

UC Santa Barbara

UC Santa Barbara Electronic Theses and Dissertations

Title

Models of Propagation in Networks

Permalink

<https://escholarship.org/uc/item/092706b9>

ISBN

9798241680303

Author

Sharma, Saurabh

Publication Date

2026-03-20

Peer reviewed|Thesis/dissertation

University of California
Santa Barbara

Models of Propagation in Networks

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

Doctor of Philosophy

in

Computer Science

by

Saurabh Sharma

Committee in charge:

Professor Ambuj Singh, Chair

Professor Joao Hespanha

Professor Jie Gao

Professor Sanjukta Krishnagoal

March 2026

The Dissertation of Saurabh Sharma is approved.

Professor Joao Hespanha

Professor Jie Gao

Professor Sanjukta Krishnagoal

Professor Ambuj Singh, Committee Chair

March 2026

Models of Propagation in Networks

Copyright © 2026

by

Saurabh Sharma

I dedicate this PhD thesis to the young American boy I bumped into at a public park the other day. He was kind enough to say hi to me. I forget his name now, but he said he was $3\frac{1}{2}$ years old. I salute his precision in articulating his age, and communicate my research findings here in a likewise manner.

Acknowledgements

First and foremost, I want to thank my supervisor Prof. Ambuj Singh for his constant feedback and thought-provoking discussions during the course of this PhD. I would also like to thank myself for playing the critic against myself. My rigorous cross-examination and critique helped me structure my ideas, experiments and presentation significantly.

Next, I would like to thank my parents and my brother, who have been constant pillars of support throughout my journey. Without them, I would not have had the audacity to even begin this long phase of research.

Lastly, I acknowledge the traditional custodians of the land upon which the University of California, Santa Barbara is located, and pay my respect to the Chumash Elders past, present and future for they hold the memories, the traditions, and the culture of this area, which has become a place of learning for people from all over the world.

Saurabh Sharma | Resume

University of California, Santa Barbara – Santa Barbara, California

☎ (805)-570-9317 • ✉ saurabhsharma@ucsb.edu

Scholar | Github | LinkedIn

Education

University of California, Santa Barbara <i>PhD in Computer Science</i> Thesis: Models of Propagation in Networks.	Santa Barbara 10/2020–Present
Universität des Saarlandes <i>Masters in Computer Science</i> Thesis: Long-tailed recognition using class-balanced experts.	Saarbrücken 4/2019–8/2020
Indian Institute of Technology Delhi <i>Bachelors in Computer Science and Engineering</i>	New Delhi 8/2012–5/2016

Academic experience

University of California, Santa Barbara <i>Graduate Student Researcher, Dynamic networks analysis and modelling group</i>	Santa Barbara 10/2020–Present
- Working on data-driven models of product adoption in social networks using graph neural networks. - Teaching assistant for various undergraduate courses and mentoring undergraduate students in early research programs.	
Max Planck Institute for Informatics <i>Researcher, Computer vision and machine learning group</i>	Saarbrücken 8/2018–8/2020
- Worked on zero-shot learning, few-shot learning and long-tailed datasets in image recognition. - Presented and lead discussion on vision and ML research papers in seminars. - Worked closely with PhD students, postdocs and faculty researchers from various backgrounds.	
Indian Institute of Technology, Bombay <i>Research Assistant, Computer vision group</i>	Mumbai 2/2018 - 7/2018
- Worked on monocular 3D human pose estimation from images and synthesizing realistic human images. - Collaborated with a diverse group of student and faculty researchers, conducted interviews for new candidates.	

Industry experience

Amazon <i>Applied Scientist Intern, ML Accelerator team, Selling Partner Support</i>	San Diego 6/2023 - 9/2023
- Developed a retrieval-augmented generation (RAG) pipeline for seller support using publicly available help articles. - Mined retriever ground truth and fine-tuned SBERT based deep retriever along with graph-based regularization. - Engineered prompts and analyzed feasibility of responses generated from FalconLite LLM on AWS.	
Amazon <i>Applied Scientist Intern, ML Accelerator team, Selling Partner Support</i>	Remote 6/2022 - 9/2022
- Developed heterogenous GNNs to leverage multi-attribute pathways for payment fraud and review abuse detection. - Extensive exploration into model architecture and learning strategies for GNN-based models in this domain.	
Goldman Sachs <i>Quantitative Analyst, Market Risk team</i>	Bangalore 6/2016 - 3/2017

- Developed software for automatic deployment of VaR and stress-testing risk computation on large-scale CPU cluster.
- Coordinated with various teams and stakeholders for risk reporting at various cadences as mandated by the SEC.

Publications

Sharma, Singh. Emergence of Phase Transitions in Complex Contagions.

Submitted to KDD, 2026.

Sharma, Singh. Dynamic Gradient Influencing for Viral Marketing Using Graph Neural Networks.

In: ***ACM Web Conference, 2025.***

Sharma, Xian, Yu, Singh. Learning Prototype Classifiers for Long-Tailed Recognition.

In: ***International Joint Conference on Artificial Intelligence, 2023.***

Sharma, Yu, Fritz, Schiele. Long-Tailed Recognition Using Class-Balanced Experts.

In: ***DAGM German Conference on Pattern Recognition, 2020.***

Xian, Sharma, Schiele, Akata. f-VAEGAN-D2: A Feature Generating Framework for Any-Shot Learning.

In: ***IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.***

Sharma, Varigonda, Sharma, Jain. Monocular 3D Human Pose Estimation by Generation & Ordinal Ranking.

In: ***IEEE/CVF International Conference on Computer Vision, 2019.***

Technical Reports

Sharma, Fazeli, Yu, Li, Retrieval Augmented Generation for Seller Support.

Amazon Internal Report 2023.

Sharma, Padmanabhan. MP-PROP: Adapting Risk Propagation to Multi-partite Graphs.

Amazon Internal Report 2022.

Awards

2020-2025: Regents in Computer Science fellowship.

Academic service

- | | |
|---|-------------------------------------|
| ◦ Reviewer for NeurIPS 24, 25. | ◦ Reviewer for ICLR 25. |
| ◦ Reviewer for ICML 25. | ◦ Reviewer for KDD24, KDD25, KDD26. |
| ◦ Reviewer for CVPR 21, 22, 23, 24, 25, 26. | ◦ Reviewer for ICCV 21, 23. |

Teaching experience

26: Teaching Assistant for Introduction to Data Structures and Algorithms at UC Santa Barbara.

21, 24, 25: Teaching Assistant for Introduction to Numerical Computing at UC Santa Barbara.

23: Teaching Assistant for Introduction to Computer Vision at UC Santa Barbara.

21: Teaching Assistant for Machine Learning (Focus on Deep Learning) at UC Santa Barbara.

21: Teaching Assistant for Capstone course in Data Science at UC Santa Barbara.

Open-source contributions

- | | |
|---------------------------------|-------------------------------|
| ◦ Code for our WebConf25 paper. | ◦ Code for our IJCAI23 paper. |
| ◦ Code for our GCPR20 paper. | ◦ Code for our ICCV19 paper. |

Abstract

Models of Propagation in Networks

by

Saurabh Sharma

The diffusion of ideas, innovations and behaviors is a complex sociological phenomena for which various computational models have been proposed by economists, algorithmists and ML researchers. However, it remains insufficiently understood due to lack of propagation data on networks and the complex interplay between dynamic network structure and node-level features. Moreover, real-world networks exhibit skewed degree distributions and propagation size distributions, thereby necessitating learning models that account for imbalanced distributions and data scarcity.

The goal of this thesis is two-pronged: (1) How do coupled representation and classifier learning models cope with imbalanced data distributions, particularly with minority classes, and how can their weaknesses in limited data settings be mitigated?, (2) How can dynamics of propagation on real-world networks, themselves admitting a natural power law degree distribution, be understood using the machinery of gradient-based optimization and high-dimensional feature representations? The two goals are naturally coupled; distributional analysis is a powerful tool for network (propagation) analysis, and network propagation modeling suffers from a lack of ground truth data, which necessitates a combination of data-driven and synthetic models rooted in sociological insights. Therefore, the major theme of this thesis is models of propagation in networks, and the minor is the limited data learning setting.

Contents

Curriculum Vitae	vi
Abstract	viii
1 Introduction	1
2 Preliminaries	6
2.1 Network Science	6
2.1.1 Real-World Network Models	6
2.1.2 Real-World Network Processes	7
2.2 Machine Learning	8
2.2.1 Maximum Likelihood Estimation	8
2.2.2 Model Optimization from Backpropagation	10
2.2.3 Robustness to Imbalanced Distributions	10
2.2.4 Robustness to Adversarial Perturbations	12
2.3 Graph Machine Learning	13
2.3.1 Graph Signal Processing	13
2.3.2 Graph Neural Networks	13
2.3.3 Robustness to Imbalanced Distributions	14
2.3.4 Robustness to Adversarial Perturbations	16
3 Learning on Imbalanced Distributions	17
3.1 Introduction	17
3.2 Related Work	20
3.3 Our Approach	22
3.3.1 Prototype Classifier	22
3.3.2 Stable Gradient Updates on Prototypes	23
3.3.3 Addressing the Biased Softmax Problem	24
3.3.4 Channel-Dependent Temperatures	25
3.3.5 Logit Adjustment for Learning Prototypes	26
3.4 Experiments	26

3.4.1	Datasets	26
3.4.2	Evaluation Protocol	28
3.4.3	Implementation Details	28
3.4.4	Ablation Study	29
3.4.5	Prototype Norms	32
3.4.6	How Do Prototypes Evolve During Training?	33
3.5	Comparison to State-of-the-Art	34
3.6	Takeaways	35
3.7	Extended Empirical Analysis	35
3.7.1	Dataset statistics	35
3.7.2	More training details	36
3.7.3	Temperature schemes	36
3.7.4	Effect of logit adjustment weight	38
3.7.5	Results on CIFAR10-LT	39
4	Network Perturbation for Viral Marketing	43
4.1	Introduction	43
4.2	Related Work	46
4.3	Preliminaries	47
4.4	Dynamic Viral Marketing Problem	49
4.4.1	Relating DVM to Influence Maximization	52
4.5	Dynamic Gradient Influencing Framework	55
4.5.1	Gradient-Guided Node Flipping	55
4.5.2	Node Flipping Budget Computation	57
4.5.3	Meta Influence Using Meta Attribute Flips	58
4.6	Experiments	60
4.6.1	Performance Comparison (RQ1)	63
4.6.2	Strategy of Meta Influence (RQ2)	63
4.6.3	Cascades Created by DGI (RQ3)	65
4.6.4	Properties of Intermediary Spreaders (RQ4)	67
4.7	Takeaways	68
4.8	Extended Analysis	69
4.8.1	Another Proof of NP-Hardness	69
4.8.2	Implementation Details	70
4.8.3	Results on Large-Scale Data	73
4.8.4	Ablation Study	74
4.8.5	Complexity	75
4.8.6	Sensitivity Analysis	77
4.8.7	Cascades in Adversarial Perturbations	79

5	Unified Propagation Model for Complex Contagion	82
5.1	Introduction	82
5.2	Related Work	87
5.3	Method	90
5.3.1	Preliminaries	90
5.3.2	Unified Propagation of Complex Contagion	92
5.3.3	Conditions for Local Incubation	95
5.3.4	Connection to Random Walks	96
5.3.5	Phase Transition Behavior	97
5.4	Experiments	98
5.4.1	RQ1: Spread Distributions.	99
5.4.2	RQ2: Spread Evolution toward Virality.	100
5.4.3	RQ3: Time to Virality	102
5.4.4	RQ4: Effect of the Model Parameters.	103
5.4.5	RQ5: Propagation of Misaligned Ideas.	106
5.4.6	RQ6: Real-world Complex Contagion.	107
5.5	Discussion	109
5.6	Takeaways	110
5.7	Extended Analysis	111
5.7.1	Comparisons to Prior Propagation Models	111
5.7.2	Data-Driven Propagation Models	114
5.7.3	Choice of Propagation Function	114
5.7.4	Maximum Likelihood Formulation	115
5.7.5	Prediction of Activation States	116
5.7.6	Limitations of Prior Data-Driven Propagation Models	118
5.7.7	Extended Propagation Model	119
5.7.8	Sampling Prop. Vectors to Maximize Spread	121
5.7.9	Candidate Generation	122
5.7.10	Monte Carlo Estimation of Spread	123
5.7.11	Beam Search Refinement	123
5.7.12	Dynamic Programming Approximation	124
5.7.13	Approximate Algorithm	124
6	Conclusion	126
	Bibliography	129

Chapter 1

Introduction

Real-world networks and processes unfolding over them have intrigued mathematicians, physicists, economists, social scientists, neuroscientists, and computer scientists for a very long time [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. The central theme among all these disciplines is to find an underlying mechanistic framework that gives rise to the network structures and processes observed in the real-world, which are decidedly different from regular networks or even purely random networks. Importantly, network structure modulates the overlying network process and likewise an unfolding network process can modify the underlying network structure.

