66.2\pm5.6$	$60.5\pm1.7$
FairCenter $0.54 \pm 0.11$ $0.53 \pm 0.13$ $0.34 \pm 0.03$ $90.0 \pm 5.5$ $96.0 \pm 2.0$ $94.1 \pm 5.0$ $64.0 \pm 15.0$ $76.0 \pm 8.0$ $64.0 \pm 8.0$ Alg-PP $0.63 \pm 0.10$ $0.52 \pm 0.05$ $0.42 \pm 0.09$ $86.8 \pm 10.9$ $91.3 \pm 7.4$ $88.1 \pm 7.0$ $70.0 \pm 24.5$ $72.9 \pm 16.3$ $60.0 \pm 20.0$ Alg-AG $0.62 \pm 0.11$ $0.56 \pm 0.18$ $0.65 \pm 0.27$ $86.8 \pm 10.9$ $83.4 \pm 5.5$ $87.2 \pm 7.9$ $70.0 \pm 24.5$ $56.9 \pm 13.2$ $60.0 \pm 20.0$ P-PoF-Alg $0.60 \pm 0.08$ $0.59 \pm 0.14$ $0.53 \pm 0.15$ $86.8 \pm 10.9$ $87.6 \pm 7.7$ $92.8 \pm 9.1$ $70.0 \pm 24.5$ $66.9 \pm 18.0$ $80.0 \pm 24.5$ H-S $0.27 \pm 0.05$ $0.25 \pm 0.02$ $0.11 \pm 0.03$ $90.8 \pm 4.3$ $97.7 \pm 1.8$ $91.1 \pm 7.1$ $53.0 \pm 13.3$ $70.0 \pm 8.4$ $66.0 \pm 21.5$ Gonzalez $0.33 \pm 0.02$ $0.09 \pm 0.03$ $88.6 \pm 3.5$ $90.4 \pm 3.9$ $89.9 \pm 4.0$ $56.0 \pm 15.0$ $60.0 \pm 0.0$ $56.0 \pm 15.0$ LP-FAIR $0.19 \pm 0.03$ $0.18 \pm 0.02$ $0.06 \pm 0.03$ $92.3 \pm 0.7$ $96.3 \pm 4.6$ $97.9 \pm 2.8$ $80.0 \pm 0.0$ $92.0 \pm 9.8$ $92.0 \pm 9.8$	$\operatorname{k-center-PBS-CC}$	$0.56\pm0.12$	$0.65\pm0.08$	$0.77\pm0.03$	$90.5\pm1.3$	$93.6\pm3.4$	$\underline{94.8\pm0.9}$	$64.0\pm22.2$	$58.0\pm20.4$	$51.0\pm21.8$
Alg-PP $0.63 \pm 0.10$ $0.52 \pm 0.05$ $0.42 \pm 0.09$ $86.8 \pm 10.9$ $91.3 \pm 7.4$ $88.1 \pm 7.0$ $70.0 \pm 24.5$ $72.9 \pm 16.3$ $60.0 \pm 20.0$ Alg-AG $0.62 \pm 0.11$ $0.56 \pm 0.18$ $0.65 \pm 0.27$ $86.8 \pm 10.9$ $83.4 \pm 5.5$ $87.2 \pm 7.9$ $70.0 \pm 24.5$ $56.9 \pm 13.2$ $60.0 \pm 20.0$ P-PoF-Alg $0.60 \pm 0.08$ $0.59 \pm 0.14$ $0.53 \pm 0.15$ $86.8 \pm 10.9$ $87.6 \pm 7.7$ $92.8 \pm 9.1$ $70.0 \pm 24.5$ $66.9 \pm 18.0$ $80.0 \pm 24.5$ H-S $0.27 \pm 0.05$ $0.25 \pm 0.02$ $0.11 \pm 0.03$ $90.8 \pm 4.3$ $97.7 \pm 1.8$ $91.1 \pm 7.1$ $53.0 \pm 13.3$ $70.0 \pm 8.4$ $66.0 \pm 21.5$ Gonzalez $0.33 \pm 0.02$ $0.09 \pm 0.03$ $88.6 \pm 3.5$ $90.4 \pm 3.9$ $89.9 \pm 4.0$ $56.0 \pm 15.0$ $60.0 \pm 0.0$ $56.0 \pm 15.0$ LP-FAIR $0.19 \pm 0.03$ $0.18 \pm 0.02$ $0.06 \pm 0.03$ $92.3 \pm 0.7$ $96.3 \pm 4.6$ $97.9 \pm 2.8$ $80.0 \pm 0.0$ $92.0 \pm 9.8$ $92.0 \pm 9.8$	FairCenter	$0.54\pm0.11$	$0.53\pm0.13$	$0.34\pm0.03$	$90.0\pm5.5$	$96.0\pm2.0$	$94.1\pm5.0$	$64.0 \pm 15.0$	$76.0\pm8.0$	$64.0\pm8.0$
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	Alg-PP	$0.63\pm0.10$	$0.52\pm0.05$	$0.42\pm0.09$	$86.8 \pm 10.9$	$91.3\pm7.4$	$88.1\pm7.0$	$70.0\pm24.5$	$\overline{72.9\pm16.3}$	$60.0\pm20.0$
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	Alg-AG	$0.62\pm0.11$	$0.56\pm0.18$	$0.65\pm0.27$	$86.8 \pm 10.9$	$83.4\pm5.5$	$87.2\pm7.9$	$70.0\pm24.5$	$56.9 \pm 13.2$	$60.0\pm20.0$
H-S Gonzalez $0.27 \pm 0.05$ $0.33 \pm 0.08$ $0.25 \pm 0.02$ $0.33 \pm 0.02$ $0.11 \pm 0.03$ $0.09 \pm 0.03$ $90.8 \pm 4.3$ $88.6 \pm 3.5$ $97.7 \pm 1.8$ $91.1 \pm 7.1$ $91.1 \pm 7.1$ $53.0 \pm 13.3$ $70.0 \pm 8.4$ $66.0 \pm 21.5$ $60.0 \pm 15.0$ LP-FAIR $0.19 \pm 0.03$ $0.18 \pm 0.02$ $0.09 \pm 0.03$ $92.3 \pm 0.7$ $96.3 \pm 4.6$ $97.9 \pm 2.8$ $97.9 \pm 2.8$ $80.0 \pm 0.0$ $92.0 \pm 9.8$ $92.0 \pm 9.8$	P-PoF-Alg	$0.60\pm0.08$	$0.59\pm0.14$	$0.53\pm0.15$	$86.8 \pm 10.9$	$87.6\pm7.7$	$92.8\pm9.1$	$70.0\pm24.5$	$66.9 \pm 18.0$	$80.0\pm24.5$
Gonzalez $0.33 \pm 0.08$ $0.33 \pm 0.02$ $0.09 \pm 0.03$ $88.6 \pm 3.5$ $90.4 \pm 3.9$ $89.9 \pm 4.0$ $56.0 \pm 15.0$ $60.0 \pm 0.0$ $56.0 \pm 15.0$ LP-FAIR $0.19 \pm 0.03$ $0.18 \pm 0.02$ $0.06 \pm 0.03$ $92.3 \pm 0.7$ $96.3 \pm 4.6$ $97.9 \pm 2.8$ $80.0 \pm 0.0$ $92.0 \pm 9.8$ $92.0 \pm 9.8$	H-S	$0.27\pm0.05$	$0.25\pm0.02$	$0.11\pm0.03$	$90.8\pm4.3$	$\textbf{97.7} \pm \textbf{1.8}$	$91.1\pm7.1$	$53.0 \pm 13.3$	$70.0\pm8.4$	$\overline{66.0\pm21.5}$
$ \text{LP-FAIR} \qquad 0.19 \pm 0.03 \ \ 0.18 \pm 0.02 \ \ 0.06 \pm 0.03 \ \ 92.3 \pm 0.7 \ \ \underline{96.3 \pm 4.6} \ \ 97.9 \pm 2.8 \ \ 80.0 \pm 0.0 \ \ 92.0 \pm 9.8 \ \$	Gonzalez	$\overline{0.33\pm0.08}$	$\overline{0.33\pm0.02}$	$0.09\pm0.03$	$88.6\pm3.5$	$90.4\pm3.9$	$89.9\pm4.0$	$56.0 \pm 15.0$	$60.0\pm0.0$	$56.0 \pm 15.0$
	LP-FAIR	$\textbf{0.19} \pm \textbf{0.03}$	$0.18 \pm 0.02$	$\textbf{0.06} \pm \textbf{0.03}$	$\textbf{92.3}\pm\textbf{0.7}$	$\underline{96.3\pm4.6}$	$\textbf{97.9} \pm \textbf{2.8}$	$80.0 \pm 0.0$	$\textbf{92.0} \pm \textbf{9.8}$	$\textbf{92.0} \pm \textbf{9.8}$

Table 6.1: Normalized cost and fairness comparison between LP-FAIR (ours) and competing baselines. The best and second-best values for each column are in bold and underlined, respectively. Our method outperforms or has performance comparable to the baselines in terms of the three evaluation metrics.

## 6.4.1 Performance

We present the results for our method LP-FAIR and competing baselines using the normalized cost (the lower the better) and fairness metrics (the higher the better). Table 6.1 shows the results produced by all algorithms. LP-FAIR has a significantly lower cost than the baselines, with a 34.5% lower cost than the best baseline (H-S) on average. Moreover, LP-FAIR consistently clusters points fairer. The best baselines for the fairness metric are k-means-PBS, FairCenter, and P-PoF-Alg, which have consistent performances overall. However, LP-FAIR outperforms them by 12.5% on average. Furthermore, our method generates results with less variance compared to the baselines, which shows the



stability of our algorithm.

Figure 6.1: Cost and fairness results, varying the number of clusters (k) for 30k random data points using all datasets and methods (better seen in color). LP-FAIR outperforms or achieves results comparable to the baselines for all k.

# 6.4.2 Varying Number of Clusters

Here, we evaluate the impact of the number of clusters (k) on cost and fairness. We vary the value of k from 3 to 10 and choose 30k random points from each dataset. Figure 6.1 shows the results. LP-FAIR achieves the best results for most values of k. In general, the cost decreases as the number of clusters increases. The exceptions are for the cases where the number of clusters generated is lower than expected. For fairness metrics, LP-FAIR outperforms or achieves comparable results to all baselines for all datasets. The values for Fairness do not present a clear trend based on k. We also evaluate the impact of the number of clusters on the macro fairness metric. Figure 6.1 shows that LP-FAIR performs better or has comparable results to the baselines. Macro Fairness tends to decrease as the number of clusters increases. That is because k affects the cluster imbalance. Eventually, this makes some clusters less fair and decreases the average score.

# 6.4.3 Quality of Clusters

Cluster quality is also a crucial metric to be considered. Table 6.2 and Table 6.3 show generated cluster counts and imbalance, respectively, for all methods and datasets. Some algorithms use less number of clusters even though the expected number is k = 5, which makes their cost higher compared to the one generating close to 5. Cluster imbalance is also critical as imbalanced clusters would result in clusters with a small number of elements in them. The elements in those clusters are unlikely to be fair, and this will affect individual fairness as well as macro fairness. LP-FAIR generates more balanced clusters (low imbalance) compared to the baselines resulting in better fairness. The primary reason for our algorithm's emphasis on balanced clusters is rooted in our fairness notion, which demands an equitable allocation of similar individuals for each individual.

**Summary.** A few key observations from the above experiments are as follows: (1) Our method (LP-FAIR) produces clusters that are individually fair to more than 95.5% points and achieves 88% Macro Fairness on average, outperforming the baselines in most of the settings; (2) LP-FAIR also produces lower cost or distance than competing baselines in all settings; (3) LP-FAIR generates better clusters in terms of numbers and imbalance. This makes LP-FAIR clusters less costly and more fair compared to the existing competitors.

Table 6.2: The mean and standard deviation of the number of clusters generated by the methods (with k = 5). Generating fewer clusters generally leads to higher costs (see Table 6.1).

	Adult	Bank	DIABETES
k-means-PBS	$3.2\pm0.52$	$3.1\pm0.35$	3.0 + -0.33
k-center-PBS-CC	$5.0 \pm 0.0$	$5.0 \pm 0.0$	$5.0 \pm 0.0$
FairCenter	$5.0 \pm 0.0$	$5.0 \pm 0.0$	$4.0\pm0.63$
Alg-PP	$2.0 \pm 0.0$	$3.2\pm0.98$	$2.0 \pm 0.0$
Alg-AG	$2.0 \pm 0.0$	$2.8\pm0.4$	$1.8\pm0.4$
P-PoF-Alg	$2.0 \pm 0.0$	$4.0\pm1.67$	$2.0 \pm 0.0$
H-S	$4.6\pm0.49$	$4.6\pm0.49$	$4.6\pm0.49$
Gonzalez	$5.0 \pm 0.0$	$5.0 \pm 0.0$	$5.0 \pm 0.0$
LP-FAIR	$5.0\pm0.0$	$5.0\pm0.0$	$4.8\pm0.4$

Table 6.3: The mean and standard deviation of cluster imbalance. Imbalanced clusters result in small clusters where the members might not be individually fair. LP-FAIR generates clusters with lower imbalance compared to the baselines.

	Adult	Bank	DIABETES
k-means-PBS	$49.7\pm31.6$	$47.5\pm27.0$	$75.6 \pm 19.2$
k-center-PBS-CC	$40.7 \pm 13.7$	$49.4 \pm 17.1$	$64.6 \pm 17.8$
FairCenter	$28.4\pm5.8$	$30.3\pm5.2$	$35.1\pm6.3$
Alg-PP	$51.4\pm6.5$	$37.1 \pm 13.6$	$63.8 \pm 14.8$
Alg-AG	$51.4\pm6.5$	$42.7 \pm 13.8$	$54.4 \pm 28.7$
P-PoF-Alg	$51.4\pm6.5$	$29.1 \pm 15.5$	$37.6\pm23.0$
H-S	$44.5\pm9.5$	$34.9\pm6.2$	$40.9 \pm 13.4$
Gonzalez	$40.9\pm10.0$	$33.3\pm5.7$	$37.1\pm10.0$
LP-FAIR	$17.9\pm2.8$	$17.6\pm2.1$	$15.8\pm4.8$

The runtime for Algorithm 4 is equivalent to solving a theoretically polynomial linear program that is practically lightning-fast. On the other hand, the runtime for Algorithm 5 is the summation of Vanilla Clustering and Individually Fair Assignment algorithms' runtimes.

Our implementation employs the scipy.optimize.linprog Python module. To account for the randomization in our algorithm, we conducted ten trials, but the best performance was consistently achieved within four trials. A single trial of our algorithm required 36.7 seconds to complete the Adult dataset. The running time for LP-FAIR and the baselines on the Adult dataset is summarized in Table 6.5.

	Adult
FairCenter	16.5s
Alg-PP	11.8s
Alg-AG	9.68s
P-PoF-Alg	9.55s
H-S	7.92s
Gonzalez	7.91s
LP-FAIR $(10 \text{ trials})$	367.5s
LP-FAIR $(4 \text{ trials})$	147s
LP-FAIR $(1 \text{ trial})$	36.7s

Table 6.4: The running times of LP-FAIR and baselines on the Adult dataset. We run our randomized algorithm ten times, but the best performance is achieved with at most four trials.

In addition, we tested the scalability of our algorithm on larger datasets by sampling 1000 data points from the Adult dataset and executing it under identical settings and hardware. The results showed that a single trial of LP-Fair required 165.88 seconds to complete. As highlighted in Table 6.5, our algorithm's practical scalability is linear with respect to the data point size.

# 6.6 Additional Experiments

# 6.6.1 Random Features

We randomly select two distance and two fairness attributes, with five different random selections, and run each experiment five times. Table 6.6.1 shows that selecting random features does not alter the performance of our method compared to Table 6.1, and LP-FAIR is the best performing or has comparable results to the baselines.

	Nor	rmalize	Fairne	ess		
	Adult	Bank	DIABETES	Adult	Bank	DIABETES
FairCenter	0.783	0.535	0.874	91.93	90.90	92.43
Alg-PP	0.632	0.415	0.852	76.60	77.38	91.00
Alg-AG	0.643	0.432	0.953	78.55	77.75	91.30
P-PoF-Alg	0.525	0.382	0.856	76.60	75.15	94.20
H-S	0.415	0.526	0.773	92.55	93.60	94.90
Gonzalez	0.472	0.359	0.548	88.50	91.20	90.50
LP-FAIR	0.302	0.232	0.189	93.48	<u>92.18</u>	96.43

Table 6.5: Normalized cost and fairness comparison between LP-FAIR (ours) and competing baselines with random feature selections. The best and second-best values for each column are in bold and underlined, respectively. Our method outperforms or has performance comparable to the baselines in terms of fairness and cost.

# 6.6.2 Neglecting Fairness Constraint

We run experiments by setting  $m_v = 0$  in Equation 6.3 to see the effect of our fairness constraint. Setting  $m_v = 0$  essentially makes all points fair after the first cluster assignments, so the cost will be minimized with K-means. Table 6.6 shows that setting  $m_v = 0$ , as expected, decreases the normalized cost, whereas the fairness performance becomes much worse.

	Nor	rmalize	Fairness			
	Adult	Bank	DIABETES	Adult	Bank	DIABETES
LP-FAIR $(m_v = 0)$	0.182	0.159	0.047	89.3	92.2	93.4
LP-FAIR	0.194	0.176	0.057	92.3	96.3	97.9

Table 6.6: The effect of removing fairness constraint from LP-FAIR. The better performances are in bold. As expected, our algorithm makes the clusters more costly while having more individually fair clusters.

# 6.7 Conclusions

We have studied the k-clustering problem with individual fairness constraints. Our notion of fairness is defined in terms of a feature-based similarity among points and guarantees that each point will have a pre-defined number of similar points in their cluster. We have provided an algorithm with probabilistic approximation guarantees for optimizing the cluster distance as well as ensuring fairness. Finally, the experimental results have shown that our proposed algorithm can produce 34.5% lower clustering cost and 12.5% higher individual fairness than previous works on average.

**Limitations:** Our definition of individual fairness might be limited and specific to certain application scenarios. We aim to generalize this definition in future work.

**Broader Impact Statement:** Our work is mainly theoretical in nature and we do not anticipate any direct negative societal impact. However, it is possible for third parties to use the techniques developed in our paper and declare a clustering as fair without proper evaluation on the domain.

# Chapter 7

# Robustness of Human Decision Making under Risk

# 7.1 Introduction

Prospect Theory has played a crucial role in enhancing our understanding of decisionmaking processes when faced with risk and uncertainty. Developed by Kahneman and Tversky [278], this theoretical framework has provided valuable insights into the biases and preferences exhibited by individuals when making choices involving potential gains or losses. Extensive research [279, 280] has focused on comprehending individual decisionmaking within the context of Prospect Theory, resulting in significant advancements in the field. Various domains, such as economics [281, 282] and politics [283], have widely studied Prospect Theory to improve their decision-making processes, given their high level of risk. However, the exploration of decision-making in collective settings, known as Group Prospect Theory, has received relatively less attention, with most scenarios involving decisions under risk being made by groups of individuals rather than individuals alone. This study seeks to bridge this gap by investigating the dynamics of decisionmaking within groups and examining the implications for risk preferences and behavior.

Understanding decision-making within groups is crucial, as many real-world scenarios involve collective decision-making processes [284, 285, 286]. Whether it's a board of directors making strategic business decisions or a committee of policymakers determining public policies, the dynamics of group decision-making significantly impact the outcomes. Group Prospect Theory aims to uncover how the interactions, influence, and social dynamics within a group shape decision-making behavior, and whether they differ from individual decision-making.

Furthermore, cognitive biases and the concept of transactive memory are relevant to group decision-making [287, 288]. Cognitive biases [289, 290] refer to systematic deviations from rationality that individuals exhibit when processing information or making judgments. These biases can impact decision-making outcomes within groups, as individuals bring their biases into the collective decision-making process. Additionally, transactive memory [291] refers to the shared knowledge and information storage system within a group, where individuals rely on each other's expertise and memory to make decisions. Understanding how cognitive biases and transactive memory influence group decision-making is an essential aspect of studying Group Prospect Theory.

To gain insights into Group Prospect Theory and provide empirical results, this study employs a combination of experimental methods and behavioral analysis. Human experiments are conducted to observe decision-making behavior as individuals transition from making choices in isolation to making decisions within a group setting. By carefully designing and controlling the experimental conditions, we aim to measure the behavioral shifts that occur during this transition and identify the factors that influence risk preferences and decision outcomes within a group context. The details of experiments will be discussed in Human Experiments Section 7.2.

In addition to examining the behavior changes associated with group decision-making,

this study also explores the robustness of groups in the face of potential attacks facilitated by artificial intelligence (AI). As AI systems become more sophisticated and capable of manipulating information and influencing decision-making processes, it is crucial to understand how groups respond to such attacks and whether their decision outcomes are compromised. By investigating the vulnerability of groups to AI-based attacks, we can assess the potential risks and develop strategies to safeguard group decision-making processes.

This paper aims to significantly contribute to the literature on Group Prospect Theory, shed light on the behavioral dynamics of decision-making within groups, and raise awareness regarding the potential risks and vulnerabilities posed by AI in collective decision-making processes. Our contributions can be summarized as follows:

- ▷ We conduct carefully designed human experiments to understand human decisionmaking processes in both individual and group settings.
- ▷ We extensively analyze the experimental results to gain insights into the behavioral shifts that occur as individuals transition from making decisions in isolation to making decisions within a group setting.
- ▷ We explore the robustness of groups in the face of AI attacks by designing an evasion attack model, which enables us to identify vulnerabilities and assess the resilience of groups against AI attacks.
- ▷ We develop a generative model for human behavior to increase the efficiency and effectiveness of attacks.

# 7.2 Human Experiments

Our experimental setup is meticulously designed to investigate human decisionmaking under conditions of risk in both individual and group settings. The experiment



Figure 7.1: Overview of our human experiments. The individual phase (IND) involves each participant independently responding to a series of m issues. Following this, groups are formed and the group phase begins. In the pre-discussion (PRE) phase, each group member individually answers an issue, engages in group discussions, and subsequently provides their responses again in the post-discussion (POST) phase. This cycle of pre-discussion and post-discussion repeats for a total of n iterations. At the end of each cycle, each member indicates the perceived influence from other group members.

consists of three distinct phases: Individual (IND), Pre-discussion (PRE), and Postdiscussion (POST). Figure 7.1 demonstrates the phases and their transitions.

During the Individual phase (IND), participants are presented with a series of m issues, also known as gambles, and are asked to make independent decisions on each issue. An example issue is illustrated in Figure 7.2. This phase allows us to observe and analyze individual behaviors in risk-based decision-making scenarios.

Following the IND phase, participants form groups consisting of three or four individuals, and the group phases begin. The group phase comprises two phases: PRE and POST. In the PRE phase, participants individually respond to similar issues before engaging in group discussions. Subsequently, they exchange opinions and discuss their respective choices with other group members.

In the POST phase, participants answer the same set of questions again, taking into consideration the information shared and opinions expressed during the group discussion.

#### **Option A:**

A gamble in which you may win \$4132 with probability P=0.07 or lose \$677 with probability (1-P)=0.93 Option B:

A gamble in which you may win \$553 with probability P=0.89 or lose \$7569 with probability (1-P)=0.11

Figure 7.2: An issue example. Each issue contains a choice dilemma of a binary gamble selection between option A and B. Each option includes gain and loss with corresponding probabilities. Individuals record their answers by selecting either option.

This two-phase modeling enables us to examine how group discussions influence immediate decision-making outcomes. Group phase cycles m times, and after each cycle, all group members record the extent of influence they perceive from the other group members, including themselves. This record generates an influence matrix, providing valuable insights into the dynamics of influence within the group.

By employing this experimental process, we aim to gain a deeper understanding of how individuals' decision-making behavior evolves when transitioning from independent decision-making to decision-making within a group. The IND phase provides a baseline for individual risk preferences, while the group phases offer insights into how group discussions and information sharing influence decision outcomes. The analysis of the influence matrix enhances our understanding of the dynamics of influence within the group and its impact on decision-making processes.

# 7.2.1 Datasets

We conducted experiments at two distinct locations: the University of California, Santa Barbara (UCSB) and Fort Bragg (FB). The statistics for each dataset from the human experiments are presented in Table 7.1. During the group phase, individuals form groups of three or four people.

	#Individuals	#Groups	#Individual Issues (m)	#Group Issues (n)
UCSB	107	30	30	12
FB	29	8	30	12

Table 7.1: The statistics of datasets from human experiments.

# 7.3 Preliminaries

#### 7.3.1 Prospect Theory



The classic approach to studying risk preference behavior involves presenting individuals with a series of choice dilemmas. In our experiments, participants are given a set of numbered issues on paper, where they must select their preferred "mixed gamble" from options X and Y. These options consist of pairs of outcomes, with X, represented as  $((x_1, p_1), (x_2, p_2))$  and Y as  $((y_1, q_1), (y_2, q_2))$ . Here,  $x_1$  and  $y_1$  represent potential gains in monetary value, while  $x_2$  and  $y_2$  denote potential losses. Choosing X leads to a payment of  $x_1$  with probability  $p_1$  or a debt of  $x_2$  with probability  $p_2$ , and the same applies to Y.

Cumulative Prospect Theory (CPT) proposes that individuals maintain an internal valuation function, denoted as V(X), which assigns psychological value to risky outcomes known as prospects. The valuation function for a mixed gamble X can be decomposed as follows:

$$V(X) = v^{+}(x_{1})w^{+}(p_{1}) + v^{-}(x_{2})w^{-}(p_{2})$$
(7.1)

where the value function  $v^{+/-}(\cdot)$  and the weighting function  $w^{+/-}(\cdot)$  depending on how the individual perceives the outcome relative to a reference point, such as a gain or loss of capital or social status.

To determine the likelihood that an individual prefers X over Y, denoted as p(X, Y), we assume it can be calculated using the Logistic transformation of the negative, weighted exponential of the difference in valuations between X and Y (i.e., Softmax activation function). The weight is denoted by the parameter rho, and p(X, Y) is given by the following equation:

$$p_{\theta}(X,Y) = 1/(1 + exp(-\rho\{V(X|\theta) - V(Y|\theta)\}))$$
(7.2)

Here,  $\theta$  represents a parameter vector, including values such as  $\alpha, \beta, \lambda, \gamma^+$ , and  $\gamma^$ which determine the evaluation of v and w in Equation 7.1.  $\rho$  reflects the extent to which prospect valuations determine choice behavior instead of random choice.

Extensive empirical analysis comparing various functions for v, w, and p has been conducted [292]. These functions, based on the suggestions of [292], include the power value function (Equations 7.3 & 7.4), the Prelec [293] weighting function (Equations 7.5 & 7.6), and the Logistic stochastic choice function (Equation 7.2). The parameter vector  $\theta = (\alpha, \beta, \lambda, \gamma^+, \gamma^-)$  determines the shape of v and w according to the following equations:

$$v^+(x) = x^\alpha, \alpha > 0 \tag{7.3}$$

$$v^{-}(x) = -\lambda |x|^{\beta}, \ \lambda, \beta > 0 \tag{7.4}$$

$$w^{+}(p) = exp(-(-\ln p)^{\gamma^{+}}), \gamma^{+} > 0$$
(7.5)

$$w^{-}(p) = exp(-(-\ln p)^{\gamma^{-}}), \gamma^{-} > 0$$
(7.6)

Expected utility theory is a special case of CPT where  $v^{+/-}$  and  $w^{+/-}$  are the identity function, represented by a parameter vector  $\theta$  of all ones. It's worth noting that the valuation function for a mixed gamble X is the same in both original prospect theory [278] and cumulative prospect theory [279] for mixed binary gambles. However, for general prospects with multiple gains or losses, the function  $w^{+/-}$  depends on a cumulative weighting function.

Applying CPT as a predictive model of choice behavior involves associating or learning a parameter vector  $\theta = (\alpha, \beta, \lambda, \gamma^+, \gamma^-)$  for each individual, sub-population, or population to capture systematic risky choice preferences. One of the significant advantages of the CPT framework is the ability to interpret behavior based on these parameters. Based on Nilsson et.al. [294], we provide a simplified interpretation below:

- $\triangleright \alpha (resp. \beta)$  reflects the sensitivity to gain (loss) outcomes.
- $\triangleright \lambda$  reflects the perceived impact of loss relative to gain.
- $\triangleright \gamma^+(resp. \gamma^-)$  reflects the degree to which gain (loss) probabilities are over- or underweighted.

#### **Prospect Theory Parameters Calculation**

Due to the non-differentiable and non-convex nature of v and w, we employ a gridsearch method to find optimal parameters for each individual. However, optimizing six parameters using grid search can be challenging. To address this issue, we follow a similar approach in [281] and simplify the model by setting  $\alpha = \beta$ ,  $\gamma^+ = \gamma^-$ . Our experimental results indicate that setting  $\rho = 1$  yields favorable outcomes. With this simplification, our model reduces the number of parameters to be optimized to three:  $\theta = (\alpha, \lambda, \gamma)$ .

We apply grid search over  $\alpha \in (0, 1]$  with a step size of 0.05,  $\lambda \in [0.25, 5]$  with a step size of 0.25, and  $\gamma \in (0, 1.5]$  with a step size of 0.05. We choose  $\theta$  that maximizes the expected value of the valuation function with Equation 7.2 over multiple issues. Since we have three phases (i.e., IND, PRE, POST), we calculate prospect theory parameters for every phase and denote them as  $\theta_i^{\text{IND}} = (\alpha_i^{\text{IND}}, \lambda_i^{\text{IND}}, \gamma_i^{\text{IND}})$  for participant *i* during the IND phase (similarly for PRE and POST).

#### **Deciding Risky Option**

During the preparation of our issues, we deliberately avoided specifying which option is riskier to prevent bias. However, for the purpose of analysis, we need to determine the risky option for each issue. To address this, we define the riskier option as the one with a higher variance.

Let's consider an option  $X = ((x_1, p_1), (x_2, p_2))$  as defined before. The variance of this option can be calculated using the following formula:

$$VAR(X) = p_1 * (x_1 - E[X])^2 + p_2 * (-x_2 - E[X])^2$$

where E[X] represents the expected value of the choice, given by:

$$E[X] = p_1 * x_1 - p_2 * x_2$$

Note that  $x_2 > 0$  represents a loss, so the formulation includes  $-x_2$  in the calculation of variance and expected value. By applying this formulation to the issue example shown in Figure 7.2, we find that the first option has a variance of approximately 777100, while the second option has a higher risk with a variance of approximately 4819054.

# 7.3.2 Behavior Calculation

One of the main objectives of this study is to investigate the changes in human behavior following discussions with other individuals. To quantify this change, it is necessary to calculate the distance between two behaviors. However, directly calculating the distance using Prospect Theory (PT) parameters may lack intuitive meaning and may not provide a comprehensive representation. To address this issue, we propose a measure of individual behavior based on their PT parameters:  $b_i = b(\theta_i = (\alpha_i, \lambda_i, \gamma_i))$ , which involves the following steps:

- ▷ Randomly sample a sequence of gambles (i.e., issues) to create a random gamble sequence.
- Profile each individual over the gamble sequence to obtain their respective valuation sequence.

Subsequently, we can calculate the pairwise distance between two individuals by computing the cosine distance between their valuation sequences (i.e., behaviors). The use of cosine distance allows us to capture the orientation of behaviors rather than solely focusing on the absolute magnitude.

## 7.3.3 Evasion Attack

In our framework, we incorporate machine learning to enhance the analysis and increase the predictive power and flexibility of our models. Additionally, by employing machine learning, we can investigate potential attacks by AI on the group decision-making process within our experiments. In the literature, two primary types of attacks on machine learning models are discussed: poisoning attacks and evasion attacks [295]. These attacks are frequently examined in various applications, particularly in high-stakes scenarios like autonomous vehicles [296]. Poisoning attacks occur during the training phase of the machine learning algorithm, while evasion attacks take place during the inference phase (after training). In this project, our focus is on studying evasion attacks, as we aim to analyze the impact of AI on group decisions after modeling the group using machine learning models (which will be described later).

Let  $d_{i,t}$  represent the decision of participant *i* on issue *t*. The decision can be determined through influence mechanisms, previous decisions, or the decisions of other group members. In an evasion attack, an attacker acting as another group member aims to influence the decision of participant *i*. The objective of a successful attack is to flip the decision of participant *i* to an alternative choice. For instance, in a scenario where all group members have reached a consensus with the same decision, the attack would involve flipping the decisions of each member while maintaining consensus, but to a different choice. A similar approach has been explored in the context of Byzantine faults [297].