Propagation in networks has recently been studied through data-driven ML models [12, 13, 14, 15], which rely upon, among other things, graph neural networks (or GNNs) [16, 17, 18]. The term “network” is overloaded with multiple senses in this regard, referring to both a real-world network such as a social network, citation network or the World-Wide-Web, as well as a layerwise matrix multiplication with intermediate non-linearities whose parameters can be optimized using machine learning, i.e., an artificial neural network (ANN). Therefore, it’s important to be clear at the outset that this thesis concerns the study of propagation in real-world networks, however, we do employ

tools such as GNNs for this purpose.

It's pertinent to look at some canonical mechanisms driving network structure and network propagation before proceeding further. The primary driver of network formation is *homophily*, through various attributes, such as age, gender, shared tastes, or even number of friends. Homophily is considered a strong driver of link formation in social networks [4], as depicted in Fig. 1.1. Nowadays, node attribute data can be easily mined from on-

line social networks (OSNs) and these are in fact used by OSN providers to suggest links to people online. The age of OSNs has therefore resulted in a rich trove of data [19] for research into real-world attributed networks. Moreover, we consider here a

few classical network propagations, for instance, book adoptions through recommendation chains and the spread of TB within a network. These are depicted in Fig. 1.2 and demonstrate a typical modular, hub-and-spoke structure where high-degree influencers cause star-like propagations in their ego-networks, while low-degree nodes act as intermediary bridges for diffusion into other communities. Further, predicting book and in general product adoption through word-of-mouth spreading is an important problem in operations research [20], while predicting disease contagion is central for epidemiology.

Since real-world network data has historically been sparse, various synthetic models have been used to sample networks that exhibit similar properties. We contrast two models specifically of interest here and they're depicted in Fig. 1.3. The first model was proposed by Paul Erdős and Alfréd Rényi [23], which while being simple is remarkable

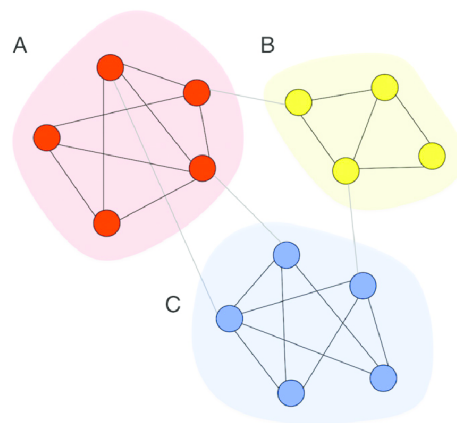


Figure 1.1: Illustration of homophily, in this case by node color, as a causal mechanism of network structure.

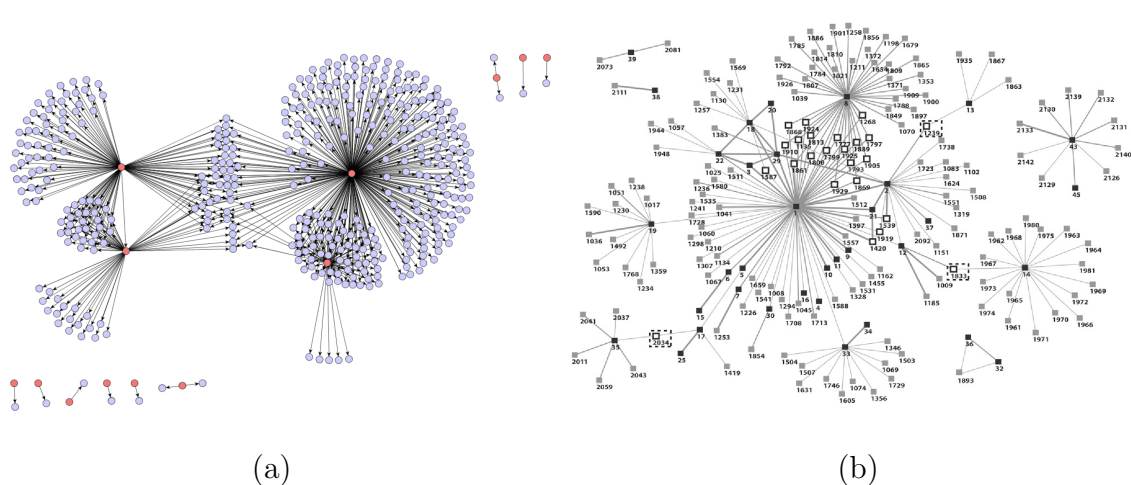


Figure 1.2: Illustration of cascading spread in (a) Book recommendations [21], (b) TB epidemic [22].

for its tractability and good properties. The Erdős-Rényi (ER) network model assumes all links are equally likely, and is therefore specified by two parameters only, the number of nodes, n , and the probability of link formation, p . They studied, among other things, the degree distribution entailed by this random network model, which turned out to be a Poisson distribution centered around np , the average degree of the network. They were also interested in the "evolution" of the random network as links were added, and at what thresholds did certain structural properties such as presence of certain subgraphs (trees, cycles of given order, complete subgraphs, etc.) and global properties (connectivity, number of connected components, etc.) emerge. Particularly, they reported the formation of a "giant component" that contains the majority of the nodes when $p > \frac{1}{2}$, and absorbs most of the isolated trees, with "danger" of absorption higher for a bigger giant component.

In [3], the Barabasi-Albert model of preferential attachment (PA) was proposed to address the inability of ER and other network models to produce the characteristic scale-free degree distribution of real-world networks is addressed. Essentially, it has two

differences from prior work: (1) The sequence of node arrival is baked into the network formation, the nodes aren't all present apriori, and (2) Arriving nodes makes links to pre-existing nodes with probability proportional to their degree centrality. This results in an early-mover advantage where nodes present in initial stages end up having a higher degree centrality and nodes arriving later having low degree, and the cumulative degree distribution follows a power law whose exponent is 3. Crucially, for linear preferential attachment, the exponent doesn't depend on the number of links formed by each arriving node. Therefore, due to its closer similarity to real-world networks, we use networks sampled from the PA model to study propagations, particularly in Chapter ??.

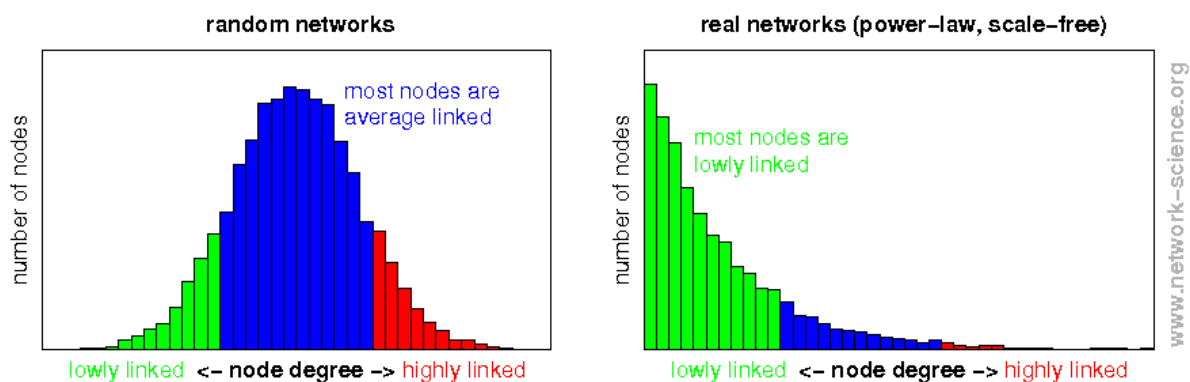


Figure 1.3: Illustration of the degree distribution found in random network models: (a) Erdos-Renyi graphs, Poisson distributed, and (b) Preferential Attachment graphs [3], exponential scale-free distribution.

Power-law distributions naturally arise in real-world datasets that haven't been artificially curated. This phenomena is qualitatively illustrated in Fig. 1.4. The presence of such distributions in the diverse domains of network science, computer vision and natural language processing prompts a fundamental question that forms the minor theme of this thesis: how do we learn machine learning models with imbalanced datasets? Specifically, how can ML models be made more robust to relative class imbalance between head

and tail classes, as well as data scarcity for tail classes? Moreover, how can we make such models trained on imbalanced distributions attain equivalent tail and head class performance at inference time?

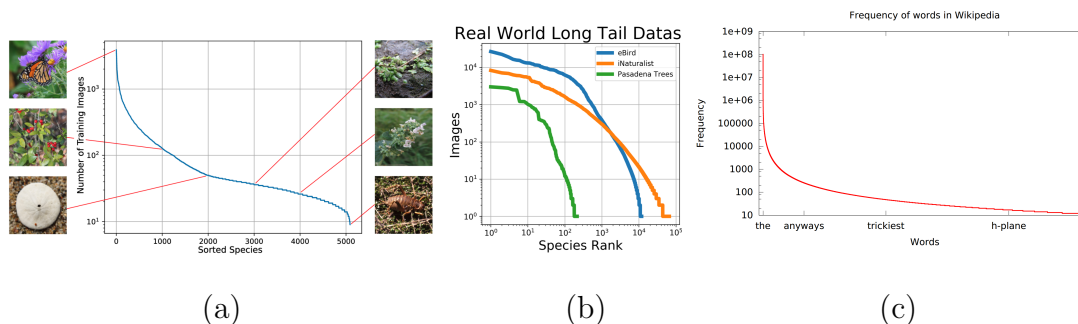


Figure 1.4: Illustration of the Zipfian power-law distribution found in real-world datasets: (a) iNaturalist [24], (b) Birds, iNaturalist, and Trees, and (c) Wikipedia words.

The remainder of this thesis follows the following structure. Chapter 2 presents preliminaries on prior work broadly in network science, machine learning and graph machine learning, along with some motivating experiments for the main chapters. Chapter ?? presents work on learning on imbalanced distributions, particularly concerning the problem of long-tailed image recognition. Chapter 4 presents a network perturbation scheme for viral marketing, particularly concerning a gradient-based network spread optimization scheme under a data-driven adoption model and budget constraints for sequential edge insertion and node attribute perturbations. Chapter ?? presents a unifying propagation model accounting for propagation affinity as well as local and global influences. By applying this model to Preferential Attachment (PA) networks, insights into phase transitions for viral concepts and existing sociological theories about the power of the periphery are provided. Finally, Chapter 6 concludes the thesis with closing remarks.

Chapter 2

Preliminaries

2.1 Network Science

2.1.1 Real-World Network Models

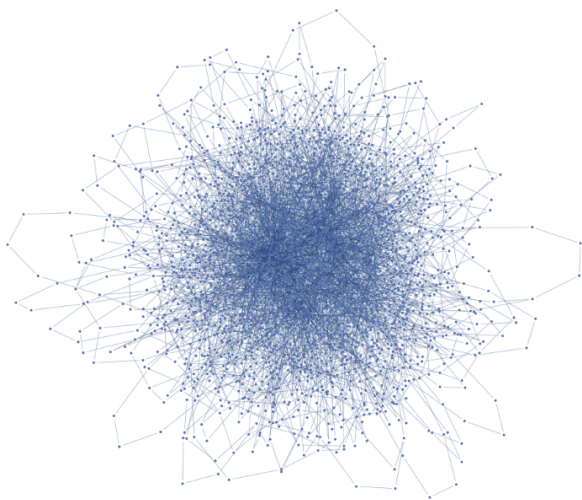


Figure 2.1: Qualitative visualization of a Preferential Attachment (PA) network and its associated core-periphery structure.