# 7.4 Analysis and Experiments

# 7.4.1 Prospect Theory Parameters for Individuals

We calculate the prospect theory parameters of individuals based on their answers during the individual (IND), pre-discussion (PRE), and post-discussion (POST) phases. The calculation process is described in the Preliminary section (Prospect Theory Parameters Calculation).

$\alpha = \beta$				$\lambda$			$\gamma^+ = \gamma$	/	
Model	IND	PRE	POST	IND	PRE	POST	IND	PRE	POST
UCSB	0.28	0.41	0.41	1.47	1.22	0.78	0.58	0.63	0.58
$\mathbf{F}\mathbf{B}$	0.36	0.56	0.54	1.42	1.16	1.43	0.50	0.60	0.56

Table 7.2: The average statistics of PT parameters for IND/PRE/POST across all individuals.

Table 7.2 shows the average statistics of the prospect theory (PT) parameters of individuals in different phases.  $\alpha$  ( $\beta$ ) increases from IND to PRE (i.e., sensitivity to gain/loss increases) for both datasets. PRE and POST  $\alpha$  values remain similar. The  $\lambda$ parameter decreases from IND to PRE for both datasets. For the UCSB dataset, the  $\lambda$ value further decreases in POST, while it becomes similar to IND for the FB dataset. Across all measurements, the parameter  $\gamma$  remains relatively stable.

## 7.4.2 Understanding Influence in a Group

After each cycle of the group discussion phase, participants are asked to assign influence values to themselves and other group members. This question aims to understand the explicit influence obtained by each member of the group.

To analyze the influence mechanism, we examine the influence matrix, which can be represented as a directed graph. The direction of the edges in the graph indicates the flow of influence. Table 7.3 provides statistics on the influence matrices, focusing on their ergodic structure and the presence of influential leaders. An influential leader is defined as a node that is reachable from all other nodes in the graph. Unfortunately, the percentage of groups with ergodic influence matrices is quite low for both datasets. Moreover, no group exhibits at least one influential leader across all issues.

We observe that there is no clear pattern in the explicit influence mechanism, likely due to noise introduced by human participants. The assignments of influence are in-

	%Ergodic	% Influential Leader	$\# {\rm Groups \ Ergodic} \ \forall$	#Groups Ergodic $\exists$	$\# {\rm Groups} \ {\rm Leader} \ \forall$	#Groups Leader $\exists$
UCSB FB	15% 39.6%	37.8% 55.2%	$\begin{array}{c} 3 \ / \ 30 \\ 0 \ / \ 8 \end{array}$	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	$\begin{array}{c} 0 \ / \ 30 \\ 0 \ / \ 8 \end{array}$	27 / 30 7 / 8

Table 7.3: Statistics of influence matrices from human experiments.  $\forall$  denotes every issue, and  $\exists$  denotes at least one issue.

consistent across different issues and group members. In some cases, participants assign influence only to themselves, resulting in an identity matrix.

We introduce the concept of "stubbornness" to describe the behavior of individuals assigning self-influence. To quantify this behavior, we calculate the average of diagonal entries in the influence matrix within a group and then compute the overall average across all groups.

Figure 7.3 presents the stubbornness measure across group issues for both the UCSB and FB datasets. The results indicate that participants from the UCSB dataset exhibit a higher level of stubbornness, indicating a stronger inclination to maintain their own opinions. In contrast, participants from the FB dataset tend to conform more to the group's opinion. Additionally, the level of stubbornness among UCSB participants tends to increase as the experiment progresses.

To address this issue, we also consider capturing the implicit influence from the data. During the experiments, participants provide their pre-discussion (PRE) and post-discussion (POST) choices, which may differ due to the group discussion. We have encountered cases where individuals do not assign explicit influence but change their answers after considering the choices of other group members. We refer to this phenomenon as an implicit influence.

Both explicit and implicit influences are utilized as features in our machine-learning framework to predict the behavior of individuals and groups. We will provide a more detailed explanation of these features later on.



Figure 7.3: Stubbornness level of individuals over group issues. UCSB participants are more stubborn than FB participants.

# 7.4.3 Predictive Power

#### **Prospect Theory**

Prospect theory parameters can be used to predict future choices by individuals using the valuation formulation described in the Preliminaries. We consider IND, PRE, and POST as different models and learn different parameters for each of them. Additionally, PT provides interpretable results, allowing us to analyze the risky behavior of individuals by examining the  $\alpha$ ,  $\lambda$ , and  $\gamma$  parameters.

We compare PT models to different baselines:

- ▷ Neural Net: Choice based on the learned weights of neural net that takes only issue parameters (e.g.,  $x_1$ ,  $x_2$ , and  $p_1$ ) as input.
- $\triangleright$  Utility: Rational choice based on utility.
- $\triangleright$  Max Gain: Choice based on maximum gain.
- $\triangleright$  Min Loss: Choice based on minimum loss.

Table 7.4 shows the prediction of choices based on different models. The PT model
Model	FB (IND)	FB (PRE)	FB (POST)	UCSB (IND)	UCSB (PRE)	UCSB (POST)
$\mathbf{PT}$	0.747	0.73	0.782	0.723	0.692	0.699
Neural Net	0.749	0.724	$0.718^{*}$	$0.704^{*}$	0.678	0.682
Utility	$0.56^{**}$	$0.606^{**}$	$0.606^{**}$	$0.56^{**}$	$0.639^{**}$	$0.645^{**}$
Max Gain	$0.396^{**}$	$0.399^{**}$	$0.417^{**}$	$0.393^{**}$	$0.431^{**}$	$0.469^{**}$
Min Loss	$0.4^{**}$	$0.397^{**}$	$0.379^{**}$	$0.374^{**}$	0.385**	0.418**

outperforms or has comparable results to the baselines for different datasets and models.

 $p^{**}p < 0.01$   $p^{*} < 0.05$ 

Table 7.4: N-fold cross-validation accuracy results for different models. The best and second-best models are shown in bold and underlined, respectively. The PT model performs best. p-values are calculated with paired t-tests between PT and baselines. The neural net model is closest to PT in terms of accuracy.

Even though prospect theory has the advantage of providing interpretable parameters, its accuracy performance may be limited due to its design and lack of flexibility in using different feature sets, such as influence mechanisms or unseen structures of group member behaviors. To overcome this issue, we can design a machine-learning model that offers more flexibility and ease of use. However, a potential disadvantage of using a machine learning model is the loss of interpretability that prospect theory provides.

#### **POST** Decision Predictor using Machine Learning

We have developed a one-layer MLP model with a hidden layer size of 128 and a logistic activation function to predict the POST choices of individuals in the UCSB dataset. The model takes into account the individual's PRE choice, as well as explicit and implicit influences from group members. The model consists of five features:

- ▷ (explicit) Average influence from group members who made high-risk choices in the PRE phase.
- ▷ (explicit) Average influence from group members who made low-risk choices in the PRE phase.
- $\triangleright$  (implicit) Number of group members who made high-risk choices in the PRE phase.

- $\triangleright$  (implicit) Number of group members who made low-risk choices in the PRE phase.
- $\triangleright$  PRE choice of the individual.

The model demonstrates an accuracy of 75% in predicting no change for low-risk choices (i.e., when the individual chooses low-risk options in both the PRE and POST phases). Similarly, it achieves an accuracy of 84% in predicting no change for high-risk choices. The accuracy significantly improves to 97% when predicting a change from low risk in the PRE phase to high risk in the POST phase, and it reaches 94% when predicting a change from high risk to low risk.



Figure 7.4: Contributions of different features in making predictions in POST choice for UCSB dataset.

Figure 7.4 provides insight into the importance of each feature in making predictions for the UCSB dataset. The importance of each feature is determined by assessing how the model's performance deteriorates when the feature is excluded. The results highlight the significant influence of both explicit and implicit factors associated with group members who selected high-risk options during the PRE phase in predicting POST choices.

#### 7.4.4 Behavioural Changes between IND - PRE - POST

We analyze the changes occurring between IND, PRE, and POST based on choices on issues, consensus, and behavior valuation curves, as described in the Preliminaries.



Figure 7.5: Confusion matrix showing the shift between low and high-risk choices based on different models. Significant changes occur from IND to POST. The predictions are based on the PT models.

#### **Choice Shift**

Figure 7.5 demonstrates the shift of individuals towards less or more risky choices for the UCSB dataset based on the issues asked during the individual phase. Please note that the choices in the figure represent predictions based on the PT models. While there is a significant change from IND to PRE and POST, the change from PRE to POST is relatively less significant. The change indicates that human behavior towards risk evolves through group discussion.

#### Consensus Shift

Figure 7.6 illustrates the consensus ratio over groups for all issues, comparing the PRE and POST choices in the UCSB dataset. The figure clearly shows that POST



Figure 7.6: Consensus ratio over groups for all issues. The groups tend to form a consensus.

choices exhibit higher consensus compared to the PRE choices, which can be attributed to the group discussion. Furthermore, it is observed that the group tends to form a consensus even in the PRE choices (before discussion), as their behaviors become more aligned with each other over time.

#### **Behavior Shift**

Figure 7.7 displays the average pairwise cosine distance between individuals within each group, comparing the IND, PRE, and POST parameters in the UCSB dataset. As anticipated, in the majority of groups, the pairwise distance decreases from the individual settings to the group settings (IND  $\rightarrow$  PRE  $\rightarrow$  POST). This observation indicates that there is a shift in behavior towards consensus as a result of the group discussion. The pairwise IND distances are generally greater than the pairwise PRE distances, which, in turn, are greater than the pairwise POST distances.



Figure 7.7: Average pairwise distance between individuals in each group for IND, PRE and POST parameters. For most groups, pairwise IND distances > pairwise PRE distances > pairwise POST distance. This shows that the behavior of individuals shifts towards consensus in a group setting.

#### 7.4.5 Robustness of Groups

One of the major concerns in group decision-making is the robustness of decisions. It raises the question of whether group decisions can be significantly different under different circumstances. Specifically, we are interested in investigating the impact of adding additional members to a group and how it might alter the decisions made. To measure this metric of group robustness, we have designed a framework to perform evasion attacks on each group. In order to formulate our evasion strategy, we first need to develop three different models that capture various aspects of group behavior. It is important to note that the experiments conducted in this section focus exclusively on the UCSB dataset.

#### **Consensus Predictor**

We have developed a predictor that determines whether a consensus has been reached within a group. For this task, we chose a one-layer Multi-Layer Perceptron (MLP) with a hidden layer size of 128 and a logistic activation function. The problem can be framed as a binary classification, where class 0 represents no consensus and class 1 represents consensus.

During training, we used six different features as input for the predictor. These features are similar to the ones used in the POST choice predictor, as we found them to be useful in capturing relevant information. The six features are as follows:

- ▷ (explicit) Average influence from group members who made high-risk choices in the PRE phase.
- ▷ (explicit) Average influence from group members who made low-risk choices in the PRE phase.
- $\triangleright$  (implicit) Number of group members who made high-risk choices in the PRE phase.
- $\triangleright$  (implicit) Number of group members who made low-risk choices in the PRE phase.
- ▷ Number of group members who made high-risk choices in the PRE phase based on PT parameters from IND.
- ▷ Number of group members who made low-risk choices in the PRE phase based on PT parameters from IND.

The last two parameters provide additional prediction power by considering the status of group members during the individual phases. Our model achieves an accuracy of 89% in predicting consensus.

To assess the importance of each feature, we conducted an analysis where we removed each feature from the model and calculated the accuracy of the modified model. The difference in accuracy before and after removing a feature indicates its importance. Figure 7.8 illustrates the feature importance in predicting consensus. The explicit and implicit low-risk influence features are found to be the most effective in predicting consensus compared to the other features. It is worth noting that each feature contributes to improving the performance of the model.



Figure 7.8: Importance of features in predicting consensus. Explicit and implicit low-risk influence are effective features.

Overall, the machine learning-based consensus predictor offers flexibility in incorporating various features and demonstrates the ability to capture patterns in the data to predict whether a group has reached a consensus.

#### Group Prospect Theory Parameter Predictor

Group Prospect Theory (GPT) parameters are defined as PT parameters in a group in which issues they had a consensus. Firstly, using this formulation, we calculated ground-truth GPT parameters for each group.

Later on, we developed supervised MLP regressors to estimate GPT parameters. Each regressor is responsible for predicting one of the GPT parameters:  $\alpha$ ,  $\lambda$ , and  $\gamma$ . The regressors are designed as MLP regressors with similar hyperparameters to the consensus predictor.

The input features used in the regressors are similar to those used in the consensuspredictor. In addition, two additional parameters are included based on the model:▷ The minimum IND-PT parameter within the group.

 $\triangleright$  The maximum IND-PT parameter within the group.

Including the minimum and maximum parameters ensures permutation invariance in the model. It is important to note that we use the IND-PT parameters (e.g.,  $\alpha$  IND-PT parameter) to predict the corresponding GPT parameters (e.g.,  $\alpha$  GPT parameter).

The model for  $\alpha$  achieves a Mean Squared Error (MSE) of 0.024 and an  $R^2$  score of 0.86. The model for  $\lambda$  has an MSE of 0.037 and an  $R^2$  score of 0.76. Finally, the model for  $\gamma$  achieves an MSE of 0.01 and an  $R^2$  score of 0.78.

These results indicate that the MLP regressors are effective in estimating the GPT parameters based on the provided input features, including the IND-PT parameters and the minimum/maximum values within the group. The low MSE values and relatively high  $R^2$  scores demonstrate the accuracy and goodness of fit of the regression models.

#### Influence Predictor

Our third model is the Influence Predictor, which aims to predict the influence matrix between group members. To develop this predictor, we use the ground-truth influence matrices provided by the participants. The model is designed as a supervised MLP with one hidden layer of size 64 and a softmax activation function to generate the influence distribution over group members for each participant.

The Influence Predictor operates on pairs of individuals, considering their IND parameters and PRE choices. Hence, we have eight input features to predict the influence of person j over person i:  $\alpha_i$ ,  $\lambda_i$ ,  $\gamma_i$ ,  $\alpha_j$ ,  $\lambda_j$ ,  $\gamma_j$ , PRE-choice<sub>i</sub>, and PRE-choice<sub>j</sub>.

The model achieves a KL divergence performance of approximately 0.146, where a smaller value indicates better performance. Similar to the previous models, we assess the importance of each feature by removing it and evaluating the model's performance. Figure 7.9 presents the feature importances for influence predictions. The results indicate that the  $\lambda$  parameters of individuals, which reflect the perceived impact of loss relative



Figure 7.9: Importance of features for influence predictions. *lambda* parameters are the most effective parameter to decide influence.

to gain, are the most effective features in determining influence. On the other hand, the choices made in the pre-discussion phase have a negligible effect on influence predictions.

These findings highlight the significance of individual  $\lambda$  parameters in capturing the influence dynamics within the group. The Influence Predictor effectively utilizes the provided features to estimate the influence matrix, providing insights into the relationships and interactions between group members.

#### **Attacking Group Decisions**

Now, we will use the earlier models for understanding the robustness of the groups. They will help us calculate new GPT parameters and a new influence matrix within a group when a new agent joins. Note that the group members are not aware of this new agent is a human or AI.

To evaluate the robustness of group behavior under an evasion attack, we introduce a new agent whose goal is to maintain the group's consensus behavior while inducing a change in the group's behavior based on new GPT parameters. The attack agent aims to maximize the cosine distance between the group behavior without the agent and the group behavior with the agent, using the consensus GPT parameters.

$$\max \operatorname{CosineDistance}(b(GPT), b(GPT'))$$
(7.7)

where GPT is the GPT parameter without the agent, GPT' is the GPT parameter with the agent, and function b calculates the behavior given PT parameters, as previously described in Preliminaries.



Figure 7.10: Robustness of groups based on cosine distance between group behavior with AI agent and group behavior without AI agent. Missing data shows there are no agents found using grid search to reach a consensus on the same issues. Group 5 is the least robust and groups 10, 15, 16, 21, 22, 23, and 24 are the most robust.

We employ a simple grid search approach to find the attack agent's PT parameters that optimize the objective. The ranking of groups based on the cosine distance between their behavior with and without the attack agent is shown in Figure 7.10. Higher cosine distances indicate a lower level of robustness, as the group behavior changes more significantly due to the evasion attack. For example, Group 5 is the least robust, experiencing the most substantial changes in behavior. On the other hand, some groups exhibit high robustness, where we were unable to find an attack agent that can reach a consensus on the same issues.

Additionally, we investigate the effect of the influence mechanism on the robustness of groups by removing the influence predictor and using the previous influence values within a group for GPT parameter predictions. The results are depicted in Figure 7.11 Although there are no significant overall changes, certain groups, such as Group 1, become less robust when the influence mechanism is not considered. This emphasizes the importance of the influence mechanism in determining the robustness of group behavior.



Figure 7.11: Robustness of groups based on cosine distance between group behavior with AI agent and group behavior without AI agent without considering influence changes in the group. Missing data shows there are no agents found using grid search to reach a consensus on the same issues. Group 5 is the least robust and groups 1, 8, 10, 12, 15, 21, 22, 23, 24, and 25 are the most robust.

Overall, these findings highlight the vulnerability of group decisions to evasion attacks and the significance of considering both the influence mechanism and GPT parameters in assessing the robustness of group behavior.

#### 7.4.6 Human Behavior Generative Model

Instead of relying on grid search and sampled human behaviors with prospect theory (PT) parameters to find an agent that either harms or benefits the group, an intriguing alternative is to leverage generative AI to optimize the objective. A human-behavior generation has emerged as a fascinating area of research, aiming to comprehend human behaviors in social contexts and their interactions with others [298, 299, 300, 301].

By training generative models on our real-world datasets encompassing a broad range of risk preferences and decision-making scenarios, we can guide the model to produce human-like behaviors that align with our desired outcomes. This approach holds promise in exploring the effects of different risk profiles and behaviors on group decision-making dynamics and identifying strategies to optimize decision outcomes while upholding consensus. The integration of generative AI into the study of group decision-making offers a powerful tool to understand the complex interplay between individual behaviors and group dynamics, opening up avenues for innovative approaches to influence and shape group behaviors without compromising consensus.

#### Framework

Our human behavior generation model comprises two main steps: pretraining a Variational Autoencoder (VAE) [302] and fine-tuning it for evasion attacks. It should be noted that the fine-tuning process can be tailored to different tasks.

In the first step, the VAE utilizes Prospect Theory (PT) parameters obtained from randomly sampled humans and the IND-PT parameter of our participants, controlled by a hyperparameter  $c \in [0, 1]$ . During each iteration of the VAE, a sample is drawn from a random set with a probability of c. This allows us to control the level of randomness in our generative model. To begin, we calculate the human behavior based on our behavior calculation, denoted as b(PT), and feed it into the VAE. The VAE then generates three parameters, namely  $\alpha'$ ,  $\lambda'$ , and  $\gamma'$ . The loss function for training the VAE involves reconstructing the input parameters  $\alpha$ ,  $\lambda$ , and  $\gamma$  using a mean squared error (MSE) loss function. Figure 7.12 illustrates VAE diagram of our proposed generative model.



Figure 7.12: VAE Diagram for Generative Model.

Once the VAE is trained, we can utilize its decoder component for fine-tuning, specifically for the evasion attack task. The VAE-decoder is optimized using a similar approach as described in the previous chapter on Attacking Group Decisions. Essentially, we aim to maximize the behavioral distance between a group's decision outcomes with and without the presence of a generated agent as in Equation 7.7.

Our generative model offers several advantages over previous methods:

- ▷ It eliminates the need for grid search across multiple parameter options, streamlining the optimization process.
- ▷ The generative model reduces the need to generate a large number of agents, saving valuable time and computational resources.
- ▷ The flexibility of our generative model allows for easy adaptation to different tasks through fine-tuning.

By leveraging these advantages, our model enhances the efficiency and adaptability of behavior generation, making it a valuable tool for various applications and scenarios.

#### Results

Figure 7.13 presents our preliminary findings comparing the generative model with the grid search method. Both approaches successfully target the majority of groups, but the grid search method exhibits a stronger capability to alter the group consensus behavior. This observation suggests that the generative model may fail to identify certain regions that were overlooked by the grid search. This discrepancy is primarily due to the grid search approach's comprehensive examination of numerous parameters. To address this issue, we reduce the resolution of the grid search by a factor of 2 for each parameter. Figure 7.14 demonstrates that reducing the resolution diminishes the attack performance of the grid search, to the extent that it fails to identify an attacking agent for Group 2 and 16. These findings highlight that the effectiveness of the generative model becomes more apparent as the number of parameters increases, making it challenging to apply grid search at a high resolution.



Figure 7.13: Comparison of two agent selection methods in finding optimal AI agent for attacking the groups. Notably, the grid search method identifies an AI agent with a more impactful attack for groups.



Figure 7.14: Comparison of two agent selection methods for finding optimal AI agents to attack groups, while considering the reduced resolution of the grid search. The results highlight the generative model's ability to maintain continuity and successfully target a larger number of groups (specifically, Groups 2 and 16) for successful attacks.

## 7.5 Conclusions

Our study builds upon previous research that has explored decision-making processes in both individual and group settings. Transitioning from an individual decision-making context to a group setting introduces new dynamics and complexities, especially under risk. Previous studies, such as the work by Askari et al. [303], have examined the role of expertise in group decision-making and highlighted the significance of influence mechanisms within a group. Expanding on this line of inquiry, we extend the analysis to include evasion attacks and evaluate the robustness of group decisions in the presence of malicious agents.

By integrating machine learning models and evasion attacks into the study of group decision-making, we provide a comprehensive framework including grid search and generative model for assessing robustness. Our approach captures individual behaviors, group consensus, and influence dynamics, allowing us to analyze the impact of different factors on group behavior under attack scenarios. This research contributes to the broader understanding of group decision-making processes, highlighting the potential vulnerabilities and the need for resilient decision-making mechanisms. Ultimately, our findings can inform the development of strategies and interventions to enhance the robustness and reliability of group decisions in various domains, ranging from organizational settings to public policy and beyond.

#### 7.5.1 Future Works

#### Human Experiment Validation on Robustness

Firstly, we will incorporate different techniques for our generative model showing its effectiveness over the grid search approach. Secondly, in order to thoroughly evaluate the effectiveness of our attacking schema, it is crucial to validate this idea through real human experiment settings. As previously mentioned, the group members will be unaware of whether other group members are humans or AI agents. This blinding approach will enable us to examine the relationship between humans and AI agents without introducing any biases. By conducting these real human experiments, we can gain valuable insights into the dynamics and interactions between human decision-makers and AI agents within a group context. This approach ensures a more realistic assessment of our attacking schema and enhances the validity of our findings.

#### Text to Prospect Theory

One intriguing avenue for further exploration is the possibility of understanding Prospect Theory parameters from textual data. This entails analyzing the language and content of written text to infer an individual's risk preferences and biases, as reflected in Prospect Theory parameters like loss aversion and probability weighting. By developing advanced natural language processing (NLP) techniques and machine learning models, it may be feasible to extract these parameters from text, thereby empowering AI agents to offer persuasive arguments and exert influence over group decision-making processes. To investigate this prospect, we will design new human experiments that incorporate dialogue during group discussions, allowing us to delve deeper into the relationship between textual data and Prospect Theory parameters.

#### Human-AI Teams on Intellective Issues

We are planning to conduct additional human experiments to analyze the interactions between humans and both other humans and AI agents, specifically focusing on intellective issues. To facilitate this, we will design a jeopardy game format, where human participants answer questions, engage in discussions with other group members, and collectively reach a consensus on the answers. During the game, participants will have the option to seek assistance from an AI agent.

In a separate scenario, we will introduce AI agents as members of the group, while keeping the humans unaware of which members are human and which are AI. This setup will allow us to examine the dynamics of group decision-making when humans interact with both humans and AI agents.

Through these experiments, we aim to gain a deeper understanding of transactive memory, cognitive biases, and the influence system that operates within groups, particularly in the context of intellective issues rather than controversial ones. By exploring these aspects, we can shed light on how human-AI interactions shape decision-making processes and uncover insights into the collective intelligence of mixed human-AI groups.

In the existing literature, various techniques have been proposed to address and understand decision-making processes in human-AI groups. These techniques include active learning [304, 305], reinforcement learning [306, 307], and theory of mind [308]. In our study, we aim to investigate and evaluate these different approaches to enhance the effectiveness and informativeness of our systems. By leveraging insights from these techniques, we can further advance our understanding of decision-making dynamics in human-AI groups and develop improved systems that facilitate collaborative decisionmaking processes.

## Chapter 8

# AI Decision Systems with Feedback Loop Active Learner

## 8.1 Introduction

Accuracy is one of the critical evaluation metrics in decision systems, especially for high-stake applications [309, 123] such as financial event detection [82], drug discovery [310], and autonomous driving [311]. Making the decision systems controlled by AI is risky because of the gray area problem [312], where AI cannot decide the actual answer and uses an artificial and pre-defined threshold. On the other hand, human decision systems are time-consuming and require an expert [313]. It leads us to the following question: Can we improve AI decision systems with the help of human expertise?

Certain high-stakes decisions, such as detecting anomalies in operating server machines wherein missing them would cause financial loss, can be easily given by AI decision systems. In order to make such decisions, AI Decision systems are created using either historical experience (previous anomaly patterns, i.e., learned features) or algorithmic design (certain behaviors are anomalies, i.e., expert-designed features). However, histor-



Figure 8.1: A short illustration of Feedback Loop Active Learner. It starts with multiple entities at which a black box AI system generates decisions. FLAL uses these decisions and entities to send queries to a human, who evaluates them and generates ground truth labels and feedback. FLAL uses this feedback (including human interest and expertise) in active learning training and stores the ground truth labels for future updates in the AI decision system.

ical experiences are not always available because of the label scarcity problem in AI for high-stakes applications. Therefore, the anomaly decision systems generally are designed as unsupervised classification models, which affects generalizability and generates multiple misclassifications. A human decision maker may solve this problem. However, human decision making is very time-consuming and not ideal where a fast decision is necessary. For instance, if a server machine fails, AI could detect this quickly compared to a human however human expertise is needed to check and confirm the detection as well as understand its root cause. In such a context, the human required should be an expert in understanding the problem. This scenario is not limited to high-stakes decisions. Credit card approval systems or insurance acceptance systems are other examples that AI may need the help of human decisions.

Feedback loop (Human-in-the-loop) systems have been studied [314] to create a bridge between AI and humans. They collect labels from the users and improve the AI decision systems. However, can we trust the user's expertise? Even if they are experts, how do we confirm their interest in asked queries? Recommender systems have been proposed to learn the interest of people [315, 316, 317]. The system ranks unseen/unused items and recommends them to the user based on their historical interest or using user interactions [318]. While collecting ground-truth labels, the selection of humans to answer particular queries is critical to improving label correctness and quality. Combining the recommender mechanism with a feedback loop system could potentially increase the performance of AI decision systems by having plenty and correct ground-truth labels.

In this paper, we are looking at specific scenarios of multiple independent entities. Each entity has temporal multidimensional features, and the AI system makes a decision for each entity and time (e.g., anomaly/failure decision). Entities will be ranked by their relevance score to the humans and queried to them to learn their expertise and interests with a pre-defined budget. It helps the framework generate accurate ground-truth labels and labeling operation will not be challenging or boring for a human since they are interested in answering.

We propose FLAL–a novel Feedback Loop Active Learner for better ground-truth labeling–which aims to learn the expertise and interest of a human before querying the entities to them using the active learning mechanism. Figure 8.1 illustrates a summary of FLAL bridging between an AI decision system and a human. FLAL collects decision for entities from the AI system, ranks entities based on their relevance score to human/s, and send queries based on the budget. The human/s answers these queries and sends them back to a FLAL, which learns their behavior towards these entities as well as stores the answers as ground truths. These ground truths will be used to improve AI decision systems in the future. Our main contributions can be summarized as follows:

▷ We highlight the limitation of current AI decision systems, human decisions, and their cooperation to generate data labels. AI decision system makes many mistakes, human

decisions are slow, and cooperation may be limited because of the lack of expertise or interest from humans.

- We propose FLAL, a novel feedback loop active learning framework, for better groundtruth generation and understanding of human behavior. It uses active learning to train the framework based on human feedback and stores generated data labels to improve AI decision systems in the future.
- ▷ We conduct experiments to verify the effectiveness of FLAL. We show that our framework performs better than competing baselines: random forest active learner, AI decision-based, and random recommendations. FLAL not only has the best performance but also converges fast.

## 8.2 Related Works

Human-in-the-Loop: Human-in-the-loop, in other words, feedback-loop, mechanisms are studied in the literature to enhance AI performance by label annotation [319, 320] and generating explanations [321, 322, 323, 324] to black-box operations. Since feedback-loop systems are generally real-time systems, they often use active learning during their training [325, 326, 319]. In our work, we also adapt similar ideas by incorporating human decisions into AI. However, our method increases the efficiency of this cooperation by learning the expertise and interest of humans before asking them questions.

**Recommenders:** Recommendation systems are one of the solutions for understanding the behavior of individuals. They are designed to infer interests and recommend items to humans based on their historical experience or user interactions [318]. The main idea is to rank all relevant items to the user and recommend the top ones. Therefore, ranking algorithms become one of the main components [327, 328, 329] in designing recommendation systems. Recently, as opposed to classical recommenders such as collaborative filtering and matrix factorization, deep-learning frameworks are applied to learn a better representation of items [316, 330, 331]. However, a lack of data availability can restrict the number of parameters to be learned. Even though we still use the advantage of deep learning recommenders, we keep our framework simple but effective. Our recommender system finds better queries based on the user's expertise and interests. In this way, the feedback loop will generate better ground truth labels.

**Temporal Embeddings:** Representation learning on time-series data has become a popular technique to reduce the dimension of the temporal data while keeping the representation (meaning) of it intact [332, 333]. Time2Vec [334] uses a sine activation function to embed the time-series data. Unsupervised time-series embedders have been proposed [335, 336] to deal with label scarcity. Franceschi et al. [335] use the triplet loss function to learn a representation of multidimensional time-series data. The trained embedder can embed any time-series data with any length. More recently, Zerveas et al. [336] propose a transformer-based framework by reconstructing the mask part of the time-series data. FLAL uses a pre-trained temporal data embedder to represent time-series data coming from the entities for better and more compact representations.

## 8.3 Methodology

#### 8.3.1 Problem Formulation

We formulate our problem as a self-supervised time-series classification. Given an entity set  $\mathcal{E} = \{E_1, E_2, \ldots, E_n\}$  where each entity represents a multivariate time-series data (i.e.,  $E_i = [x_{i1}, x_{i2}, \ldots, x_{it}]$ ), an unsupervised AI decision system  $\mathcal{D}$  which generates decision probability  $d_{it}$  for each entity and timestamp (i.e.,  $\mathcal{D}(E_{it}) = d_{it}$ ), a user set  $\mathcal{U} =$  $\{U_1, U_2, \ldots, U_m\}$ , and interest labels  $\mathcal{Y} \in \{0, 1\}^{m \times n \times t}$  for each user to certain timestamp



Figure 8.2: Feedback Loop Active Learner steps. (1) It starts with embedding stream data using pre-trained embedders. (2) User embedding mapper maps the embedding space into a more personalized space. (3) Feature extractor generates learned or expert-designed features to tackle the cold-start problem for recommenders. (4) Generates relevance scores based on AI decision system and extracted features. It sends queries to users for ground truth generation. (5) User generates ground truths and relevancy of the query. They send them back to the framework. Later, FLAL updates its components using an active learning mechanism and keeps ground truth information for future updates on the AI decision system. Notice that the user's interest (relevancy) in queries can also be inferred using interaction detectors.

and entity; our goal is to learn a function  $\hat{F} : \mathcal{E}, \mathcal{D} \to \mathcal{Y}$  that approximates the expertise of users.

#### 8.3.2 FLAL: Feedback Loop Active Learner

We introduce FLAL, a feedback loop active learner that generates ground-truth labels for the AI decision system while learning human expertise and/or interest. FLAL performs user mapping and feature extraction that optimizes the human expert's predictions. As a result, it obtains better ground-truth labels for the AI decision system. Figure 8.2 describes the steps of FLAL in detail. FLAL finds a global embedding space using pre-trained time-series embedders and translates the global embedding space into personalized embedding space. To overcome the cold-start problem, FLAL extracts features from user embedding space and incorporates AI decisions into this feature set. Finally, it calculates relevance scores for each entity and sends the top Q to human experts for evaluation. The human classifies each query which in turn generates ground truth information and adds feedback (explicit or implicit) about their expertise or interest in a certain query. FLAL trains its framework using this feedback via an active learning mechanism and stores the ground truth information to improve AI decision systems in the future when necessary.