Real-world systems are naturally modeled as graphs, where nodes represent entities and edges encode interactions. Empirical studies show that such networks deviate significantly from random graph assumptions, instead exhibiting heavy-tailed degree distributions, clustering, and community structure [3, 1]. As shown in Fig. 2.1, a small number of nodes act as hubs with disproportionately high connectivity, while the majority of nodes have low degree. This structural heterogeneity is a

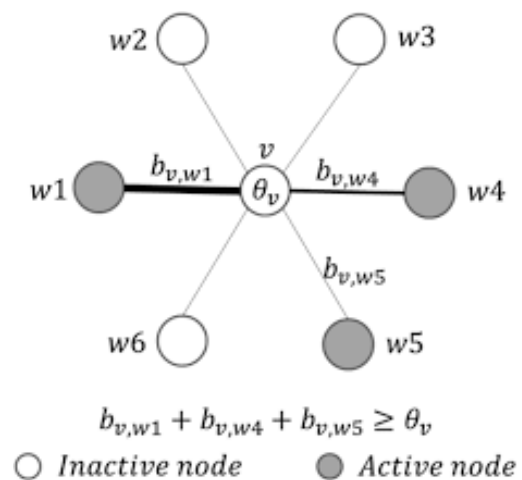
defining characteristic of real-world networks.

These properties have direct implications for downstream learning. Heavy-tailed connectivity introduces imbalance in influence, while community structure constrains information flow. Models that ignore these structural biases often fail to generalize across nodes with varying connectivity patterns. The structure illustrated in Fig. 2.1 highlights the need for learning methods that account for both local and global graph statistics when modeling real-world systems.

2.1.2 Real-World Network Processes



(a)



(b)

Figure 2.2: L-R: (a) Depiction of Stanley Millgram’s experiments in social psychology [25]. Participants were asked to route letters to the address marked on the envelope through their personal networks. Findings showed that high degree nodes usually acted as intermediaries as the letters made their way from the MidWest to the East Coast. (b) Granovetter’s Linear Threshold (LT) model for discrete time-step cascades in a network [26].

In addition to evolving structure, real-world networks support dynamic processes such as diffusion, cascade propagation, and opinion dynamics. Fig. 2.2a depicts the early experiments by Stanley Millgram in social networks, where letters from the Midwest could be routed to the East Coast by intermediary high-degree nodes. Further, classical models such as Linear Threshold (LT), depicted in Fig. 2.2b, provide a discrete time-step based model of how information spreads [27]. Likewise, the Independent Cascade (IC) model, depicted in Fig. 2.3, provides a discrete time step based model where nodes get one chance to probabilistically infect their neighbors. Similar to the LT model in that it's based on a linear activation rule, the networked coordination game model depicted in Fig. 2.4 triggers cascades based on a game-theoretic payoff of aligning self-behavior with neighbors' behaviors.

However, real-world processes are more complex, involving stochastic interactions, heterogeneous transmission probabilities, and temporal dependencies. The interplay between topology and dynamics introduces challenges for modeling. Highly connected nodes can amplify propagation, while community boundaries can inhibit it. Furthermore, propagation is often influenced by latent factors not captured by simple models. Diffusion patterns vary significantly depending on both structure and node behavior, motivating more realistic propagation models.

2.2 Machine Learning

2.2.1 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a fundamental principle for parameter learning. Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ and model $p_\theta(y|x)$, MLE seeks parameters θ that maximize the likelihood:

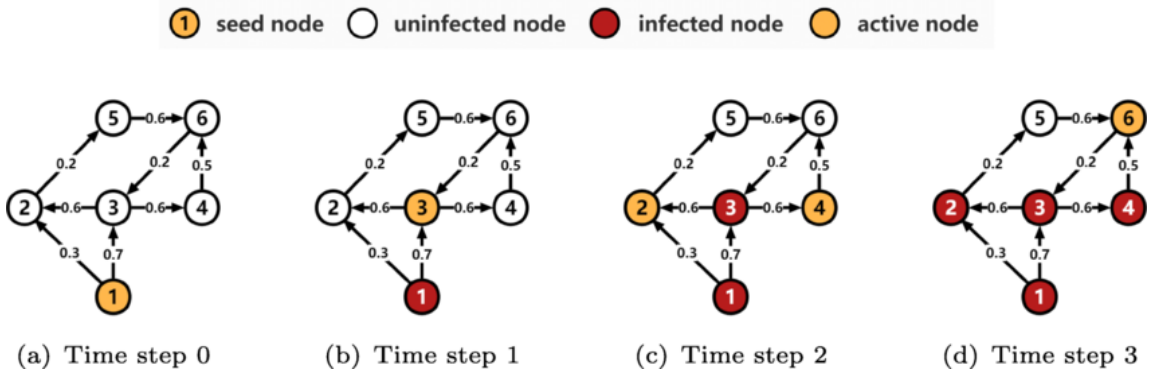


Figure 2.3: The Independent Cascade (IC) model for discrete time-step cascades [28]. Each node gets one chance to infect others probabilistically once it's activated.

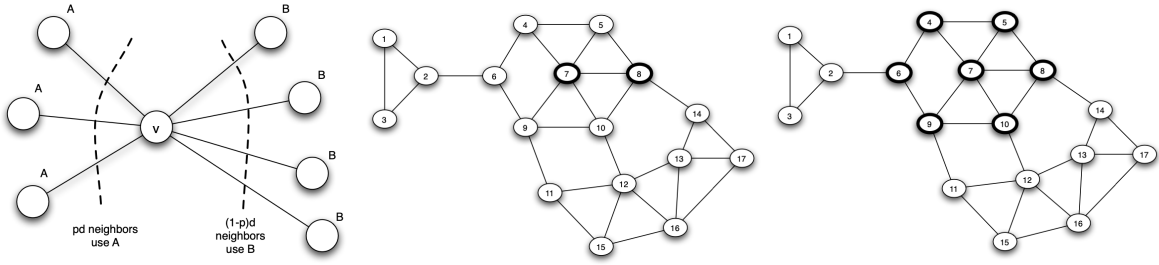


Figure 2.4: A depiction of Kleinberg's networked coordination game model [6]. (a) v must choose between behavior A and behavior B , based on what its neighbors are doing, (b) Two nodes are the initial adopters, and (c) The process ends after 3 steps. Starting with nodes 7 and 8, the new behavior A spreads to some but not all the remaining nodes.

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^N p_{\theta}(y_i | x_i)$$

Equivalently, this is written as minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = - \sum_{i=1}^N \log p_{\theta}(y_i | x_i)$$

In classification tasks, this reduces to cross-entropy loss. MLE provides a principled statistical framework, but it assumes that the training distribution matches the true data

distribution.

In practice, this assumption often fails, especially under class imbalance or distribution shift. As a result, models trained with MLE may overfit majority patterns while underrepresenting rare but important cases. This limitation motivates alternative training strategies that improve robustness beyond likelihood maximization.

2.2.2 Model Optimization from Backpropagation

Modern machine learning models are optimized using gradient-based methods, with backpropagation providing an efficient mechanism to compute gradients. Given a loss function $\mathcal{L}(\theta)$, parameters are updated via:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(\theta)$$

where η is the learning rate. Backpropagation computes gradients by recursively applying the chain rule through the computational graph [29].

While effective, optimization dynamics strongly influence generalization. Non-convex loss landscapes, noisy gradients, and hyperparameter choices all affect convergence. Importantly, standard backpropagation does not account for imbalance or adversarial effects, which can lead to biased or unstable solutions. These issues motivate modifications to training procedures that explicitly improve robustness.

2.2.3 Robustness to Imbalanced Distributions

Class imbalance arises when class frequencies are highly skewed, as depicted in Fig. 2.6. Under standard training, models prioritize majority classes, resulting in poor performance on minority classes. This effect is due to both representation bias and classifier bias [30, 31]. In effect, imbalance distorts learned feature spaces and decision boundaries.

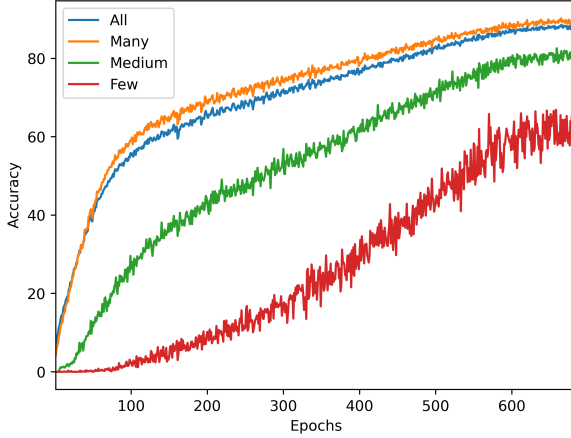


Figure 2.5: Evolution of classifier performance with training iterations on the long-tailed CIFAR100-LT dataset with imbalance ratio 200.

Fig. 2.5 shows how performance on different bands in the spectrum of class sizes evolves in standard training.

To mitigate these issues, techniques such as re-weighting, re-sampling, and decoupled training have been proposed. These approaches adjust the influence of samples during training to better reflect their importance. As shown in Fig. 2.7, the simple strategy of re-weighting classifier norm can significantly improve performance on underrepresented classes. Robust learning under imbalance is therefore essential in real-world applications.

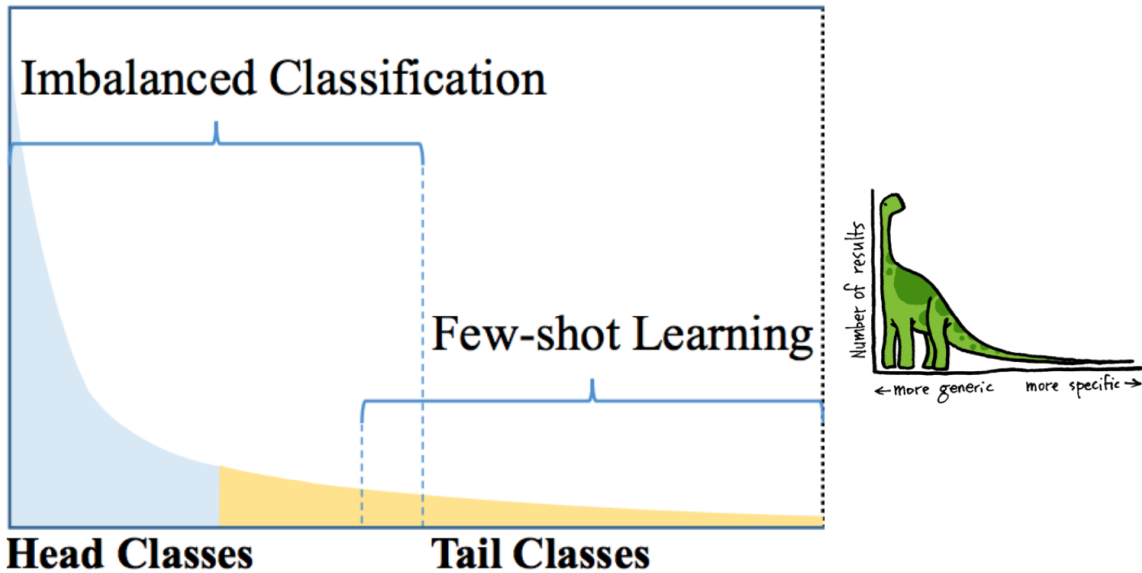


Figure 2.6: The problem of long-tailed image recognition (LTR) [32].

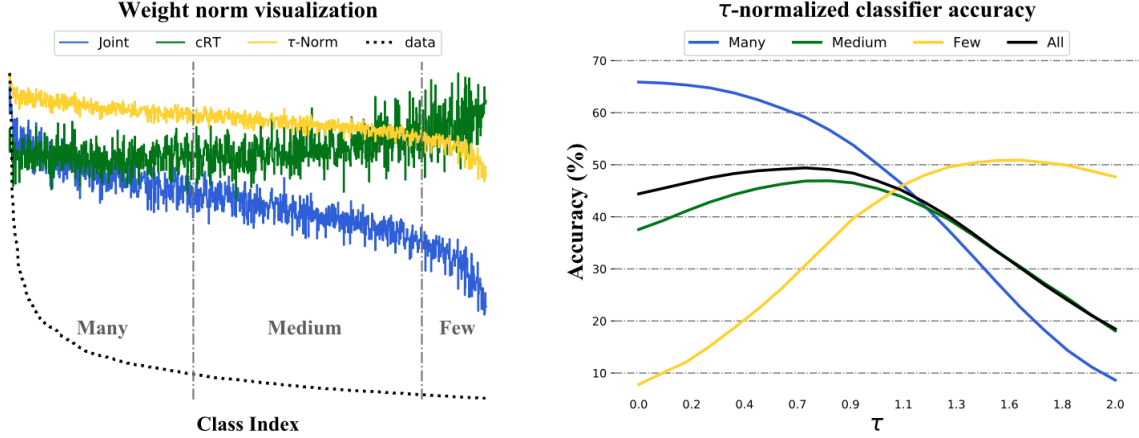


Figure 2.7: (a) The correlation of softmax classifier norm to class size [31], and (b) Effect of classifier renormalization on performance across the spectrum of classes.