#### Stream Data Embedder

To increase the expressiveness and compactness of our data, we use an unsupervised multivariate time-series embedder to represent the entity's time series. This part of our algorithm is pre-trained with the data, which is not used in our experiments. We used [335] for our stream data embedder since it is more flexible to different time-series lengths and generates good representations for anomaly data compared to [336]. During the embedding of the time-series data, we consider the last  $\tau$  timestamps.

$$h_{it} = S([x_{i(t-\tau)}; \dots; x_{it}])$$
 (8.1)

where  $h_{it}$  is an embedding of  $E_{it}$ , S is an unsupervised multivariate time-series embedder,  $x_{it}$  is multivariate time-series data for  $E_{it}$ , and  $[\cdot; \cdot]$  is the row concatenation operator.



Figure 8.3: An illustration of Stream Data Embedder. The pre-trained unsupervised embedder takes input from all entities' time series data (up to  $\tau$  timestamp history) and embeds them into d dimensional space. This allows a better and more compact representation of time-series data.



Figure 8.4: The usefulness example of user embedding mapper. Since the global embedding space will group related entities together and a specific user may be interested in different types of entities, the user embedding mapper will learn how to regroup entities. This example shows that the algorithm may detect the anomalies for different reasons: data issues, seasonal change, and authentication-related. Therefore, if only data issues and authentication-related anomalies (red entities) are relevant to the user, the algorithm groups them together to make the embedding space personalized.

#### User Embedding Mapper

Global embedding space may not be as representative as user embedding space, where one can understand the expertise of users. Therefore, we have a user embedding mapper that maps generated embeddings from the stream data embedder to the user embedding space. This allows them to distinguish between relevant and irrelevant entities to the user. Figure 8.4 shows an example of the usefulness of user embedding mapper. For an anomaly detection problem, multiple reasons can cause an anomaly. But the users are often experts on a certain subset of those anomalies and the user embedding mapper will map these types close to each other. As a result, they can be separated from the other anomaly types or normal ones. The user embeddings are generated as follows:

$$h_{it}^A = g_A(h_{it}) \tag{8.2}$$

where  $h_{it}^A$  is a user embedding for User A, and g is the user embedding mapper. g can be designed as any function, such as the identity or neural networks.

#### Feature Extractor

Since we do not know any information about the users and cannot conduct an initial survey as most recommenders do, we need to extract features from the user embedding space. Features can be learned or designed for a specific application scenario. An example of an expert-designed feature for anomaly applications can be the average distance from one item to others, which will likely be higher for anomaly cases. However, learned features are shown as more expressive than expert-designed features because it is hard to design or engineer all useful features. Feature extractor can also be seen as a function layer on top of the user embedding mapper. So it will learn new features from  $h_{it}^A$ :

$$h_{it}^{\prime A} = f(h_{it}^A) \tag{8.3}$$

where h' has smaller dimension than h, and f is a feature extractor function. f can also represent a set of learned and expert-designed functions. In that case, h' will be a concatenation of extracted features.

#### Recommender

In order to find better queries for specific users, we calculate each entity's relevance score to a user. Note that the AI decision system D already calculates decision probability  $d_{it}$  for entity i. Even though this probability may not be fully correct, we can incorporate it into relevance score calculation to ease the cold-start problem. Furthermore, we will also use the extracted features from the feature extractor. The relevance score of  $E_{it}$  for a user A calculation will be as the following:

$$r_{it}^A = w_1 \times d_{it} + \sum W \odot h_{it}^{\prime A}$$

$$\tag{8.4}$$

where  $w_1$  and W are learned weights. Also, these weights may tell us a story about the importance of AI decision systems and extracted features for different users. Once relevance scores are calculated for all entities, the recommender will send the top Qrelevant entities to the user to get feedback.

#### User Feedback

The users will have a list of queries to be checked and answered. The user will respond to each query with two answers: (1) what should be the decision of the AI system? (2) what is their expertise/interest in this query? The first answer is stored to update the AI decision system if necessary, while the second is used to train FLAL. Note that our main algorithm is not controlling the answering part done by the user. If a query has no response, it means no decision (i.e., do not use to improve the AI decision system) and no expertise (improve FLAL with the information that the user is not an expert). Furthermore, an interaction system can be designed to understand the expertise or interest of the user in queries by looking at their click count or other related metrics. However, this is out of the scope of this project. The expertise information will be stored in  $e_{it}^A \in \{0, 1\}$ , and used to update FLAL.

#### 8.3.3 Training FLAL

We train our framework based on active learning principles since the problem requires learning the behavior of users in real-time to collect better ground truth labels for AI decision systems. The collected ground truth information will be stored to update the AI decision system if necessary. Our active learning mechanism focuses on updating the recommender using feedback from expertise information. For each timestamp t, we train our objective which aims to sort relevance scores of entities,  $R_t^A = [r_{1t}^A, \ldots, r_{nt}^A]$  based on the expertise information array  $E_t^A = [e_{1t}^A, \ldots, e_{Qt}^A]$ . More specifically we use a contrastive loss for our active learning training which contains three different terms as follows:

$$\max L_{\text{ALL}}(E_t^A, R_t^A) = x_1 * \sum_{\substack{i < Q \\ e_{it}^A = 1}} \sum_{\substack{Q > j > i \\ e_{jt}^A = 0}} \sigma(r_{it}^A - r_{jt}^A) \rightarrow L_{\text{WIDEN}}$$

$$+ x_2 * \sum_{\substack{i < Q \\ e_{it}^A = 1}} \sum_{\substack{Q > j > i \\ e_{jt}^A = 0}} \sigma(r_{it}^A - r_{jt}^A) \rightarrow L_{\text{NARROW}}$$

$$+ x_3 * \sum_{\substack{j < Q \\ e_{it}^A = 1}} \sum_{\substack{k > = Q}} \sigma(r_{kt}^A - r_{jt}^A) \rightarrow L_{\text{RECOVER}}$$

where  $L_{\text{WIDEN}}$  widens the gap between correctly ranked positive and negative samples,  $L_{\text{NARROW}}$  narrows the gap between wrongly ranked positive and negative samples, and  $L_{\text{RECOVERY}}$  recovers unrecommended by narrowing the gap between wronglyrecommended samples and unrecommended samples (which may contain useful recommendations). We use  $x_{1,2,3}$  as a tunable hyperparameter to value each term respectively. They can be optimized based on the scenario and the need of an application.

#### Example scenario for our training

Let  $R_t^A = [1.5, 1.2, 1.1, 0.9, 0.3, 0.2], Q = 4, E_t^A = [1, 0, 0, 1, ?, ?], x_1 = 0.50, x_2 = 0.75,$ and  $x_3 = 0.25$ , then our objective will be calculated as follows:

$$\begin{split} L_{\text{ALL}}(E_t^A, R_t^A) &= 0.50 * (\sigma(1.5 - 1.2) + \sigma(1.5 - 1.1)) \rightarrow L_{\text{WIDEN}} \\ &+ 0.75 * (\sigma(0.9 - 1.2) + \sigma(0.9 - 1.1)) \rightarrow L_{\text{NARROW}} \\ &+ 0.25 * (\sigma(0.3 - 1.2) + \sigma(0.2 - 1.2) + \\ &\sigma(0.3 - 1.1) + \sigma(0.2 - 1.1)) \rightarrow L_{\text{RECOVER}} \end{split}$$

#### 8.3.4 User Simulation

To see the effectiveness of our feedback-loop part, we need to simulate the user answers. One way to do this is by using ground truth information for the AI decision system if it is available (simulating that the user's expertise/interest is the ground truth). We apply this strategy in this paper. However, this would allow only one user available in



Figure 8.5: The example scenario of the user simulation. The user space will be randomly assigned in the embedding space. The entities inside of this space will be relevant to the user. So user embedding mapper should learn how to map relevant items to this space based on the answers by the user. This will allow better ground truth generation by the user.

the system. To extend the number of users to more than one, we propose a new way of simulating users. Each user is represented as a Gaussian latent space in the entity embedding space. The user space is assigned randomly. If a query entity is in this space, the user is considered an expert. The user embedding mapper will map related entities into this space. Figure 8.5 shows the mapping example.

## 8.4 Experiments

We illustrate the empirical verification of FLAL compared to three baselines on a dataset. First, we compare method performance with precision at Section 8.4.3. Later on, we conduct two ablation studies: the effect of user embedding mapper (Section 8.4.4) and objective function weights (Section 8.4.5).

#### 8.4.1 Dataset

We use a public Server Machine [337] dataset for our experiments. The dataset contains 38 time-series data with various lengths. For our purpose, we chucked the data into 100 time series (entities) with a length of 365. Each time series consists of 38 different features. At any point in the time series, machine activity is classified as normal or failure. Failure represents an anomaly.

#### 8.4.2 Experimental Settings

#### Baselines

We used three baselines to compare our method.

**Random:** It makes random recommendations in the recommender step to the user.

**AI System Decisions:** It only uses AI decision system probability to recommend entities to the user.

**Random Forest Active Learner:** It combines uncertainty and confidence scores for each entity and recommends them to the user. The model is trained using active learning with a random forest as the estimator and the same settings as FLAL.

#### Other Settings

**Model selection:** We select user embedding mapper as a linear layer as a result of ablation study (See Section 8.4.4), feature extractor generates 15 learned features with linear layer, and recommender is also a linear layer to generate a relevance score for an entity. To simulate user feedback, we use the ground truth information of the dataset.

**Hyperparameters:** We tune hyperparameters of FLAL using grid search. We optimize our model using Adam optimizer with a learning rate of 0.0001, L2 normalization weight on model weights 0.001. We select loss function weights  $x_1$ ,  $x_2$ , and  $x_3$  from {0.0, 0.5, 1.0} (See Section 8.4.5). In our experiments,  $\tau$  is set to 127 (the length of the multivariate time series data becomes 128 with the current snapshot) and the embedding size d is 128. We set Q to 10 and 20 for different runs. The number of recommended items is also set to Q.

**Evaluation metrics:** Since the real evaluation can only consider feedback from the recommended entities, we compare precision metrics in our experiments. At each round, we calculate precision@Q and average precision@Q.

#### 8.4.3 Performance

Figure 8.6 shows precision@Q and average precision@Q, where Q is 10 and 20. We calculate a cumulative average of precision performance at each step. Note that this performance will reflect improved AI decision system performance since we use the user's interest/expertise label as a ground truth decision. Our method, Feedback Loop Active Learner, outperforms the competing baselines at all reported metrics, especially after 10-20 steps. Another essential requirement of learning human interest is convergence speed. FLAL converges in around 50 steps, faster than the best baseline Random Forest Active Learner.

Notice that the precision performance of the AI Decision System is not enough. However, active learner mechanisms improve the performance of the decision system drastically, while random decisions are still worse than the original AI decision system. Another notable difference between Q = 10 and Q = 20 is in precision@Q performance. When Q = 20, the performance drops below 0.6. However, this essentially could happen because the number of anomalies in the data is hardly more than 12 (i.e.,  $20 \times 0.6$ ) at a certain time. This shows us that we should optimize the number of recommended entities based on the number of anomalies at every step instead of using the same values as the budget of Q. The average precision is less vulnerable to this issue since the leading zeros do not affect the result. For both Q, the performance is close to each other. AI Decision System

Random





Figure 8.6: Test scores for Precision@Q and AveragePrecision@Q. FLAL outperforms the competing baselines in both metrics and with different  $Q = \{10, 20\}$ . FLAL's performance becomes strictly better in around 10-20 steps and converges around 50 steps. Another active learning mechanism, Random Forest Active Learner, also generates better performance compared to the original AI decision system performance. Random recommendation mechanism expectedly is the worst.



Figure 8.7: Ablation study of user embedding mapper. The linear and non-linear layers have competing performances at both metrics. The linear mapping converges slowly, and 2 non-linear layers suffer from the lack of available data points.

#### 8.4.4 Different User Embedding Mapper

Figure 8.7 shows an ablation study on different user embedding mapper layers using precision@10 and precision@20. We compare identity, linear, nonlinear, and nonlinear-2 (2 nonlinear layers). The identity layer returns the same embedding space and the non-linear layers use a sigmoid activation. The result suggests that one linear layer captures enough information as much as one nonlinear layer. On the other hand, the identity layer has gradually increasing performance for Precision@10, but with a slower convergence rate. Nonlinear-2 has the worst performance. The reason could be overfitting since the lack of data points.

#### 8.4.5 Loss Function Term Sensitivity

Figure 8.8 shows a sensitivity analysis on objective function terms for the Machine dataset. Each block represents the last step cumulative average of precision@10 scores from FLAL parameterized by different  $x_{1,2,3}$  values. All terms contribute to the perfor-



Figure 8.8: Sensitivity analysis of choosing  $x_{1,2,3}$  for our loss terms on the Machine dataset. The reported blocks show an average of precision@10 across all steps of active learning.  $x_1$  is the most effective term in our loss function as the absence of it generates much worse performance.  $x_2$  and  $x_3$  have similar effects since they aim to narrow between negative and positive samples.

mance, while the absence of the first term (i.e.,  $x_1 = 0$ ) makes the model performance very bad. This behavior is expected since the first term is the core part of the contrastive loss. The second term is not effective as the first term for the Machine dataset, but still increasing the value of  $x_2$  makes the performance better. The third term has a similar effect as the second term. They both aim to narrow the positive and negative sample rankings. The best performance is achieved when hyperparameters are equal to 1, or  $x_1 = 1, x_2 = 0.5$ , and  $x_3 = 0.5$ .

## 8.5 Conclusions

We investigate better and more effective ground truth generation by incorporating recommendation systems into AI decision systems and human collaboration for entitybased time-series data. We propose FLAL understanding the expertise and interest of a human over queries to make feedback more eligible and accurate using active learning. FLAL trains a personalized embedding mapper; uses features extraction and AI sys-
tem decisions to solve the cold-start problem of recommender systems. FLAL performs better than competing baselines: random forest active learners, AI decision-based, and random recommenders; and it converges fast. Furthermore, our ablation studies show that the linear user embedding mapper is learning enough information and each term in the objective function contributes to the result.

Future works: We want to investigate this problem using different datasets and our proposed user simulation setting. We also desire to conduct human experiments to show the effectiveness of FLAL in real settings. Furthermore, we will optimize the number of recommendations instead of using it as the budget Q.

## Chapter 9

## Conclusions

The rapid growth of data generated in our digital world has necessitated the development of effective techniques for extracting key information from the abundance of available data. Representation learning using deep learning has emerged as a powerful approach to uncovering meaningful features from data, enabling informed decision-making in various domains. However, the lack of transparency and explainability in deep learning models poses challenges in ensuring trust, accountability, and ethical use of AI systems.

Addressing the need for explainability and transparency, this dissertation focuses on the transparent representation of learning on graphs. Graphs, or networks, provide a versatile framework for modeling diverse applications and data types, incorporating attributes and temporal dynamics. The dissertation explores the development of transparent graph representation learning methods that not only encode graphs effectively but also provide insights into the decision-making process of the models. By making deep learning models more interpretable, we can enhance trust between humans and AI and mitigate potential ethical concerns.

Furthermore, the dissertation emphasizes the importance of human-AI collaboration and the robustness of group decisions in decision-making tasks. While AI algorithms excel at processing large amounts of data and making quick decisions, human expertise brings nuanced understanding and domain knowledge to enhance the accuracy and effectiveness of AI systems. By leveraging the strengths of both humans and AI, we can develop collaborative decision-making frameworks that mutually benefit from each other's capabilities.

The contributions of this dissertation encompass advancements in transparent graph representation learning, including the development of explainable and fair models, as well as exploring the dynamics of human and AI collaboration. By incorporating transparency into graph representation learning, we aim to provide interpretable models that can be trusted and understood by humans. Additionally, investigating the collaboration between humans and AI contributes to the development of decision-making frameworks that leverage the strengths of both entities. As the field continues to evolve, transparent representation learning and human-AI collaboration will play crucial roles in harnessing the power of data-driven decision-making for the benefit of society.

## Bibliography

- M. Arriola, M. Kosan, Z. Huang, S. Sharma, and A. Singh, Evolving graph autoencoders for multi-scale anomaly detection in attributed networks, On going project (2023).
- [2] M. Kosan, A. Silva, S. Medya, B. Uzzi, and A. Singh, Event detection on dynamic graphs, arXiv preprint arXiv:2110.12148 (2021).
- [3] Z. Huang, M. Kosan, S. Medya, S. Ranu, and A. Singh, Global counterfactual explainer for graph neural networks, in Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM '23, pp. 141–149, 2023.
- [4] M. Kosan, A. Silva, and A. Singh, Robust ante-hoc graph explainer using bilevel optimization, arXiv preprint arXiv:2305.15745 (2023).
- [5] Z. Huang, M. Kosan, A. Silva, and A. Singh, *Link prediction without graph neural networks*, arXiv preprint arXiv:2305.13656 (2023).
- [6] D. Kar, M. Kosan, D. Mandal, S. Medya, A. Silva, P. Dey, and S. Sanyal, Feature-based individual fairness in k-clustering, in Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23, p. 2772–2774, 2023.
- [7] M. Kosan, Z. Huang, F. Bullo, N. Friedkin, and A. Singh, Robustness of human decision making under risk, On going project (2023).
- [8] M. Kosan, L. He, S. Agrawal, H. Liu, and C. Chetia, Ai decision systems with feedback loop active learner, pp. 44–58.
- [9] Z. Li, D. Sun, R. Zhu, and Z. Lin, *Detecting event-related changes in organizational networks using optimized neural network models*, *PloS one* (2017).
- [10] K. Leetaru and P. A. Schrodt, *Gdelt: Global data on events, location, and tone,* 1979–2012, in ISA annual convention, 2013.
- [11] D. M. Romero, B. Uzzi, and J. Kleinberg, Social networks under stress, in WWW, 2016.

- [12] X. Lu and C. Brelsford, Network structure and community evolution on twitter: human behavior change in response to the 2011 japanese earthquake and tsunami, Scientific reports 4 (2014) 6773.
- [13] T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).
- [14] W. Hamilton, Z. Ying, and J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).
- [15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, A comprehensive survey on graph neural networks, arXiv preprint arXiv:1901.00596 (2019).
- [16] S. Georgousis, M. P. Kenning, and X. Xie, Graph deep learning: State of the art and challenges, IEEE Access (2021).
- [17] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, An end-to-end deep learning architecture for graph classification, in AAAI, vol. 32, 2018.
- [18] F. Errica, M. Podda, D. Bacciu, and A. Micheli, A fair comparison of graph neural networks for graph classification, in ICLR, 2020.
- [19] A. Nicolicioiu, I. Duta, and M. Leordeanu, *Recurrent space-time graph neural networks*, vol. 32, pp. 12838–12850, 2019.
- [20] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, Structured sequence modeling with graph convolutional recurrent networks, in NeurIPS, Springer, 2018.
- [21] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, and C. E. Leisersen, Evolvegcn: Evolving graph convolutional networks for dynamic graphs, arXiv preprint arXiv:1902.10191 (2019).
- [22] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, T-gcn: A temporal graph convolutional network for traffic prediction, IEEE ITS (2019).
- [23] S. Deng, H. Rangwala, and Y. Ning, Learning dynamic context graphs for predicting social events, in SIGKDD, 2019.
- [24] T. Sakaki, M. Okazaki, and Y. Matsuo, Earthquake shakes twitter users: real-time event detection by social sensors, in WWW, 2010.
- [25] N. Ramakrishnan, P. Butler, S. Muthiah, N. Self, R. Khandpur, P. Saraf, W. Wang, J. Cadena, A. Vullikanti, et. al., 'beating the news' with embers: forecasting civil unrest using open source indicators, in SIGKDD, 2014.
- [26] F. Atefeh and W. Khreich, A survey of techniques for event detection in twitter, Computational Intelligence 31 (2015), no. 1 132–164.

- [27] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova, Anomaly detection in dynamic networks: a survey, WIREs Computational Statistics (2015).
- [28] S. Rayana and L. Akoglu, Less is more: Building selective anomaly ensembles with application to event detection in temporal graphs, in SDM, 2015.
- [29] L. Akoglu and C. Faloutsos, Event detection in time series of mobile communication graphs, in Army science conference, 2010.
- [30] L. Peel and A. Clauset, Detecting change points in the large-scale structure of evolving networks, in AAAI, 2015.
- [31] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, et. al., Grids-a graph based intrusion detection system for large networks, in NISSC, 1996.
- [32] F. Chen and D. B. Neill, Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs, in SIGKDD, 2014.
- [33] P. Rozenshtein, A. Anagnostopoulos, A. Gionis, and N. Tatti, Event detection in activity networks, in SIGKDD, 2014.
- [34] W. Hu, Y. Yang, Z. Cheng, C. Yang, and X. Ren, Time-series event prediction with evolutionary state graph, in WSDM, 2021.
- [35] C. C. Aggarwal and K. Subbian, Event detection in social streams, in SDM, 2012.
- [36] Y. Ning, R. Tao, C. K. Reddy, H. Rangwala, J. C. Starz, and N. Ramakrishnan, Staple: Spatio-temporal precursor learning for event forecasting, in SDM, 2018.
- [37] N. M. Kriege, F. D. Johansson, and C. Morris, A survey on graph kernels, Applied Network Science 5 (2020), no. 1 1–42.
- [38] N. Kriege and P. Mutzel, Subgraph matching kernels for attributed graphs, in *ICML*, 2012.
- [39] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, Efficient graphlet kernels for large graph comparison, in AISTATS, 2009.
- [40] H. Kashima, K. Tsuda, and A. Inokuchi, Marginalized kernels between labeled graphs, in ICML, 2003.
- [41] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, *Dyrep: Learning representations over dynamic graphs*, in *ICLR*, 2019.

- [42] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, Continuous-time dynamic network embeddings, in Companion Proceedings of the The Web Conference 2018, pp. 969–976, 2018.
- [43] U. Singer, I. Guy, and K. Radinsky, Node embedding over temporal graphs, arXiv preprint arXiv:1903.08889 (2019).
- [44] Y. Lu, X. Wang, C. Shi, P. S. Yu, and Y. Ye, Temporal network embedding with micro-and macro-dynamics, in CIKM, pp. 469–478, 2019.
- [45] M. Gori, G. Monfardini, and F. Scarselli, A new model for learning in graph domains, in IJNN, IEEE, 2005.
- [46] D. K. Duvenaud, D. Maclaurein, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, *Convolutional networks on graphs for learning molecular fingerprints*, in *NeurIPS*, pp. 2224–2232, 2015.
- [47] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).
- [48] J. Skarding, B. Gabrys, and K. Musial, Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey, arXiv preprint arXiv:2005.07496 (2020).
- [49] S. Yan, Y. Xiong, and D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in AAAI, 2018.
- [50] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in AAAI, 2019.
- [51] A. Taheri, K. Gimpel, and T. Berger-Wolf, Learning to represent the evolution of dynamic graphs with recurrent models, in WebConf, 2019.
- [52] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, Dysat: Deep neural representation learning on dynamic graphs via self-attention networks, in WSDM, 2020.
- [53] M. Niepert, M. Ahmed, and K. Kutzkov, *Learning convolutional neural networks* for graphs, in *ICML*, 2016.
- [54] J. B. Lee, R. Rossi, and X. Kong, *Graph classification using structural attention*, in *SIGKDD*, 2018.
- [55] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, *Hierarchical graph representation learning with differentiable pooling*, arXiv preprint arXiv:1806.08804 (2018).

- [56] J. Li, Y. Rong, H. Cheng, H. Meng, W. Huang, and J. Huang, Semi-supervised graph classification: A hierarchical graph perspective, in The Web Conference, pp. 972–982, 2019.
- [57] T. Fawcett, An introduction to roc analysis, Pattern recognition letters 27 (2006), no. 8 861–874.
- [58] Y. Ho and S. Wookey, The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling, IEEE Access 8 (2020) 4806–4813.
- [59] S. Wang, Z. Chen, B. Liu, and S. Emery, *Identifying search keywords for finding* relevant social media posts, in AAAI, 2016.
- [60] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word* representations in vector space, arXiv preprint arXiv:1301.3781 (2013).
- [61] C. Bergmeir and J. M. Benítez, On the use of cross-validation for time series predictor evaluation, Information Sciences 191 (2012) 192–213.
- [62] S. Jain and B. C. Wallace, Attention is not explanation, arXiv preprint arXiv:1902.10186 (2019).
- [63] D. Pruthi, M. Gupta, B. Dhingra, G. Neubig, and Z. C. Lipton, Learning to deceive with attention-based explanations, arXiv preprint arXiv:1909.07913 (2019).
- [64] S. Wiegreffe and Y. Pinter, Attention is not not explanation, arXiv preprint arXiv:1908.04626 (2019).
- [65] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, Advances in neural information processing systems 32 (2019) 9240.
- [66] L. Van der Maaten and G. Hinton, Visualizing data using t-sne., JMLR 9 (2008), no. 11.
- [67] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, TISSEC (2003).
- [68] T. Jéron, H. Marchand, S. Pinchinat, and M.-O. Cordier, Supervision patterns in discrete event systems diagnosis, in WODES, 2006.
- [69] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. 2016.
- [70] Y.-L. Boureau, J. Ponce, and Y. LeCun, A theoretical analysis of feature pooling in visual recognition, in ICML, pp. 111–118, 2010.
- [71] J. Tang, J. Li, Z.-C. Gao, and J. Li, *Rethinking graph neural networks for anomaly detection*, .

- [72] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, Automating the construction of internet portals with machine learning, Information Retrieval 3 (2000) 127–163.
- [73] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, Graph attention networks, in ICLR, 2018.
- [74] Y. Wang, B. Feng, G. Li, S. Li, L. Deng, Y. Xie, and Y. Ding, *Ganadvisor: An efficient runtime system for gan acceleration on gpus*, in OSDI, 2021.
- [75] Y. Wang, B. Feng, and Y. Ding, *Qgtc: accelerating quantized graph neural networks via gpu tensor core*, in *PPoPP*, 2022.
- [76] S. Nishad, S. Agarwal, A. Bhattacharya, and S. Ranu, Graphreach: Position-aware graph neural network using reachability estimations, in IJCAI, 2021.
- [77] M. Jiang, Z. Li, S. Zhang, S. Wang, X. Wang, Q. Yuan, and Z. Wei, Drug-target affinity prediction using graph neural network and contact maps, RSC advances 10 (2020), no. 35 20701–20712.
- [78] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. J. Jiang, E. M. Songhori, S. Wang, Y. Lee, E. Johnson, O. Pathak, S. Bae, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, A. Babu, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, *Chip placement with deep reinforcement learning*, *CoRR* abs/2004.10746 (2020).
- [79] S. Manchanda, A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs, in NeurIPS, 2020.
- [80] R. Bhattoo, S. Ranu, and N. Krishnan, Learning articulated rigid body dynamics with lagrangian graph neural network, in NeurIPS, 2022.
- [81] A. Thangamuthu, G. Kumar, S. Bishnoi, R. Bhattoo, N. M. A. Krishnan, and S. Ranu, Unravelling the performance of physics-informed graph neural networks for dynamical systems, in NeurIPS, 2022.
- [82] M. Kosan, A. Silva, S. Medya, B. Uzzi, and A. Singh, Event detection on dynamic graphs, arXiv preprint arXiv:2110.12148 (2021).
- [83] S. Medya, M. Rasoolinejad, Y. Yang, and B. Uzzi, An exploratory study of stock price movements from earnings calls, in WebConf, 2022.
- [84] S. Gupta, S. Manchanda, S. Bedathur, and S. Ranu, TIGGER: scalable generative modelling for temporal interaction graphs, in AAAI, 2022.

- [85] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, *Protein function prediction via graph kernels*, *Bioinformatics* 21 (2005), no. suppl\_1 i47-i56.
- [86] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, Like like alike: joint friendship and interest propagation in social networks, in WebConf, 2011.
- [87] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, Parameterized explainer for graph neural network, in NeurIPS, 2020.
- [88] M. Vu and M. T. Thai, *Pgm-explainer: Probabilistic graphical model explanations* for graph neural networks, in NeurIPS, 2020.
- [89] H. Yuan, J. Tang, X. Hu, and S. Ji, Xgnn: Towards model-level explanations of graph neural networks, in SIGKDD, 2020.
- [90] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, *Cf-gnnexplainer: Counterfactual explanations for graph neural networks*, in *AISTATS*, 2022.
- [91] M. Bajaj, L. Chu, Z. Y. Xue, J. Pei, L. Wang, P. C.-H. Lam, and Y. Zhang, *Robust counterfactual explanations on graph neural networks*, in *NeurIPS*, vol. 34, pp. 5644–5655, 2021.
- [92] C. Abrate and F. Bonchi, Counterfactual graphs for explainable classification of brain networks, in SIGKDD, 2021.
- [93] J. Tan, S. Geng, Z. Fu, Y. Ge, S. Xu, Y. Li, and Y. Zhang, Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning, in WebConf, pp. 1018–1027, 2022.
- [94] P. Voigt and A. Von dem Bussche, The eu general data protection regulation (gdpr), A Practical Guide, 1st Ed., Cham: Springer International Publishing 10 (2017), no. 3152676 10-5555.
- [95] J. Xiong, Z. Xiong, K. Chen, H. Jiang, and M. Zheng, Graph neural networks for automated de novo drug design, Drug Discovery Today 26 (2021), no. 6 1382–1393.
- [96] J. Kazius, R. McGuire, and R. Bursi, Derivation and validation of toxicophores for mutagenicity prediction, Journal of medicinal chemistry 48 (2005), no. 1 312–320.
- [97] A. Sanfeliu and K.-S. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE transactions on systems, man, and cybernetics (1983), no. 3 353–362.

- [98] K. Borgwardt, N. Schraudolph, and S. Vishwanathan, *Fast computation of graph kernels*, in *NeurIPS*, 2006.
- [99] F. Costa and K. De Grave, *Fast neighborhood subgraph pairwise distance kernel*, in *ICML*, 2010.
- [100] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, Weisfeiler-lehman graph kernels., Journal of Machine Learning Research 12 (2011), no. 9.
- [101] K. Rawal and H. Lakkaraju, Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses, in NeurIPS, 2020.
- [102] Y. Liang and P. Zhao, Similarity search in graph databases: A multi-layered indexing approach, in ICDE, pp. 783–794, 2017.
- [103] R. Ranjan, S. Grover, S. Medya, V. Chakravarthy, Y. Sabharwal, and S. Ranu, Greed: A neural framework for learning graph distance functions, in NeurIPS, 2022.
- [104] R. Pemantle, Vertex-reinforced random walk, Probability Theory and Related Fields 92 (1992), no. 1 117–136.
- [105] B. Perozzi, R. Al-Rfou, and S. Skiena, *Deepwalk: Online learning of social representations*, in *SIGKDD*, 2014.
- [106] J. Klicpera, A. Bojchevski, and S. Günnemann, *Predict then propagate: Graph* neural networks meet personalized pagerank, in *ICLR*, 2018.
- [107] Z. Huang, A. Silva, and A. Singh, A broader picture of random-walk based graph embedding, in SIGKDD, 2021.
- [108] Z. Huang, A. Silva, and A. Singh, Pole: Polarized embedding for signed networks, in WSDM, 2022.
- [109] Q. Mei, J. Guo, and D. Radev, *Divrank: the interplay of prestige and diversity in information networks*, in *SIGKDD*, 2010.
- [110] D. Natarajan and S. Ranu, A scalable and generic framework to mine top-k representative subgraph patterns, in ICDM, 2016.
- [111] A. Metwally, D. Agrawal, and A. E. Abbadi, *Efficient computation of frequent* and top-k elements in data streams, in *ICDT*, 2005.
- [112] N. Wale and G. Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, in ICDM, 2006.