2.2.4 Robustness to Adversarial Perturbations

Adversarial perturbations reveal vulnerabilities in learned models, where small input changes can cause large prediction shifts [33]. These perturbations often exploit linearities in high-dimensional models. Fig. 2.8 illustrates how imperceptible changes can lead to incorrect classifications.

Robustness to such perturbations requires modifying training objectives or architectures. Approaches such as adversarial training improve resilience but often incur computational cost. Further, robustness is closely tied to the geometry of decision boundaries, emphasizing the need for stable and semantically meaningful representations.

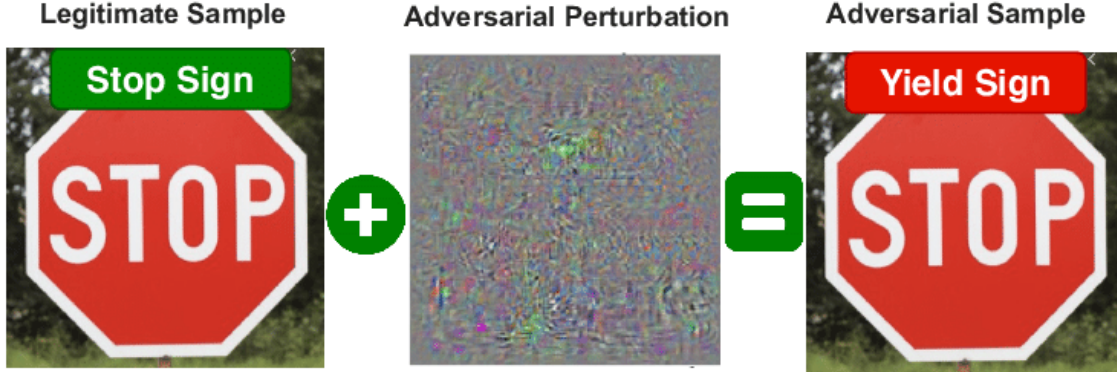


Figure 2.8: Qualitative illustration of an adversarial attack on an image classifier [33].

2.3 Graph Machine Learning

2.3.1 Graph Signal Processing

Graph Signal Processing (GSP) extends classical signal processing to graph domains. Given a graph Laplacian L , signals can be analyzed in the spectral domain via eigen decomposition. This enables filtering operations that generalize convolution to irregular structures [34].

These ideas form the theoretical foundation for many graph learning methods. By interpreting graph operations as localized filters, GSP provides insight into how information propagates across networks. This perspective is crucial for understanding modern graph neural architectures.

2.3.2 Graph Neural Networks

Graph Neural Networks (GNNs) learn node representations through iterative message passing. They are widely employed in transductive semi-supervised learning on graphs, particularly for node classification and link prediction, as depicted in Fig. 2.9. Typically,

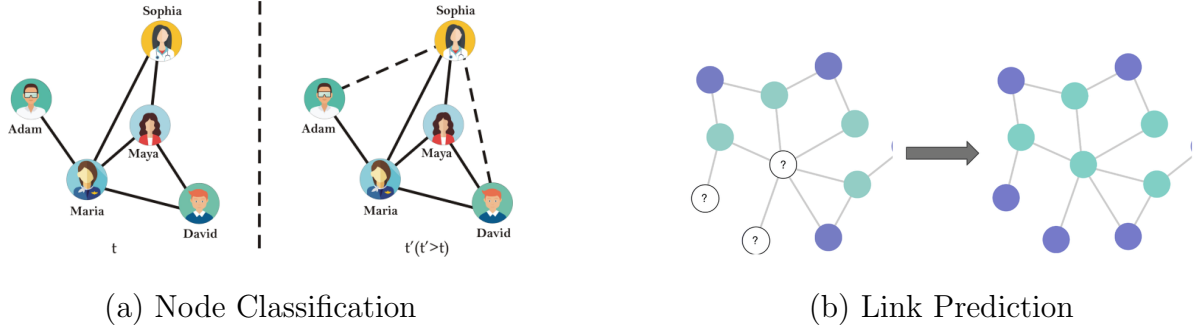


Figure 2.9: Depiction of the two essential problems in transductive semi-supervised learning on graphs [16].

at layer k of a GNN, node embeddings are updated as:

$$h_i^{(k+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} W^{(k)} h_j^{(k)} \right)$$

where $\mathcal{N}(i)$ denotes neighbors of node i . This process aggregates local information to build expressive representations [16, 35].

Despite their success, GNNs face challenges such as over-smoothing and sensitivity to graph structure since repeated aggregation can blur distinctions between nodes. Designing architectures that balance locality and discrimination remains an active area of research.

2.3.3 Robustness to Imbalanced Distributions

Imbalance in graph data arises when certain node classes are underrepresented. This problem is compounded by structural dependencies, where minority nodes may also have limited connectivity. Fig. 2.10 illustrates how imbalance affects representation quality at boundary nodes in graph settings. Further, our experiments with Deep Anomaly Detection [?], whose pipeline is depicted in Fig. 2.11, reveal that graph autoencoders

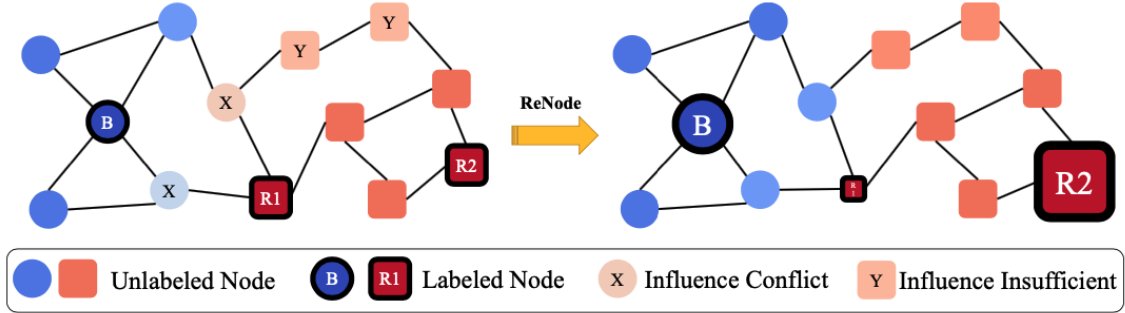


Figure 2.10: Effect of topological imbalance in graph learning. Reweighting center and boundary nodes in the training loss based on how much the influence in their neighborhood conflicts with other labelled nodes alleviates the problem [36].

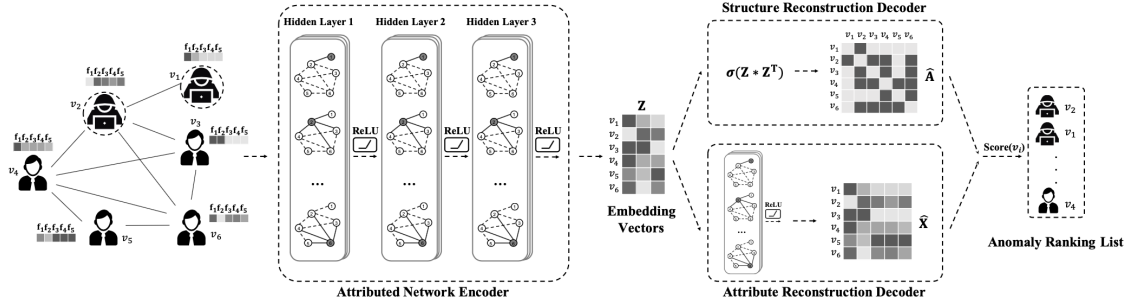


Figure 2.11: The pipeline for deep anomaly detection on attributed networks [37].

inherit bias towards degree centrality within the learnt node embedding norm, similar to what's observed in image data for class size. Therefore, degree centrality in network data plays a similar role to class size in image data. This correlation of node embedding norm to degree centrality is shown in Fig. 2.12.

Approaches to address this include re-weighting node contributions, modifying aggregation schemes, and designing specialized objectives. These methods aim to ensure that minority nodes are adequately represented during training. Consequently, handling imbalance in graphs requires accounting for both feature and structural factors.

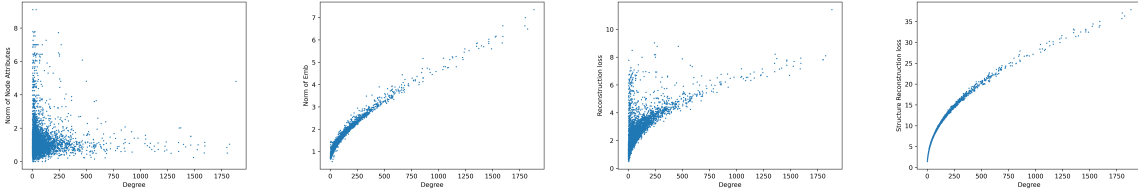


Figure 2.12: L-R: (a) Node attribute norm, (b) Node embedding norm, (c) Overall reconstruction error, and (d) Structure reconstruction error. While attribute norm in (a) is not correlated to the degree, the embedding norm is in (b). This imbalance can fool downstream tasks relying on this embedding, for eg. node classification, anomaly detection, link prediction. Moreover, both the total reconstruction (c) and structure reconstruction error (d) for anomaly detection are correlated to degree (undesirable).

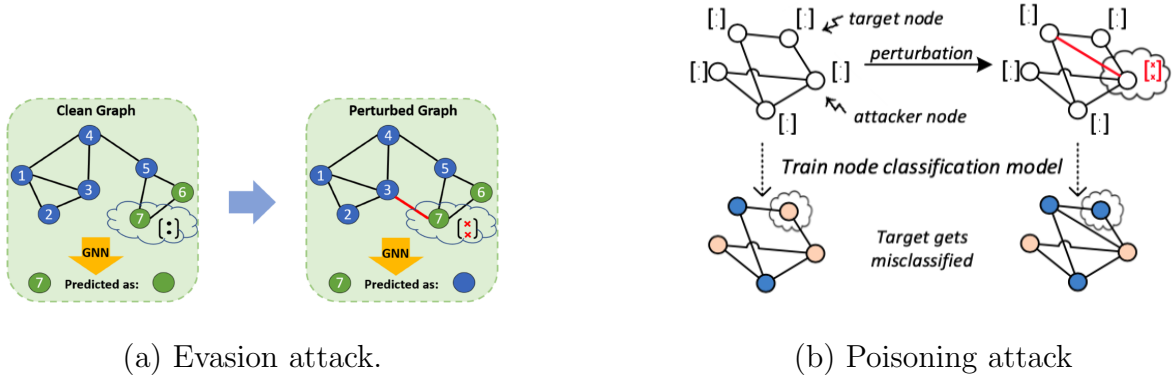


Figure 2.13: Depiction of adversarial attack on a GNN node classifier. (a) Evasion attacks are done at inference time, and (b) Poisoning attacks are done at training time.

2.3.4 Robustness to Adversarial Perturbations

Graph models are vulnerable to adversarial attacks that modify node features or edges. Even small structural changes can significantly alter predictions [38]. Fig. 2.13 provides a schematic representation of how such perturbations fool GNNs. Robust graph learning methods aim to mitigate these effects through adversarial training, robust aggregation, and graph sanitization.

Chapter 3

Learning on Imbalanced Distributions

3.1 Introduction

Imbalanced datasets are prevalent in the natural world due to the fundamental power-law distribution of objects [39]. Past decades have seen a lot of research in class-imbalanced learning [40, 41], and more recently, due to its far-reaching relevance in the era of deep-learning, the problem of long-tailed recognition (LTR) has received significant attention [30, 42, 32] in the field of computer vision.