- [113] K. Riesen and H. Bunke, Iam graph database repository for graph based pattern recognition and machine learning, in Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pp. 287–297, Springer, 2008.
- [114] P. D. Dobson and A. J. Doig, Distinguishing enzyme structures from non-enzymes without alignments, Journal of molecular biology 330 (2003), no. 4 771–783.
- [115] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).
- [116] E. Sarubbi, P. F. Seneci, M. R. Angelastro, N. P. Peet, M. Denaro, and K. Islam, *Peptide aldehydes as inhibitors of hiv protease*, *FEBS letters* **319** (1993), no. 3 253–256.
- [117] T. Zhao, G. Liu, D. Wang, W. Yu, and M. Jiang, Learning from counterfactual links for link prediction, in ICML, 2022.
- [118] Y. Xie, C. Shi, H. Zhou, Y. Yang, W. Zhang, Y. Yu, and L. Li, Mars: Markov molecular sampling for multi-objective drug discovery, arXiv preprint arXiv:2103.10432 (2021).
- [119] X. Kong, W. Huang, Z. Tan, and Y. Liu, Molecule generation by principal subgraph mining and assembling, Advances in Neural Information Processing Systems 35 (2022) 2550–2563.
- [120] L. Faber, A. K. Moghaddam, and R. Wattenhofer, Contrastive graph neural network explanation, in ICML Workshop on Graph Representation Learning and Beyond, p. 28, International Conference on Machine Learning, 2020.
- [121] G. Vilone and L. Longo, Explainable artificial intelligence: a systematic review, arXiv preprint arXiv:2006.00093 (2020).
- [122] H. Yuan, H. Yu, S. Gui, and S. Ji, Explainability in graph neural networks: A taxonomic survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).
- [123] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (2019), no. 5 206–215.
- [124] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee, Protgnn: Towards self-explaining graph neural networks, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 9127–9135, 2022.

- [125] 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, pp. 1263–1272, PMLR, 2017.
- [126] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, How powerful are graph neural networks?, in ICLR, 2018.
- [127] H. Gao and S. Ji, Graph u-nets, in international conference on machine learning, pp. 2083–2092, PMLR, 2019.
- [128] D. Mesquita, A. Souza, and S. Kaski, Rethinking pooling in graph neural networks, Advances in Neural Information Processing Systems 33 (2020) 2220–2231.
- [129] J. Baek, M. Kang, and S. J. Hwang, Accurate learning of graph representations with graph multiset pooling, in ICLR, 2021.
- [130] C. Molnar, Interpretable machine learning. Lulu. com, 2020.
- [131] 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, pp. 12241–12252, PMLR, 2021.
- [132] Y. Xie, S. Katariya, X. Tang, E. Huang, N. Rao, K. Subbian, and S. Ji, Task-agnostic graph explanations, arXiv preprint arXiv:2202.08335 (2022).
- [133] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, Graph information bottleneck for subgraph recognition, in International Conference on Learning Representations, 2021.
- [134] B. Colson, P. Marcotte, and G. Savard, An overview of bilevel optimization, Annals of operations research 153 (2007), no. 1 235–256.
- [135] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, Bilevel programming for hyperparameter optimization and meta-learning, in International Conference on Machine Learning, pp. 1568–1577, PMLR, 2018.
- [136] K. Huang and M. Zitnik, Graph meta learning via local subgraphs, Advances in Neural Information Processing Systems 33 (2020).
- [137] G. Wan and H. Schweitzer, Edge sparsification for graphs via meta-learning, in 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 2733–2738, IEEE, 2021.
- [138] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, and L. Wang, Deep graph structure learning for robust representations: A survey, arXiv preprint arXiv:2103.03036 (2021).

- [139] L. Franceschi, M. Niepert, M. Pontil, and X. He, Learning discrete structures for graph neural networks, in International conference on machine learning, pp. 1972–1982, PMLR, 2019.
- [140] Q. Sun, J. Li, H. Peng, J. Wu, X. Fu, C. Ji, and S. Y. Philip, Graph structure learning with variational information bottleneck, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 4165–4174, 2022.
- [141] H. Zhou, A. Vani, H. Larochelle, and A. Courville, *Fortuitous forgetting in connectionist networks, arXiv preprint arXiv:2202.00155* (2022).
- [142] I. Alabdulmohsin, H. Maennel, and D. Keysers, The impact of reinitialization on generalization in convolutional neural networks, arXiv preprint arXiv:2109.00267 (2021).
- [143] E. Grefenstette, B. Amos, D. Yarats, P. M. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala, *Generalized inner loop meta-learning*, arXiv preprint arXiv:1910.01727 (2019).
- [144] P. Yanardag and S. Vishwanathan, Deep graph kernels, in SIGKDD, pp. 1365–1374, 2015.
- [145] T. M. Mitchell and T. M. Mitchell, *Machine learning*, vol. 1. McGraw-hill New York, 1997.
- [146] M. Zitnik, R. Sosič, M. W. Feldman, and J. Leskovec, Evolution of resilience in protein interactomes across the tree of life, Proceedings of the National Academy of Sciences 116 (2019), no. 10 4426–4433.
- [147] K. K. Yang, Z. Wu, C. N. Bedbrook, and F. H. Arnold, Learned protein embeddings for machine learning, Bioinformatics 34 (2018), no. 15 2642–2648.
- [148] P. A. Papp, K. Martinkus, L. Faber, and R. Wattenhofer, Dropgnn: random dropouts increase the expressiveness of graph neural networks, Advances in Neural Information Processing Systems 34 (2021).
- [149] C. Agarwal, O. Queen, H. Lakkaraju, and M. Zitnik, Evaluating explainability for graph neural networks, Scientific Data 10 (2023), no. 1 144.
- [150] A. Antoniou, H. Edwards, and A. Storkey, *How to train your maml, arXiv* preprint arXiv:1810.09502 (2018).
- [151] J. Yang, K. Ji, and Y. Liang, Provably faster algorithms for bilevel optimization, Advances in Neural Information Processing Systems 34 (2021) 13670–13682.
- [152] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, Arnetminer: extraction and mining of academic social networks, in SIGKDD, 2008.

- [153] C. Li, J. Ma, X. Guo, and Q. Mei, Deepcas: An end-to-end predictor of information cascades, in WebConf, 2017.
- [154] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, Deepinf: Social influence prediction with deep learning, in SIGKDD, 2018.
- [155] M. Jamali and M. Ester, *Trustwalker: a random walk model for combining trust-based and item-based recommendation*, in *SIGKDD*, 2009.
- [156] F. Monti, M. Bronstein, and X. Bresson, *Geometric matrix completion with* recurrent multi-graph neural networks, in NeurIPS, 2017.
- [157] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, *Kgat: Knowledge graph* attention network for recommendation, in SIGKDD, 2019.
- [158] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. Cohen, Open domain question answering using early fusion of knowledge bases and text, in EMNLP, 2018.
- [159] S. K. Sahu, F. Christopoulou, M. Miwa, and S. Ananiadou, *Inter-sentence relation* extraction with document-level graph convolutional neural network, in ACL, 2019.
- [160] L. Yao, C. Mao, and Y. Luo, Graph convolutional networks for text classification, in AAAI, 2019.
- [161] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, *Graph networks as learnable physics engines for inference and control*, in *ICML*, 2018.
- [162] B. Ivanovic and M. Pavone, The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs, in ICCV, 2019.
- [163] A. L. da Silva, F. Kocayusufoglu, S. Jafarpour, F. Bullo, A. Swami, and A. Singh, Combining physics and machine learning for network flow estimation, in ICLR, 2020.
- [164] L. Lü and T. Zhou, *Link prediction in complex networks: A survey, Physica A:* statistical mechanics and its applications **390** (2011), no. 6 1150–1170.
- [165] V. Martínez, F. Berzal, and J.-C. Cubero, A survey of link prediction in complex networks, ACM computing surveys (CSUR) 49 (2016), no. 4 1–33.
- [166] Y. Qi, Z. Bar-Joseph, and J. Klein-Seetharaman, Evaluation of different biological data and computational classification methods for use in protein interaction prediction, Proteins: Structure, Function, and Bioinformatics 63 (2006), no. 3 490–500.

- [167] D. Liben-Nowell and J. Kleinberg, The link-prediction problem for social networks, Journal of the American society for information science and technology 58 (2007), no. 7 1019–1031.
- [168] Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009), no. 8 30–37.
- [169] T. Martin, B. Ball, and M. E. Newman, Structural inference for uncertain networks, Physical Review E 93 (2016), no. 1 012306.
- [170] A. Bahulkar, B. K. Szymanski, N. O. Baycik, and T. C. Sharkey, *Community* detection with edge augmentation in criminal networks, in ASONAM, 2018.
- [171] B. Wilder, E. Ewing, B. Dilkina, and M. Tambe, *End to end learning and optimization on graphs*, *NeurIPS* (2019).
- [172] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, Simplifying graph convolutional networks, in ICML, 2019.
- [173] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, Robust graph representation learning via neural sparsification, in International Conference on Machine Learning, pp. 11458–11468, PMLR, 2020.
- [174] 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, 2019.
- [175] M. Zhang and Y. Chen, Link prediction based on graph neural networks, in NeurIPS, 2018.
- [176] S. Yun, S. Kim, J. Lee, J. Kang, and H. J. Kim, *Neo-gnns: Neighborhood* overlap-aware graph neural networks for link prediction, in *NeurIPS*, 2021.
- [177] L. Pan, C. Shi, and I. Dokmanić, Neural link prediction with walk pooling, in ICLR, 2022.
- [178] M. E. Newman, Clustering and preferential attachment in growing networks, Physical review E 64 (2001), no. 2 025102.
- [179] L. A. Adamic and E. Adar, Friends and neighbors on the web, Social networks 25 (2003), no. 3 211–230.
- [180] A.-L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, Evolution of the social network of scientific collaborations, Physica A: Statistical mechanics and its applications **311** (2002), no. 3-4 590–614.

- [181] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson, Combining label propagation and simple models out-performs graph neural networks, in ICLR, 2021.
- [182] Q. Li, Z. Han, and X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in AAAI, 2018.
- [183] J. Ma, B. Chang, X. Zhang, and Q. Mei, Copulagnn: towards integrating representational and correlational roles of graphs in graph neural networks, in ICLR, 2020.
- [184] T. N. Kipf and M. Welling, Variational graph auto-encoders, arXiv preprint arXiv:1611.07308 (2016).
- [185] I. Chami, Z. Ying, C. Ré, and J. Leskovec, Hyperbolic graph convolutional neural networks, in NeurIPS, 2019.
- [186] Y. Zhang, X. Wang, C. Shi, N. Liu, and G. Song, Lorentzian graph convolutional networks, in WebConf, 2021.
- [187] L. Cai, J. Li, J. Wang, and S. Ji, Line graph neural networks for link prediction, IEEE TPAMI (2021).
- [188] Z. Yan, T. Ma, L. Gao, Z. Tang, and C. Chen, Link prediction with persistent homology: An interactive view, in ICML, 2021.
- [189] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang, Neural bellman-ford networks: A general graph neural network framework for link prediction, in NeurIPS, 2021.
- [190] Y. Chen, Y. R. Gel, and H. V. Poor, Bscnets: Block simplicial complex neural networks, in AAAI, 2022.
- [191] J. Davis and M. Goadrich, *The relationship between precision-recall and roc* curves, in *ICML*, 2006.
- [192] T. Saito and M. Rehmsmeier, The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets, PloS one 10 (2015), no. 3 e0118432.
- [193] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, in NeurIPS, 2020.
- [194] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, Translating embeddings for modeling multi-relational data, in NeurIPS, 2013.

- [195] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, Asymmetric transitivity preserving graph embedding, in SIGKDD, pp. 1105–1114, 2016.
- [196] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, Arbitrary-order proximity preserved network embedding, in SIGKDD, 2018.
- [197] Z. Huang, A. Silva, and A. Singh, A broader picture of random-walk based graph embedding, in SIGKDD, 2021.
- [198] B. Yang, S. W.-t. Yih, X. He, J. Gao, and L. Deng, *Embedding entities and relations for learning and inference in knowledge bases*, in *ICLR*, 2015.
- [199] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, *Rotate: Knowledge graph embedding* by relational rotation in complex space, in *ICLR*, 2018.
- [200] H. Schütze, C. D. Manning, and P. Raghavan, Introduction to information retrieval, vol. 39. Cambridge University Press Cambridge, 2008.
- [201] X. Chen, X. Cheng, and S. Mallat, Unsupervised deep haar scattering on graphs, in NeurIPS, 2014.
- [202] K. Sohn, Improved deep metric learning with multi-class n-pair loss objective, in NeurIPS, 2016.
- [203] B. McFee and G. Lanckriet, *Metric learning to rank*, in *ICML*, 2010.
- [204] F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, Deep metric learning to rank, in CVPR, 2019.
- [205] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, Learning with average precision: Training image retrieval with a listwise loss, in ICCV, 2019.
- [206] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, and N. M. Robertson, Ranked list loss for deep metric learning, in ICCV, 2019.
- [207] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, An efficient boosting algorithm for combining preferences, JMLR 4 (2003), no. Nov 933–969.
- [208] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, *Listwise approach to learning to rank: theory and algorithm*, in *ICML*, 2008.
- [209] S. Bruch, An alternative cross entropy loss for learning-to-rank, in WebConf, 2021.
- [210] L. Cai and W. Y. Wang, *Kbgan: Adversarial learning for knowledge graph* embeddings, in NAACL, 2018.
- [211] P. Wang, S. Li, and R. Pan, Incorporating gan for negative sampling in knowledge representation learning, in AAAI, 2018.

- [212] V. Satuluri and S. Parthasarathy, *Bayesian locality sensitive hashing for fast similarity search*, in *VLDB*, 2012.
- [213] D. C. Anastasiu and G. Karypis, *L2ap: Fast cosine similarity search with prefix l-2 norm bounds*, in *ICDE*, 2014.
- [214] Z. Liu, D. Lai, C. Li, and M. Wang, Feature fusion based subgraph classification for link prediction, in CIKM, 2020.
- [215] C. L. Giles, K. D. Bollacker, and S. Lawrence, Citeseer: An automatic citation indexing system, in ACM conference on Digital libraries, 1998.
- [216] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (2008), no. 3 93–93.
- [217] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, *Image-based* recommendations on styles and substitutes, in SIGIR, 2015.
- [218] M. Fey and J. E. Lenssen, Fast graph representation learning with PyTorch Geometric, in ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.
- [219] Z. Yang, W. Cohen, and R. Salakhudinov, *Revisiting semi-supervised learning* with graph embeddings, in *ICML*, 2016.
- [220] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, *Pitfalls of graph* neural network evaluation, arXiv preprint arXiv:1811.05868 (2018).
- [221] T. Zhou, L. Lü, and Y.-C. Zhang, Predicting missing links via local information, The European Physical Journal B 71 (2009), no. 4 623–630.
- [222] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et. al., Pytorch: An imperative style, high-performance deep learning library, in NeurIPS, 2019.
- [223] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, *Learning to rank using gradient descent*, in *ICML*, 2005.
- [224] L. Katz, A new status index derived from sociometric analysis, Psychometrika 18 (1953), no. 1 39–43.
- [225] L. Page, S. Brin, R. Motwani, and T. Winograd, *The pagerank citation ranking:* Bringing order to the web., tech. rep., Stanford InfoLab, 1999.
- [226] G. Jeh and J. Widom, Simrank: a measure of structural-context similarity, in SIGKDD, 2002.

- [227] A. Grover and J. Leskovec, node2vec: Scalable feature learning for networks, in SIGKDD, pp. 855–864, 2016.
- [228] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec, in WSDM, 2018.
- [229] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, *Stability of graph communities across time scales*, *PNAS* **107** (2010), no. 29 12755–12760.
- [230] Z. Huang, A. Silva, and A. Singh, Pole: Polarized embedding for signed networks, in WSDM, 2022.
- [231] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, Graphran: Generating realistic graphs with deep auto-regressive models, in International conference on machine learning, pp. 5708–5717, PMLR, 2018.
- [232] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, *Learning deep generative models of graphs*, in *ICML*, 2018.
- [233] A. Grover, A. Zweig, and S. Ermon, Graphite: Iterative generative modeling of graphs, in ICML, 2019.
- [234] T. Zhao, G. Liu, S. Günnemann, and M. Jiang, Graph data augmentation for graph machine learning: A survey, arXiv preprint arXiv:2202.08871 (2022).
- [235] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, *Data augmentation for graph neural networks*, in *AAAI*, 2021.
- [236] Y. Chen, L. Wu, and M. Zaki, Iterative deep graph learning for graph neural networks: Better and robust node embeddings, Advances in Neural Information Processing Systems 33 (2020) 19314–19326.
- [237] Y. Yang, L. Wu, R. Hong, K. Zhang, and M. Wang, *Enhanced graph learning for* collaborative filtering via mutual information maximization, in SIGIR, 2021.
- [238] A. Singh, Q. Huang, S. L. Huang, O. Bhalerao, H. He, S.-N. Lim, and A. R. Benson, Edge proposal sets for link prediction, arXiv preprint arXiv:2106.15810 (2021).
- [239] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, *Machine bias*, *ProPublica*, *May* 23 (2016), no. 2016 139–159.
- [240] A. Chouldechova, Fair prediction with disparate impact: A study of bias in recidivism prediction instruments, Big data 5 (2017), no. 2 153–163.
- [241] A. Fuster, P. Goldsmith-Pinkham, T. Ramadorai, and A. Walther, Predictably unequal? the effects of machine learning on credit markets, The Effects of Machine Learning on Credit Markets (October 1, 2020) (2020).