A common underlying assumption of recent work in LTR is that softmax classifiers learned from imbalanced training datasets are biased towards head classes. They seek to rectify head class bias in the classifier through data-resampling [31], loss reshaping [43, 44, 45], ensemble-based models [46, 47, 48], and knowledge transfer [32, 49]. Particularly, it was shown in [31] that deep models trained on imbalanced datasets have a high correlation of classifier norm to class size. To achieve similar classifier norms, they freeze the feature extractor and retrain the classifier with balanced class sampling in a

second stage. Alternatively, they normalize every classifier by its L_2 norm or a power of its norm, leading to balanced classifier norms. More recently, [50] showed that balanced weights in both the feature extractor and classifier can be achieved by regularizing the model with higher weight decay and/or MaxNorm constraints.

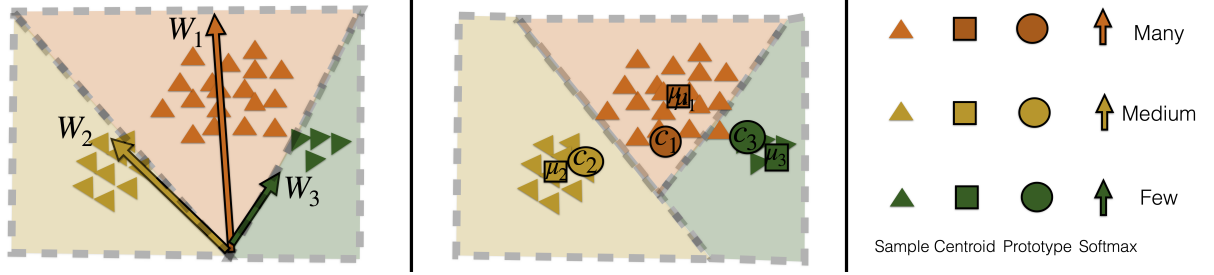


Figure 3.1: An illustration of Softmax vs Prototype classifiers for long-tailed data. Softmax classifiers have both a direction and a magnitude, indicated by the orientation and length of the classifier vector. During imbalanced training, the length of softmax classifiers gets correlated to the class size and leads to classification boundaries biased towards classes with Many samples. In contrast, prototype classifiers circumvent this shortcoming by using Euclidean distances to learnable prototypes within representation space, leading to more fair decision boundaries.

The shortcoming of correlating softmax classifier norm to class size is directly connected to the classification model. We show an illustration in Fig. 3.1. Consider the problem of learning a classifier f_W parameterized by weight matrix W . We assume the feature backbone g_θ is fixed and known. The classifier outputs logit scores $f_W(g(x)) = W^T g(x)$, and the posterior probability $p(y|g(x)) \propto p(y)p(g(x)|y) \propto (W^T g(x))_y$. Note that softmax classifier has both a magnitude and direction associated to it. In the ideal case, the directions get aligned with the class means μ_y to encode $p(g(x)|y)$ [51, 52]. But for an imbalanced training dataset, to encode the prior of the label distribution $p(y)$ into classifier weights, the learned classifier norm gets correlated to the class size.

To circumvent this shortcoming of softmax in the long-tailed classification setting, it is advantageous to change the classification model from dot-product to distances in the representation space. More specifically, associate a learnable prototype c_y to each class y , and let $p(y|x) \propto -d(g(x), c_y)$, where $d(\cdot, \cdot)$ is a distance metric in the representation space. With a suitable choice of distance metric, such as Euclidean or cosine distances, we obtain a simple and effective prototype classifier which avoids the biased classifier norm problem. While prior work in long-tailed recognition explores the closely related Nearest-Class-Mean classifier [31], which is a special case of our model with fixed prototypes, the full potential of prototype based classifiers has not been fully explored. In this work we thoroughly analyze prototype classifiers and show that they offer a promising alternative to softmax in long-tailed recognition.

We theoretically analyze and show that the Euclidean distance based prototype classifier has stable gradient-based optimization properties for learning prototypes. In particular, the L2 norm of the gradient on a prototype c_y from a point x is always equal in magnitude to the misclassification probability. Further, the gradient is oriented along the line joining c_y to x . This leads to stable gradient updates that aren't affected by outliers that may be far from the prototypes.

We also allow for variable scales along the different dimensions of the feature representations in the prototype distance. This is enabled by learning channel dependent temperature parameters in the prototype distance function. We find that this is beneficial since it's a means of learnable feature scaling and/or selection in the prototype classifier. It also lets the model learn a general Mahalanobis distance metric parameterized by a diagonal matrix.

Overall, here is a summary of our main contributions:

(1) We propose a novel learnable prototype classifier for long-tailed recognition to counter the shortcoming of softmax to correlate classifier norm to class size. This of-

fers a promising distance-based alternative to dot-product classifiers that is relatively underexplored in the long-tailed setting.

(2) We theoretically analyze the properties of Euclidean distance based prototype classifiers that leads to stable gradient-based optimization which is robust to outliers.

(3) We enhance prototype classifiers with channel dependent temperature parameters in the prototype distance function, allowing for learnable feature scaling and/or selection.

(4) We evaluate our proposed prototype classifier on four benchmark datasets, CIFAR10-LT, CIFAR100-LT, ImageNet-LT and iNaturalist18. Our results show that the prototype classifier outperforms or is comparable to the recent state-of-the-art methods.

3.2 Related Work

Prototype-based classifiers. Prototype and Nearest-Class-Mean classifiers appear prominently in metric learning and few-shot learning. [53] proposed learning a Mahalanobis distance metric on top of fixed representations, which leads to a NCM classifier based on the Euclidean distance in the learned metric space. [54] learn deep representations for few shot learning by minimizing the prototype based loss on task-specific episodes. [55] learn deep representations by optimizing the NCM classification objective on the entire training data. [56] learn NCM classifiers by maintaining a fixed number of samples to compute prototypes or centroids. In this work, we neither learn a distance metric or finetune representations using the NCM objective; instead our Prototype classifier directly learns the prototypes from pre-trained representations for long-tailed recognition.

Long-tailed recognition. The literature on long-tailed recognition is quite vast. Data resampling or reweighting is the most commonplace strategy against imbalanced datasets.

This generally takes three forms: (i) Oversampling minority class samples by adding small perturbations to the data [57, 58], (ii) Undersampling majority class samples by throwing away some data [59], and (iii) Uniform or class-balanced sampling based on the number of samples in a class [60, 61]. However, oversampling minority class samples has been shown to lead to overfitting and undersampling majority class samples can cause poor generalization [41]. Recently, [31] learn the representations using instance-based sampling and retrain the softmax classifier in a second step using uniform sampling while keeping underlying representations fixed. We also use the two-stage training strategy in our work; however, we avoid learning an additional softmax classifier and instead use Prototype classifiers as it operates in the representation space directly. Another line of prior work focuses on engineering loss functions suited for imbalanced datasets. Focal loss [62] reshapes the standard cross-entropy loss to push more weight on misclassified and/or tail class samples. Class-balanced loss [30] uses a theoretical estimate of the volume occupied by a class to reweigh the loss function. [42] offset the label’s logit score with a power of class size to achieve a higher decision boundary margin for tail classes. [43] apply the offset to all the logits, and change the class size exponent from $1/4$ in prior work to 1 and use a meta-sampler to retrain the classifier. [44] apply a label-dependent adjustment to the logit score before softmax which is theoretically consistent. In our work, we build upon the logit adjustment approach and show how it can also be adapted to Prototype classifiers. Many other papers employ a specialized ensemble of experts to reduce tail bias and model variance. [63] train experts on class-balanced subsets of the training data and aggregate them using a joint confidence calibration layer. [46] also train experts on class-balanced subsets and distill them into a unified student model. [47] explicitly enforce diversity in experts and aggregate them using a routing layer. [48] train two experts using regular and reversed sampling together with an adaptive learning strategy. In other works, self-supervised contrastive pretraining

is used to improve feature representations for long-tailed recognition [45, 64]. More recently, [50] show that quite good performance on long-tailed datasets is possible by simply tuning weight decay properly. Thus, we build our Prototype classifiers on top of the representation learning backbone provided by them. However, the gain of Prototype classifiers over Softmax is invariant to the choice of the backbone, as we show in our experiments.

3.3 Our Approach

3.3.1 Prototype Classifier

The Prototype classifier assigns probability scores to classes based on distances to learnable prototypes in representation space. We assume that the representations $g(x)$ are fixed. In other words, the weights θ of the feature extractor g_θ are learned in a prior stage and fixed to enable learning of prototype classifiers. The decoupling of representation and classifier learning is commonly used to train softmax classifiers in long-tailed recognition [31]; here we adopt the same strategy to learn prototype classifiers.

Let the learnable prototype for class y be denoted by c_y . Under the assumption that the class-conditional distribution $p(g(x)|y)$ is a Gaussian with unit variance centered around the prototype c_y , we get the following probabilistic model for the class posterior $p(y|g(x))$:

$$\begin{aligned} \log p(y|g(x)) &\propto -\frac{1}{2}d(g(x), c_y) \\ &= \log \frac{e^{-\frac{1}{2}d(g(x), c_y)}}{\sum_{y'} e^{-\frac{1}{2}d(g(x), c_{y'})}} \end{aligned} \tag{3.1}$$

This leads to the nearest class prototype decision function for classification:

$$y^* = \operatorname{argmin}_{y \in \{1, \dots, \bar{y}\}} d(g(x), c_y)$$

Different choices of distance metric lead to different prototype classifiers. While squared Euclidean distance and Euclidean distance lead to equivalent decision boundaries, we find that Euclidean distance is more amenable for learning of the prototypes due to its stable gradient. The Prototype classifier with Euclidean distance metric specifically uses:

$$d(g(x), y) = \sqrt{(g(x) - c_y)^T (g(x) - c_y)} \quad (3.2)$$

The prototypes are learned using the Maximum Likelihood Estimation principle by minimizing the Negative Log Likelihood objective function:

$$L = -\frac{1}{N} \sum_i^N \log p(y_i | g(x_i)) \quad (3.3)$$

The remainder of this section is organized as follows: in Section 3.3.2, we show how Euclidean distance leads to stable gradient updates on prototypes, then in Section 3.3.3 we show how learning prototypes addresses the biased softmax problem, followed by Section 3.3.4 which details our channel-dependent temperatures to enhance prototype learning, and Section 3.3.5 which describes logit adjustment for learning prototypes. Our entire pipeline is presented in Algorithm 2.

3.3.2 Stable Gradient Updates on Prototypes

In Prototype classifier, we use the Euclidean distance in Eq. 3.2 due to its stable gradient optimization. We now describe the gradient updates on the prototypes.

Theorem 3.1 *Let L_{xy} denote the NLL loss on a single sample x with label y . Then, the L_2 norm of the gradient on prototype c_z is given by,*

$$\left\| \frac{\partial L_{xy}}{\partial c_z} \right\|_2 = \begin{cases} 1 - p(y|g(x)) & : z = y \\ p(z|g(x)) & : z \neq y \end{cases}$$

Further, if $z = y$, then the gradient descent direction lies along $g(x) - c_z$, else it is exactly the opposite direction.

Proof: Let $Z(x) = \sum_{y'} e^{-\frac{1}{2}d(g(x), c_{y'})}$ denote the normalization term over all the classes, and δ_{yz} denotes the Kronecker delta. Then, we get a closed form expression for the gradient using repeated application of the chain rule,

$$\begin{aligned} \frac{\partial L_{xy}}{\partial c_z} &= \frac{\partial(-\log \frac{e^{-\frac{1}{2}d(g(x), c_y)}}{Z(x)})}{\partial c_z} \\ &= \frac{\partial(-\log \frac{e^{-\frac{1}{2}\sqrt{c_y^T c_y - 2g(x)^T c_y + g(x)^T g(x)}}}{Z(x)})}{\partial c_z} \\ &= \frac{\delta_{yz}(1 - p(z|g(x)))(c_z - g(x))}{d(g(x), c_z)} \\ &\quad + \frac{(1 - \delta_{yz})p(z|g(x))(g(x) - c_z)}{d(g(x), c_z)} \end{aligned}$$

■

From the gradient update, we see that if sample x belongs to class y , then the updated prototype is shifted in the direction $(g(x) - c_z)$, weighted by the misclassification probability $(1 - p(z|x))$, otherwise it's shifted in the opposite direction weighted by the misclassification probability $p(z|x)$.

In both cases, the L_2 norm of the gradient is equal to the misclassification probability, leading to stable gradient updates. Importantly, it is invariant to $d(g(x), c_z)$, thus both inliers and outliers contribute equally. On the other hand, if we use squared Euclidean distance, it is easy to see that the L_2 norm of the gradient is scaled by $d(g(x), c_z)$, which is highly sensitive to outliers. We also verify empirically in Section 3.4.4 that Euclidean distance outperforms squared Euclidean distance for learning prototypes.

3.3.3 Addressing the Biased Softmax Problem

We show how learning prototypes addresses the biased softmax problem by drawing a connection with the softmax classifier. We consider below a Prototype classifier based on the squared Euclidean distance metric:

$$\begin{aligned}
-\frac{1}{2}d(g(x), c_y) &= -\frac{1}{2}(g(x) - c_y)^T(g(x) - c_y) \\
&= -\frac{1}{2}(c_y^T c_y - 2g(x)^T c_y + g(x)^T g(x))
\end{aligned}$$

Since the term $-\frac{1}{2}g(x)^T g(x)$ is shared by all classes, it can be ignored. Thus, the Prototype classifier with squared Euclidean distance metric is a softmax classifier with weight c_y and bias $-\frac{1}{2}c_y^T c_y$. Now, we know that softmax can increase or decreases the norm of its weight terms to account for an imbalanced prior label distribution $p(y)$ [31]. However, the same is not true for the Prototype classifier, because the bias term negates the gains from increasing or decreasing the norm of the weight term. Due to this coupling of the weight and bias terms, the Prototype classifier is more robust to imbalanced distributions and learns equinorm prototypes. We further show in our experiments in Section 3.4.5 that the learned prototypes are actually very similar in norm.

3.3.4 Channel-Dependent Temperatures

While computing distances for prototype classifiers, it is worthwhile to consider that all dimensions in the representation space may not have the same distance scale. Specifically, for a large distance along a dimension with high variance may not mean as much a large distance along a dimension with low variance. Thus, we define the channel-dependent temperature (CDT) prototype distance function using learnable temperatures T_i ,

$$d_{CDT}(g(x), y) = \sqrt{\sum_{i=1}^d \frac{(g(x) - c_y)_i^2}{T_i}} \quad (3.4)$$

A high value of temperature T implies low sensitivity to distances along that dimension, and a low value implies high sensitivity. Learnable temperatures allow the model to

learn feature scaling in the distance function. By letting the temperature approach very high values the model can also filter out potentially harmful features, i.e, feature selection. Finally, the CDT distance can also be interpreted as a Mahalanobis distance $\sqrt{(g(x) - c_y)^T \Sigma^{-1} (g(x) - c_y)}$, where Σ is a diagonal matrix with the temperatures on the diagonal.

3.3.5 Logit Adjustment for Learning Prototypes

Recent work by [44] suggests that, for a classifier trained on an imbalanced dataset, it is important to perform an additive adjustment to the logits based on the distribution prior. Motivated by this strategy, we modify the prototype classifier loss function in Eq. 3.3 as follows:

$$L_{xy} = -\log \frac{e^{-\frac{1}{2}d(g(x), c_y) + \tau \cdot \log N_y}}{\sum_{y'} e^{-\frac{1}{2}d(g(x), c_{y'}) + \tau \cdot \log N_{y'}}} \quad (3.5)$$

Here N_y denotes the number of training samples for class y and τ is a hyperparameter. Logit adjustment is only done during training and not during inference. It places a higher penalty on misclassifying tail class samples relative to head classes. Empirical results show that logit adjustment does yield better results for learning prototype classifiers.

3.4 Experiments

3.4.1 Datasets

We evaluate our proposed method on the following three benchmark long-tailed datasets:

1. **CIFAR100-LT** [42]: CIFAR100 consists of 60K images from 100 classes. Following prior work, we exponentially decay number of training samples and control the

Algorithm 1: Learning prototype classifier

Require: Pretrained feat reps $g(x)$, class means μ_y **Require:** Class-sizes N_y , logit-adj weight τ , lrs α, β

```

1 Initialize prototypes  $c_y = \mu_y$  and temperatures  $T_i = 1$ ;
2 for  $j \leftarrow 0$  to  $num\_batches$  do
3   Sample class-balanced batch  $(g(x_k), y_k)_{k=1}^{batchsize}$ ;
4    $\forall k \ \& \ \forall y'$  compute  $d_{CDT}(g(x_k), c_{y'})$  using Eq. 3.6 Compute  $L = \sum_k L_{x_k y_k}$ 
      using Eq. 3.7;
5    $\forall y'$  update  $c_{y'} \leftarrow c_{y'} - \alpha \nabla_{c_{y'}} L$ ;
6    $\forall T_i$  update  $T_i \leftarrow T_i - \beta \nabla_{T_i} L$ ;
7 end
```

degree of data imbalance with an imbalance factor β . $\beta = N_{max}/N_{min}$, where N_{max} and N_{min} are the maximum and minimum number of training images per class respectively. N_{max} is kept fixed at 500, and we report on three commonly used imbalance ratios $\beta \in [100, 50, 10]$. The validation set consists of 100 images per class and is also used as the test set.

2. **ImageNet-LT** [32]: This is a long-tailed split of ImageNet. ImageNet-LT has an imbalanced training set with 115,846 images for 1,000 classes from ImageNet-1K [65]. The class frequencies follow a natural power-law distribution [39] with a maximum of 1,280 and a minimum of 5 images per class. The validation and testing sets are balanced and contain 20 and 50 images per class respectively.
3. **iNaturalist-LT** [24]: This is a species classification dataset that naturally follows a long-tailed distribution. There are 8,142 classes, with a maximum of 118,800 images per class and a minimum of 16. The validation set contains 3 images per

class and is also used as the test set.

3.4.2 Evaluation Protocol

To distinguish the performance of the model on head vs tail classes, we report average top-1 accuracy on balanced test sets across four splits, *Many*: classes with ≥ 100 samples, *Med*: classes with $20 \sim 100$ samples, *Few*: classes < 20 samples, and *All* classes. Since the test sets are balanced, average accuracy and mean precision are the same.

3.4.3 Implementation Details

We train prototype classifiers on fixed representations from a prior pre-trained model. Our training pipeline consists of two stages, representation learning followed by prototype learning. To compare fairly with prior work, we use the same backbone architectures: ResNet-32 for CIFAR100, ResNeXt50 for ImageNet-LT and ResNet-50 for iNaturalist18.

Representation learning. We train all models end to end with an auxilliary softmax classifier by minimizing the average cross-entropy classification loss. At the end of training we discard the softmax classifier. We use instance-balanced data sampling [31] and regularize model weights using high weight decay [50], which achieves balanced weights in the feature extractor. We use the SGD optimizer with momentum 0.9, learning rate 0.01 along with cosine learning rate schedule [66] decaying to 0 for all models. For CIFAR100-LT and ImageNet-LT we train for 200 epochs with batch size 64 and for iNaturalist18 we train for 200 epochs with batch size 512. The weight decay parameter is found using Bayesian hyperparameter search [67]. We use random cropping and horizontal flipping for training data augmentation. All our training is done on NVIDIA V100 GPUs.

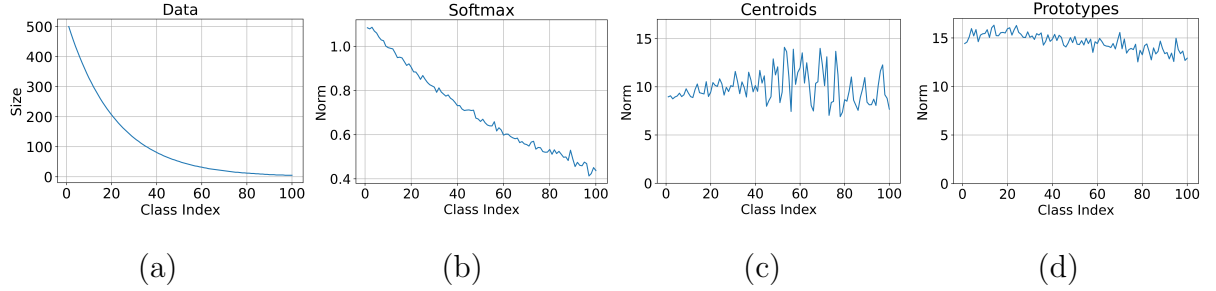


Figure 3.2: Comparison of softmax and prototype norms. L-R: a) Class size vs index, b),c) and d): Norm of softmax classifier, centroids and prototypes vs class index respectively on CIFAR100-LT. Classes are sorted according to decaying class size. Prototype classifier achieves balanced norms, in contrast to the biased softmax classifier and the imbalanced centroid norms.

Prototype learning. We freeze representations and compute empirical centroids to initialize the class prototypes. We minimize the average logit-adjusted cross entropy loss with logit-adjustment weight $\tau = 1/4$. We train for 1 epoch with class-balanced sampling, set prototype learning rate 4, CDT learning rate 0.005 and use the SGD optimizer with momentum 0.9. We also use random cropping, horizontal flipping, Autoaug [68] and Cutout [69] for data augmentation.

3.4.4 Ablation Study

Model components. The benefits gained by adding different components to our Prototype classifier (PC) are investigated in an ablation study over CIFAR100-LT with imbalance ratio 100, shown in Table 3.1. We make several observations: (i) NCM, which is a special case of our model where the empirical centroids are used as class prototypes, i.e. $c_y = \frac{1}{N_i}g(x)$, makes considerable gains above softmax, boosting *All* accuracy from 46.12 to 48.67 and *Few* accuracy from 11.34 to 14.13. NCM does not suffer from classifier norm

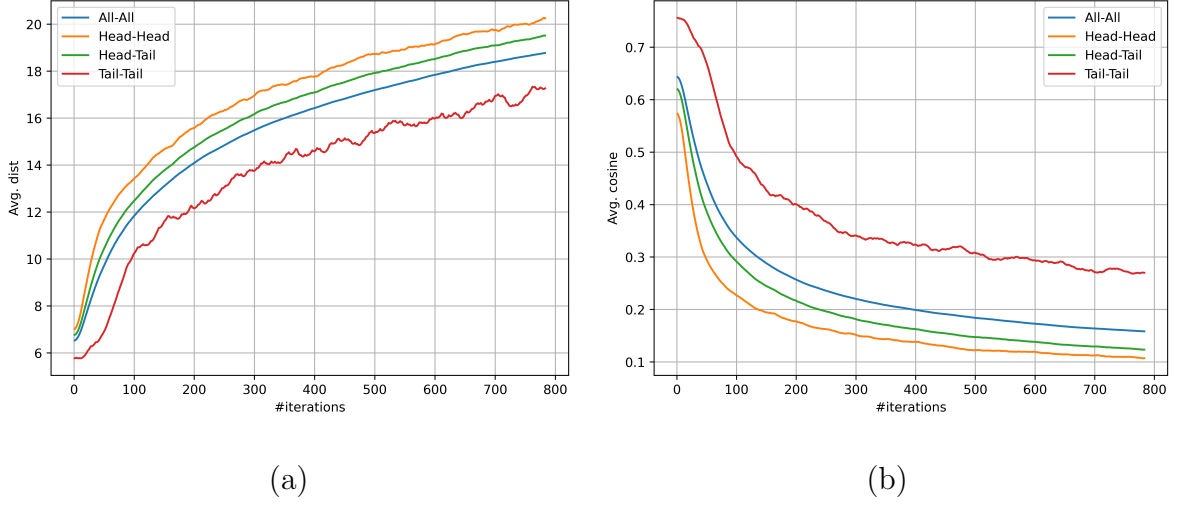


Figure 3.3: Evolution of prototypes during training. a) The average Euclidean distance between prototypes vs number of training iterations. b) The average cosine similarity between prototypes vs number of training iterations, a lower value implies higher angular separation. We compare across pairs of All-All, Head-Head, Head-Tail and Tail-Tail class prototypes. Euclidean and angular separation increase monotonically. Tail-Tail classes have smaller separation than Head-Head, which is due to the Minority Collapse effect in imbalanced training.

correlation to class size like softmax and can do better even without any learning. (ii) Prototype classifier (PC) lifts us above NCM significantly, achieving 4% improvement on *All* accuracy and 10% improvement on *Few* accuracy. Thus, learning prototypes gives a far superior nearest-prototype model than simply using the class-centroids as done by previous work. (iii) Adding channel dependent temperatures (CDT) to PC further improves the results by 0.5% on *All* accuracy and 4% on *Few* accuracy. CDT enhances the model by effecting distinct distance scales for all the features. (iv) Logit adjustment (LA) further improves results for PC. By combining both CDT and LA to PC we're able to gain 1% and 9% improvements over the baseline PC in *All* and *Few* accuracies

Method	Many	Med	Few	All
Softmax	76.12	46.13	11.34	46.12
NCM	76.54	50.40	14.13	48.67
PC	74.00	55.17	24.50	52.56
+ CDT	70.77	55.29	28.63	53.06
+ LA	69.29	54.63	32.67	53.17
+ CDT + LA	70.17	55.89	33.93	53.41

Table 3.1: Ablation study for CIFAR100-LT with imbalance ratio 100. NCM, PC, CDT and LA refer to Nearest-Class-Mean, Prototype classifier, Channel-dependent Temperature and Logit adjustment respectively. NCM alone with class centroids outperforms Softmax. PC via learning prototypes outperforms NCM by 4% on *All* accuracy. Adding CDT and LA further yields 1% improvement in *All* accuracy.

respectively. In the remainder, PC refers to our best model PC + CDT + LA.