- [242] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, Dissecting racial bias in an algorithm used to manage the health of populations, Science 366 (2019), no. 6464 447–453.
- [243] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, Certifying and removing disparate impact, in proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 259–268, 2015.
- [244] A. Agarwal, A. Beygelzimer, M. Dudik, J. Langford, and H. Wallach, A reductions approach to fair classification, in International Conference on Machine Learning, pp. 60–69, PMLR, 2018.
- [245] M. Abbasi, S. A. Friedler, C. Scheidegger, and S. Venkatasubramanian, Fairness in representation: quantifying stereotyping as a representational harm, in Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 801–809, 2019.
- [246] C. Jung, S. Kannan, and N. Lutz, A center in your neighborhood: Fairness in facility location, arXiv preprint arXiv:1908.09041 (2019).
- [247] F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii, Fair clustering through fairlets, in Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 5036–5044, 2017.
- [248] A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner, Scalable fair clustering, in International Conference on Machine Learning, pp. 405–413, PMLR, 2019.
- [249] S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani, Fair algorithms for clustering, in Advances in Neural Information Processing Systems, pp. 4955–4966, 2019.
- [250] S. Ahmadian, A. Epasto, R. Kumar, and M. Mahdian, Clustering without over-representation, in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 267–275, 2019.
- [251] S. Mahabadi and A. Vakilian, Individual fairness for k-clustering, in International Conference on Machine Learning, pp. 6586–6596, PMLR, 2020.
- [252] D. Chakrabarty and M. Negahbani, Better algorithms for individually fair k-clustering, NeurIPS (2021).
- [253] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, Fairness through awareness, in Proceedings of the 3rd innovations in theoretical computer science conference, pp. 214–226, 2012.

- [254] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen, How much can behavioral targeting help online advertising?, in Proceedings of the 18th international conference on World wide web, pp. 261–270, 2009.
- [255] I. O. Bercea, M. Groß, S. Khuller, A. Kumar, C. Rösner, D. R. Schmidt, and M. Schmidt, On the cost of essentially fair clusterings., in APPROX-RANDOM, 2019.
- [256] C. Rösner and M. Schmidt, Privacy preserving clustering with constraints, in 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- [257] M. Schmidt, C. Schwiegelshohn, and C. Sohler, Fair coresets and streaming algorithms for fair k-means clustering, arXiv preprint arXiv:1812.10854 (2018).
- [258] L. Huang, S. Jiang, and N. Vishnoi, Coresets for clustering with fairness constraints, Advances in Neural Information Processing Systems 32 (2019) 7589–7600.
- [259] A. Vakilian and M. Yalçıner, Improved approximation algorithms for individually fair clustering, arXiv preprint arXiv:2106.14043 (2021).
- [260] R. Chhaya, A. Dasgupta, J. Choudhari, and S. Shit, On coresets for fair regression and individually fair clustering., in AISTATS, pp. 9603–9625, 2022.
- [261] D. Chakrabarti, J. P. Dickerson, S. A. Esmaeili, A. Srinivasan, and L. Tsepenekas, A new notion of individually fair clustering: α-equitable k-center, in International Conference on Artificial Intelligence and Statistics, pp. 6387–6408, PMLR, 2022.
- [262] B. Brubach, D. Chakrabarti, J. P. Dickerson, A. Srinivasan, and L. Tsepenekas, Fairness, semi-supervised learning, and more: A general framework for clustering with stochastic pairwise constraints, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 6822–6830, 2021.
- [263] M. Kleindessner, P. Awasthi, and J. Morgenstern, A notion of individual fairness for clustering, arXiv preprint arXiv:2006.04960 (2020).
- [264] M. Ghadiri, S. Samadi, and S. Vempala, Socially fair k-means clustering, in Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 438–448, 2021.
- [265] M. Kleindessner, P. Awasthi, and J. Morgenstern, Fair k-center clustering for data summarization, in International Conference on Machine Learning, pp. 3448–3457, 2019.

- [266] S. Samadi, U. Tantipongpipat, J. Morgenstern, M. Singh, and S. Vempala, The price of fair pca: one extra dimension, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 10999–11010, 2018.
- [267] M. Kleindessner, S. Samadi, P. Awasthi, and J. Morgenstern, Guarantees for spectral clustering with fairness constraints, in International Conference on Machine Learning, pp. 3458–3467, 2019.
- [268] M. Donini, L. Oneto, S. Ben-David, J. Shawe-Taylor, and M. Pontil, Empirical risk minimization under fairness constraints, in Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 2796–2806, 2018.
- [269] L. E. Celis, L. Huang, V. Keswani, and N. K. Vishnoi, Classification with fairness constraints: A meta-algorithm with provable guarantees, in Proceedings of the conference on fairness, accountability, and transparency, pp. 319–328, 2019.
- [270] A. Chouldechova and A. Roth, The frontiers of fairness in machine learning, arXiv preprint arXiv:1810.08810 (2018).
- [271] A. D. Selbst, D. Boyd, S. A. Friedler, S. Venkatasubramanian, and J. Vertesi, Fairness and abstraction in sociotechnical systems, in Proceedings of the conference on fairness, accountability, and transparency, pp. 59–68, 2019.
- [272] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, A survey on bias and fairness in machine learning, ACM Computing Surveys (CSUR) 54 (2021), no. 6 1–35.
- [273] C. Bazgan, Z. Tuza, and D. Vanderpooten, The satisfactory partition problem, Discrete applied mathematics 154 (2006), no. 8 1236–1245.
- [274] R. Kohavi et. al., Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid., in Kdd, vol. 96, pp. 202–207, 1996.
- [275] S. Moro, P. Cortez, and P. Rita, A data-driven approach to predict the success of bank telemarketing, Decision Support Systems 62 (2014) 22–31.
- [276] D. S. Hochbaum and D. B. Shmoys, A best possible heuristic for the k-center problem, Mathematics of operations research 10 (1985), no. 2 180–184.
- [277] T. F. Gonzalez, Clustering to minimize the maximum intercluster distance, Theoretical computer science 38 (1985) 293–306.
- [278] D. Kai-Ineman and A. Tversky, Prospect theory: An analysis of decision under risk, Econometrica 47 (1979), no. 2 363–391.

- [279] A. Tversky and D. Kahneman, Advances in prospect theory: Cumulative representation of uncertainty, Journal of Risk and uncertainty 5 (1992) 297–323.
- [280] K. D. Edwards, Prospect theory: A literature review, International review of financial analysis 5 (1996), no. 1 19–38.
- [281] N. Barberis, M. Huang, and T. Santos, Prospect theory and asset prices, The quarterly journal of economics 116 (2001), no. 1 1–53.
- [282] N. C. Barberis, Thirty years of prospect theory in economics: A review and assessment, Journal of Economic Perspectives 27 (2013), no. 1 173–196.
- [283] J. Mercer, Prospect theory and political science, Annu. Rev. Polit. Sci. 8 (2005) 1–21.
- [284] R. Y. Hirokawa and M. S. Poole, Communication and group decision making, vol. 77. Sage, 1996.
- [285] D. Black, On the rationale of group decision-making, Journal of political economy 56 (1948), no. 1 23–34.
- [286] S. Kiesler and L. Sproull, Group decision making and communication technology, Organizational behavior and human decision processes 52 (1992), no. 1 96–123.
- [287] F. Tschan, N. K. Semmer, A. Gurtner, L. Bizzari, M. Spychiger, M. Breuer, and S. U. Marsch, Explicit reasoning, confirmation bias, and illusory transactive memory: A simulation study of group medical decision making, Small Group Research 40 (2009), no. 3 271–300.
- [288] D. Levi and D. A. Askay, *Group dynamics for teams*. Sage Publications, 2020.
- [289] M. G. Haselton, D. Nettle, and P. W. Andrews, The evolution of cognitive bias, The handbook of evolutionary psychology (2015) 724–746.
- [290] D. Kahneman and A. Tversky, Subjective probability: A judgment of representativeness, Cognitive psychology 3 (1972), no. 3 430–454.
- [291] D. M. Wegner, T. Giuliano, and P. T. Hertel, Cognitive interdependence in close relationships, Compatible and incompatible relationships (1985) 253–276.
- [292] H. P. Stott, Cumulative prospect theory's functional menagerie, Journal of Risk and uncertainty 32 (2006) 101–130.
- [293] D. Prelec, The probability weighting function, Econometrica (1998) 497–527.
- [294] H. Nilsson, J. Rieskamp, and E.-J. Wagenmakers, *Hierarchical bayesian parameter* estimation for cumulative prospect theory, Journal of Mathematical Psychology 55 (2011), no. 1 84–93.

- [295] P. Dixit and S. Silakari, Deep learning algorithms for cybersecurity applications: A technological and status review, Computer Science Review **39** (2021) 100317.
- [296] W. Jiang, H. Li, S. Liu, X. Luo, and R. Lu, Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles, IEEE transactions on vehicular technology 69 (2020), no. 4 4439–4449.
- [297] M. Castro, B. Liskov, et. al., Practical byzantine fault tolerance, in OsDI, vol. 99, pp. 173–186, 1999.
- [298] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, Generative modeling of multimodal multi-human behavior, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3088–3095, IEEE, 2018.
- [299] Y. Ren, V. Cedeno-Mieles, Z. Hu, X. Deng, A. Adiga, C. Barrett, S. Ekanayake,
  B. J. Goode, G. Korkmaz, C. J. Kuhlman, et. al., Generative modeling of human behavior and social interactions using abductive analysis, in 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 413–420, IEEE, 2018.
- [300] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, Generative agents: Interactive simulacra of human behavior, arXiv preprint arXiv:2304.03442 (2023).
- [301] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, et. al., Imitating human behaviour with diffusion models, arXiv preprint arXiv:2301.10677 (2023).
- [302] D. P. Kingma and M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).
- [303] O. Askarisichani, E. Y. Huang, K. S. Sato, N. E. Friedkin, F. Bullo, and A. K. Singh, Expertise and confidence explain how social influence evolves along intellective tasks, arXiv preprint arXiv:2011.07168 (2020).
- [304] D. Jarrett, A. Hüyük, and M. van der Schaar, Online decision mediation, in Advances in Neural Information Processing Systems (A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds.), 2022.
- [305] M.-A. Charusaie, H. Mozannar, D. Sontag, and S. Samadi, Sample efficient learning of predictors that complement humans, in International Conference on Machine Learning, pp. 2972–3005, PMLR, 2022.
- [306] D. Asmar and M. Kochenderfer, Collaborative decision making using action suggestions, in Advances in Neural Information Processing Systems (A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds.), 2022.

- [307] Q. Li, Z. Peng, H. Wu, L. Feng, and B. Zhou, Human-AI shared control via policy dissection, in Thirty-Sixth Conference on Neural Information Processing Systems, 2022.
- [308] M. Sclar, G. Neubig, and Y. Bisk, Symmetric machine theory of mind, in International Conference on Machine Learning, pp. 19450–19466, PMLR, 2022.
- [309] K. Zheng, S. Cai, H. R. Chua, W. Wang, K. Y. Ngiam, and B. C. Ooi, Tracer: A framework for facilitating accurate and interpretable analytics for high stakes applications, in Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 1747–1763, 2020.
- [310] M. Kosan, Z. Huang, S. Medya, S. Ranu, and A. Singh, *Global counterfactual* explainer for graph neural networks, arXiv preprint arXiv:2210.11695 (2022).
- [311] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving, in Proceedings of the IEEE international conference on computer vision, pp. 2722–2730, 2015.
- [312] T. Stevens, Knowledge in the grey zone: Ai and cybersecurity, Digital War 1 (2020), no. 1 164–170.
- [313] V. Lai, C. Chen, Q. V. Liao, A. Smith-Renner, and C. Tan, Towards a science of human-ai decision making: a survey of empirical studies, arXiv preprint arXiv:2112.11471 (2021).
- [314] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, A survey of human-in-the-loop for machine learning, Future Generation Computer Systems (2022).
- [315] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, *Recommender systems survey*, *Knowledge-based systems* 46 (2013) 109–132.
- [316] S. Zhang, L. Yao, A. Sun, and Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Computing Surveys (CSUR) 52 (2019), no. 1 1–38.
- [317] Y. Peng, A survey on modern recommendation system based on big data, arXiv preprint arXiv:2206.02631 (2022).
- [318] R. Sabitha, S. Vaishnavi, S. Karthik, and R. Bhavadharini, User interaction based recommender system using machine learning, Intelligent Automation and Soft Computing 31 (2022), no. 2 1037–1049.
- [319] R. M. Monarch, Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI. Simon and Schuster, 2021.

- [320] Y. Liang, L. He, and X. Anthony'Chen, Human-centered ai for medical imaging, Artificial Intelligence for Human Computer Interaction: A Modern Approach (2021) 539–570.
- [321] H. Dong, V. Suárez-Paniagua, W. Whiteley, and H. Wu, Explainable automated coding of clinical notes using hierarchical label-wise attention networks and label embedding initialisation, Journal of biomedical informatics 116 (2021) 103728.
- [322] T. Dash, S. Chitlangia, A. Ahuja, and A. Srinivasan, A review of some techniques for inclusion of domain-knowledge into deep neural networks, Scientific Reports 12 (2022), no. 1 1–15.
- [323] Y. Kang, Y.-W. Chiu, M.-Y. Lin, F.-Y. Su, and S.-T. Huang, Towards model-informed precision dosing with expert-in-the-loop machine learning, in 2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI), pp. 342–347, IEEE, 2021.
- [324] C. Chandler, P. W. Foltz, and B. Elvevåg, Improving the applicability of ai for psychiatric applications through human-in-the-loop methodologies, Schizophrenia Bulletin (2022).
- [325] Z. Liu, J. Wang, S. Gong, H. Lu, and D. Tao, Deep reinforcement active learning for human-in-the-loop person re-identification, in Proceedings of the IEEE/CVF international conference on computer vision, pp. 6122–6131, 2019.
- [326] S. Budd, E. C. Robinson, and B. Kainz, A survey on active learning and human-in-the-loop deep learning for medical image analysis, Medical Image Analysis 71 (2021) 102062.
- [327] G. Shani and A. Gunawardana, *Evaluating recommendation systems*, in *Recommender systems handbook*, pp. 257–297. Springer, 2011.
- [328] M. Zehlike, K. Yang, and J. Stoyanovich, Fairness in ranking, part ii: Learning-to-rank and recommender systems, ACM Computing Surveys (CSUR) (2022).
- [329] Y. Zhang, X. Chen, et. al., Explainable recommendation: A survey and new perspectives, Foundations and Trends® in Information Retrieval 14 (2020), no. 1 1–101.
- [330] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et. al., Wide & deep learning for recommender systems, in Proceedings of the 1st workshop on deep learning for recommender systems, pp. 7–10, 2016.

- [331] H. Wang, N. Wang, and D.-Y. Yeung, Collaborative deep learning for recommender systems, in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1235–1244, 2015.
- [332] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, *Representation learning for dynamic graphs: A survey.*, *JMLR* 21 (2020), no. 70 1–73.
- [333] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, Time-series representation learning via temporal and contextual contrasting, arXiv preprint arXiv:2106.14112 (2021).
- [334] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, S. Wu, C. Smyth, P. Poupart, and M. Brubaker, *Time2vec: Learning a vector* representation of time, arXiv preprint arXiv:1907.05321 (2019).
- [335] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, Unsupervised scalable representation learning for multivariate time series, Advances in neural information processing systems **32** (2019).
- [336] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, A transformer-based framework for multivariate time series representation learning, in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 2114–2124, 2021.
- [337] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 2828–2837, 2019.