Distance metrics. We experiment with different distance metrics for the Prototype classifier, particularly (a) Euclidean distance from Eq. 3.2, (b) Squared Euclidean distance and (c) Cosine distance: $d(g(x), c_y) = 1 - \frac{g(x)^T c_y}{\|g(x)\| \|c_y\|}$. The results on CIFAR100-LT are depicted in Table 3.2. We make some important observations: (i) Squared euclidean distance does considerably worse, with *All* accuracy dropping by more than 30%. This agrees with our theoretical analysis that the Squared euclidean distance is highly sensitive to outliers for learning prototypes. (ii) Cosine distance underperforms Euclidean distance on *All* and *Tail* accuracy. This suggests that the geometry of the representation space is closer to Euclidean and not hyperspherical.

Method	Many	Med	Few	All
Softmax	76.12	46.13	11.34	46.12
PC + Euclidean	70.17	55.89	33.93	53.41
PC + Squared Euclidean	39.89	21.77	8.10	24.01
PC + Cosine	72.94	54.80	24.77	52.14

Table 3.2: Effect of different distance metrics in learning prototypes on CIFAR100-LT. Euclidean distance outperforms Squared Euclidean and Cosine distance. Note the really bad performance of Squared Euclidean, which agrees with our theoretical analysis that its prone to outliers.

Feature backbones. Since Prototype classifier is agnostic to the feature backbone used to generate representations, we experiment with various representation learning and classifier learning schemes from the LTR literature. Particularly, we use CE, DRO-LT [45] and WD [50] as pre-trained feature backbones. For classifier learning, we use CE, cRT [31], WD + MaxNorm [50] and Prototype classifier (PC). The results on CIFAR100-LT are depicted in Table 3.3. We observe that Prototype classifier outperforms the competing classifier learning schemes for all the different feature backbones, thus showing that the performance gain is agnostic to the choice of backbone. Our best results are obtained using the WD baseline, which applies strong weight decay for regularizing feature weights.

3.4.5 Prototype Norms

To understand the advantage of prototype classifier over softmax, we plot the norm of softmax classifier, empirical centroids and learned prototypes In Fig. 3.2. We recover

	Classifier learning			
Representation Learning	CE	cRT	WD + MaxNorm	PC (Ours)
CE	38.3	41.3	43.4	44.1
DRO-LT	41.2	47.3	47.8	48.2
WD	46.1	50.1	53.3	53.4

Table 3.3: Effect of different representation and classifier learning schemes for CIFAR100-LT. PC outperforms competing classifier learning schemes for all different backbones.

the correlation of classifier norm to class size for softmax, as shown by prior work. For empirical centroids, it is interesting to see that this correlation does not hold; however the centroid norms are far from balanced. After learning the prototypes, we see that the norms become more balanced, with a slight favor towards the head classes. Since higher norm implies a negative bias in the prototype classifier, this actually favors the tail classes slightly.

3.4.6 How Do Prototypes Evolve During Training?

We are interested in the evolution of the class prototypes as training proceeds. Particularly, we seek to know the average separation in terms of Euclidean distance, as well as the average cosine similarity between the class prototypes. In Fig. 3.3, we plot the average of these values between pairs of All-All classes, Head-Head classes, Head=Tail classes and Tail-Tail classes. The average distance increases monotonically for all the pairs. The separation between Head-Head classes is greater than Tail-Tail classes. Similarly, the average cosine similarity decreases monotonically for all the pairs, therefore

the angular separation increases. There is higher angular separation between Head-head classes over Tail-Tail classes. Thus, the prototype classifier learns prototypes that are better separated than the empirical centroids, leading to lower classification error using the nearest-prototype decision function. It also mitigates Minority Collapse, wherein Minority Class means collapse towards the same point [70].

3.5 Comparison to State-of-the-Art

Compared methods. We compare against recently published and relevant methods in the LTR field. We choose representative models from the different solutions spaces- (i) data reweighting: Classifier retraining (cRT) [31] and DisAlign [71], (ii) loss reshaping: Focal loss [62], Label-distribution aware margin (LDAM) loss [42] and Logit-adjustment loss [44], (iii) Ensemble-based models: RIDE [47], (iv) Self-supervised contrastive learning: DRO-LT [45] and PaCO [64], and (v) Weight Regularization: WD, and WD + WD + Max [50].

Results. Our results on CIFAR100-LT, ImageNet-LT and iNaturalist18 are reported in Table 3.7, Table 3.8 and Table 3.9 respectively. The numbers of compared methods are taken from their respective papers. Prototype classifier outperforms the prior state-of-the-art methods DRO-LT, DisAlign etc. on all imbalance ratios in CIFAR100. We outperform WD + WD + Max, with whom we share the weight decay tuned baseline WD, on all three benchmark datasets. For ImageNet-LT, Prototype classifier outperforms all prior state-of-the-arts on *Tail* and *All* accuracy except models with “bells and whistles,” particularly RIDE, which learns and fuses multiple models into an ensemble. A similar conclusion can be derived on iNaturalist18, where our results rival prior state-of-the-arts, except the ensemble model RIDE, and PaCO, which uses self-supervised pre-training and

aggressive data augmentation techniques [72, 73]. However, since Prototype classifier is a classifier layer that can be applied to any feature backbone, our method is complimentary to ensembles and self-supervised pre-training. We leave this exploration for future work.

3.6 Takeaways

In this chapter, we studied Prototype classifiers to address the biased softmax problem for long-tailed recognition. Prototype classifiers do not suffer from the biased softmax problem of correlating classifier norm to class size. This is because they rely on distances to learnable prototypes in representation space instead of dot-product with learnable weights, which leads to a coupling of the weight and bias terms. Our theoretical analysis shows that Euclidean distance is stable for gradient-based optimization, and this is confirmed empirically. We further enable independent distance scales in the prototype distance function using channel-dependent temperatures. Our results on four benchmark long-tailed datasets show that Prototype classifier outperforms or is comparable to the prior state-of-the-art. Moreover, experiments reveal that the learned prototypes are equinorm and are much better separated in representation space than centroids.

3.7 Extended Empirical Analysis

3.7.1 Dataset statistics

We detail the dataset statistics for the three benchmark long tailed recognition datasets in Table 3.4.

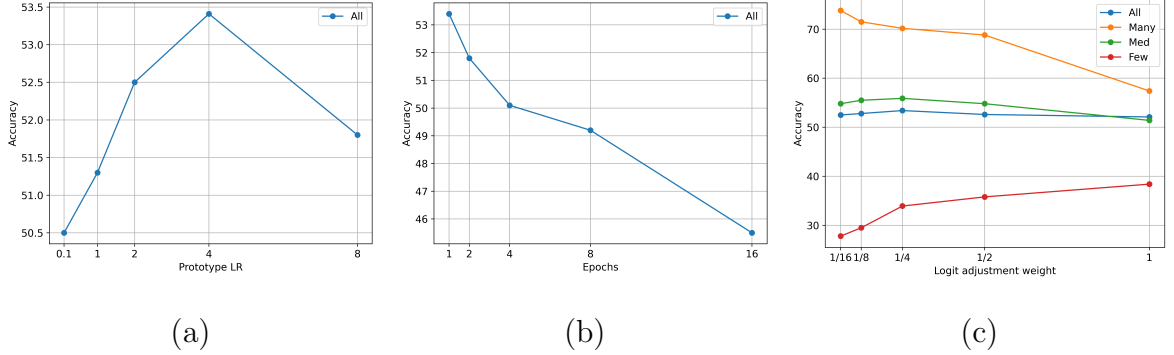


Figure 3.4: L-R: a) Effect of the prototype learning rate on CIFAR100-LT. b) Effect of the number of training epochs on CIFAR100-LT. c) Effect of the logit-adjustment weight τ on CIFAR100-LT.

3.7.2 More training details

We mention here a few more pertinent training details. The prototype learning is most optimal with high prototype learning rate. The effect of the prototype learning rate is shown in Fig. 3.4a. Further, we find that training for 1 epoch is enough. The effect of training epochs is shown in Fig. 3.4b. Training beyond 1 epoch leads to lower validation accuracy. Thus, the prototype learning has very fast convergence and is quick to train.

3.7.3 Temperature schemes

We discuss here some other temperature schemes for learning prototypes. Apart from Channel-dependent temperatures, we also experiment with Class-dependent temperatures. We vary the temperature according to class, allowing for different distance scales specific to the local neighborhood of the class. Specifically, we only allow a per-class

Dataset	Attribute	Many	Medium	Few	All
CIFAR10-LT (Imba 100)	Classes	8	2	0	10
	Samples	12273	133	0	12406
CIFAR100-LT (Imba 100)	Classes	35	35	30	100
	Samples	8824	1718	305	10847
CIFAR100-LT (Imba 50)	Classes	41	40	19	100
	Samples	10331	2008	269	12608
CIFAR100-LT (Imba 10)	Classes	70	30	0	100
	Samples	17743	2130	0	19573
ImageNet-LT	Classes	391	473	136	1,000
	Samples	89,293	24,910	1,643	115,846
iNaturalist18	Classes	842	4076	3599	8,142
	Samples	433,806	258,340	133,061	437,513

Table 3.4: Statistics for training data in CIFAR10-LT, CIFAR100-LT, ImageNet-LT and iNaturalist18.

temperature T_y :

$$d_{Class}(g(x), y) = \sqrt{\frac{\sum_{i=1}^d (g(x) - c_y)_i^2}{T_y}} \quad (3.6)$$

Secondly, we also experiment with Dense temperatures, allowing different temperature scales per-class as well as per-channel T_{yi} :

$$d_{Dense}(g(x), y) = \sqrt{\sum_{i=1}^d \frac{(g(x) - c_y)_i^2}{T_{yi}}} \quad (3.7)$$

The three schemes, Channel temps, Class temps and Dense temps are compared in Table 3.5. We find that both Class temps and Dense temps have similar results and underperform Channel temps. We hypothesize this is because Channel-temps encourage knowledge transfer from Many to Few shot classes by enabling globally shared feature selection.

Method	Many	Med	Few	All
PC	74.00	55.17	24.50	52.56
+ Channel T	70.77	55.29	28.63	53.06
+ Class T	72.33	54.83	25.62	52.47
+ Dense T	71.28	54.77	24.11	52.17

Table 3.5: Effect of different temperature schemes learning prototypes, i.e Channel-dependent Temperature, Class-Dependent and Dense temperatures respectively, on CIFAR100-LT. Channel-dependent does the best.

3.7.4 Effect of logit adjustment weight

In Fig. 3.4c, we study the effect of the logit adjustment weight on the different performance metrics. We observe a clear trade-off between *Few* accuracy on one hand and *All*, *Many* and *Med* on the other, with higher weight favoring *Few*. Consider that the logit adjustment $\log N_y$ leads to a higher loss on tail classes during the optimization process. Therefore, increasing the weight parameter leads to tail loss being over-emphasized during training and causing lower tail class error at the price of higher head class error. We choose $\tau = 1/4$ as the optimal weight achieving good accuracy across the spectrum.

3.7.5 Results on CIFAR10-LT

The results for CIFAR10-LT with imbalance ratio 100 are depicted in Table 3.6. The results are similar to CIFAR100-LT, we surpass the competing state-of-the-art methods cRT [31], LDAM [42], DRO-LT [45], and WD + WD + Max [50].

	All
CE	72.1
cRT	73.5
LDAM	77.03
DRO-LT	82.6
WD	78.6
WD + WD + Max	<u>82.9</u>
NCM	80.1
PC (Ours)	83.7

Table 3.6: Comparison to state-of-the-art on CIFAR10-LT with imbalance ratio 100. Prototype classifier achieves superior results.

Imbalance ratio	100	50	10
CE	38.32	43.85	55.71
Focal Loss	38.41	44.32	55.78
LDAM Loss	39.60	44.97	56.91
cRT	41.24	46.83	57.93
LogitAdjust	42.01	47.03	57.74
DRO-LT	47.31	57.57	63.41
PaCO	52.00	56.00	64.20
RIDE	49.10	-	-
WD	46.08	52.71	66.03
WD + WD + Max	<u>53.35</u>	<u>57.71</u>	<u>68.67</u>
NCM	48.67	53.62	67.82
PC (Ours)	53.41	57.75	69.12

Table 3.7: Comparison to state-of-the-art on CIFAR100-LT. Prototype classifier achieves superior results across all imbalance ratios.

	Many	Med	Few	All
CE	<u>65.9</u>	37.5	7.7	44.4
Focal Loss	36.41	29.9	16.0	30.5
cRT	61.8	46.2	27.3	49.6
DRO-LT	64.0	49.8	33.1	53.5
DisAlign	61.3	52.2	31.4	52.9
WD	68.5	42.4	14.2	48.6
WD + WD + Max	62.5	50.4	<u>41.5</u>	<u>53.9</u>
NCM	62.0	48.7	27.6	50.1
PC (Ours)	63.5	<u>50.8</u>	42.7	54.9
SOTA with “bells and whistles”				
RIDE	67.9	52.3	36.0	56.1
PaCO	63.2	51.6	39.2	54.4

Table 3.8: Comparison to state-of-the-art on ImageNet-LT. We outperform all prior state-of-the-art with single models and no self-supervised pretraining or aggressive data augmentation.

	Many	Med	Few	All
CE	<u>72.2</u>	63.0	57.2	61.7
Focal Loss	-	-	-	61.1
cRT	69.0	66.0	63.2	65.2
DRO-LT	-	-	-	69.7
DisAlign	69.0	71.1	70.2	70.6
WD	74.5	66.5	61.5	65.4
WD + WD + Max	71.2	70.4	<u>69.7</u>	<u>70.2</u>
NCM	61.0	63.5	63.3	63.1
PC (Ours)	71.6	<u>70.6</u>	70.2	70.6
SOTA with “bells and whistles”				
RIDE	66.5	72.1	71.5	71.3
PaCO	69.5	72.3	73.1	72.3

Table 3.9: Comparison to state-of-the-art on iNaturalist18. We outperform all prior state-of-the-art with single models and no self-supervised pretraining or aggressive data augmentation.

Chapter 4

Network Perturbation for Viral Marketing

4.1 Introduction

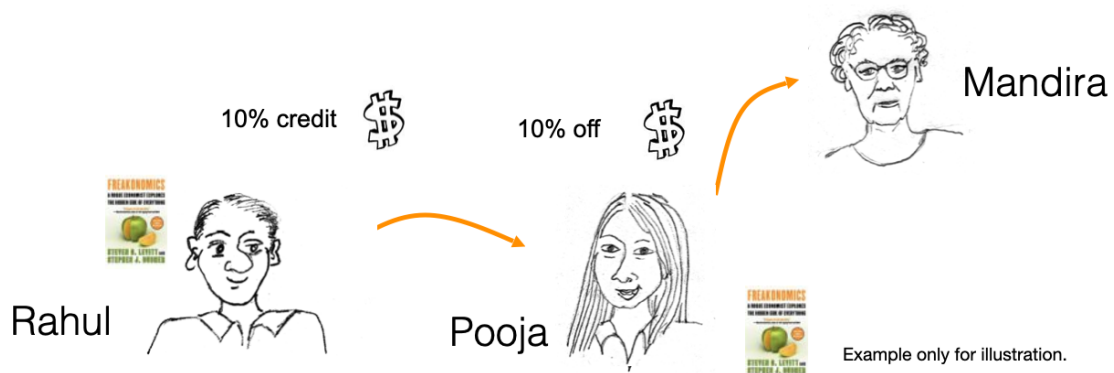


Figure 4.1: An illustration of viral marketing [20], where customers themselves carry out the promotional effort through referrals.

Viral marketing is a highly significant strategy used to maximize the adoption of products [74, 20], through diffusion in a social network of users [75]. This process is de-

picted in Fig. 4.1. Prior work is based mainly on postulated network propagation models of viral phenomena that focus on static topologies and ignore node attributes [76]. Furthermore, while finding the influential seed set has been extensively studied, the problem of making “dynamic” topological and attribute perturbations to maximize spread from adopters to non-adopters has not been addressed.

Instead of designed/postulated network propagation models, we adopt data-driven models, specifically non-linear Graph Neural Networks (GNNs) [16, 77] to learn a propagation model directly from attributed network data, and then use it to forecast future states of the spread after the network is perturbed. We train a GNN model on the initial state of the attributed network to learn a data-driven mapping from user attributes and neighborhood to its adoption label. Thereafter, the GNN parameters are fixed and the decision boundary of the GNN is used to identify adopters and non-adopters after the network is perturbed. This *self-labeling* technique allows us to study the effect of perturbations on user adoption by alleviating the issue of data scarcity regarding users with unseen combinations of attribute and neighborhoods.

In order to model the effect of perturbations, we propose a realistic model that can be used to strategically accelerate spread from adopters to non-adopters. The attributed networks we consider are unweighted, undirected graphs with binary node attributes and labels, where states of both attributes and labels correspond to the adoption of marketable products. Accordingly, at any given time, (a) new edges can be added only between adopters and non-adopters, as in referral marketing [78], and (b) adopters can further adopt similar products or products by flipping corresponding attributes from 0 to 1, as in joint or co-marketing [79].

The resulting Dynamic Viral Marketing (DVM) problem seeks to find the minimum budget and minimal dynamic perturbation set to attain a spread goal. We show that DVM is NP-Hard and relate it to the Influence Maximization (IM) problem [76]. Similar

to IM under the linear threshold model, nodes in DVM flip when the sum of incoming influence edge weights exceeds the node’s adoption threshold. Despite the similarity, the two problems are different as incoming influence edge weights and node thresholds in DVM are dynamic and governed by the underlying GNN propagation model.

Motivated by the connection of IM to DVM, we develop the Dynamic Gradient Influencing (DGI) framework to solve the DVM problem. DGI unrolls in discrete steps; each step involves flipping a non-adopter node that has the lowest budget and maximum downstream influence. We use gradient-guided node flipping to find the required dynamic perturbations. We develop an efficient node flipping budget computation approach using bisection search to maintain node budgets at each step. To estimate a node’s downstream influence, we develop the gradient based “Meta Influence” heuristic and the corresponding “Meta Attribute Flips” to increase the potency of edge perturbations.

Our contributions are as follows:

- We propose the novel Dynamic Viral Marketing (DVM) problem to find the minimum budget and a minimal dynamic perturbation set to attain a spread goal; a non-linear GNN acts as the propagation model and perturbations are restricted by referral and co-marketing constraints. We show DVM is NP-Hard and connect it to influence maximization.
- We develop the Dynamic Gradient Influencing (DGI) framework that targets low budget and high influence non-adopters, using (a) efficient budget computation, and (b) a novel Meta Influence heuristic with Meta Attribute Flips.
- We comprehensively evaluate DGI on 3 real-world attributed networks and show gains of 24% on budget and 37% on AUC over multiple baselines. Further, we extensively analyze cascade patterns created by DGI and intermediary nodes.

4.2 Related Work

Models for Network Diffusion The modeling of the diffusion of innovations in a network through the process of social contagion is a long studied topic [6]. [26] developed a threshold based model of collective behaviour where individuals are influenced by the proportion of others who come to a particular decision. [75] studied a coordination game of direct benefits from aligning choices with neighbors in a social network. In epidemiology, the spread of biological disease is studied using probabilistic transmission models of Susceptible, Infected and Recovered (SIR) individuals [4]. The use of the network value of customers for marketing was first explored in [74, 20]. Consequently, Influence Maximization (IM), the problem of finding the most influential seed set for viral marketing has been studied extensively [76, 14, 80]. Dynamic Viral Marketing lies within the broad class of spreading network processes and we show how it is connected to IM.

Graph Neural Networks Graph Neural Networks (GNN) are message passing neural networks that operate on attributed networks and have shown great success in problems such as node classification, link prediction, recommendation systems, and community detection. Various GNN architectures have been proposed since their first inception—Graph Convolutional Networks [16], GraphSAGE [35], Graph Attention Networks [17], Simplifying Graph Convolutional Networks [18]. We refer the reader to [81] for an extensive survey of graph neural networks. We use GNNs as the underlying propagation model for Dynamic Viral Marketing.

Gradient-based Network Optimization Gradient-based network optimization is used in many combinatorial network optimization problems. In the context of adversarial attacks on GNNs, perturbations are made using gradient optimization on the input network structure to reduce the accuracy of a GNN classifier [82, 83, 84]. Global attacks

with dynamic budget adjustment for Topology PGD [84] are considered by [85]. In [86], reinforcement learning policies are optimized to solve Maximum Coverage, Vertex Cover, and Influence Maximization problems on networks. In [87], graph neural networks are optimized to create graph embeddings and predict influence of nodes for solving Influence Maximization. We use gradient-based network optimization within the DGI framework.

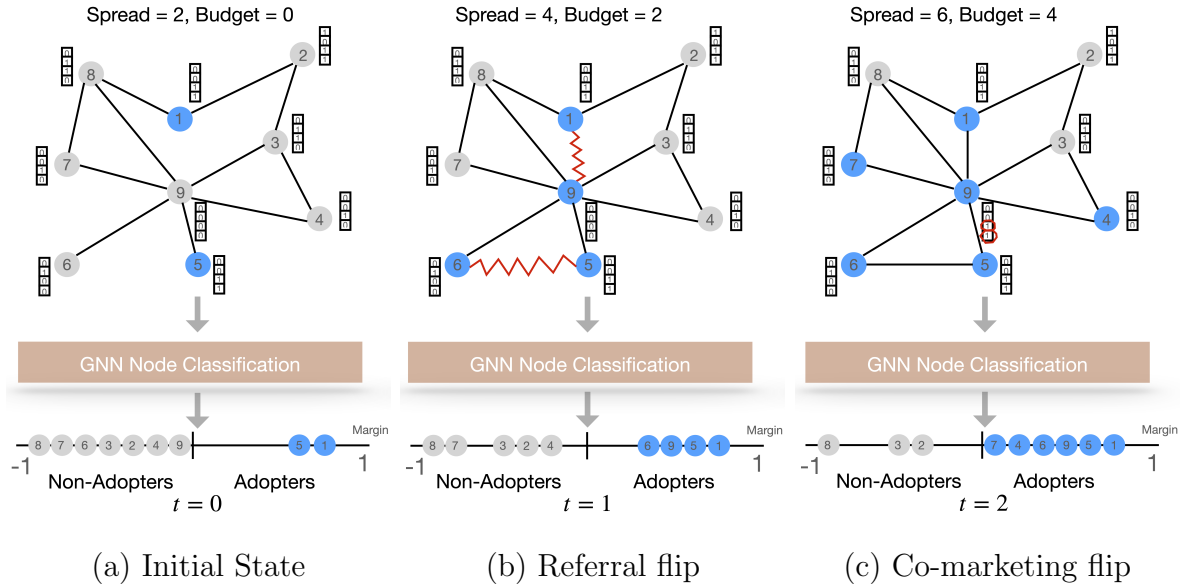


Figure 4.2: Overview of the Dynamic Viral Marketing (DVM) problem. At each time step, adopters and non-adopters in an attributed user network are specified using a GNN classifier. Referral and co-marketing perturbations accelerate spreading from adopters to non-adopters. DVM seeks to find the minimum budget and dynamic perturbation set to attain a spread goal.

4.3 Preliminaries

Consider a graph $G = (A, X)$, with the associated adjacency matrix $A \in \{0, 1\}^{n \times n}$ and node attribute matrix $X \in \{0, 1\}^{n \times d}$ respectively, and node labels $Y \in \{0, 1\}^n$.

We refer to the associated node-ids as $\mathcal{V} = \{1, \dots, n\}$. We denote the node feature $x_v \in \{0, 1\}^d$, and the node label $y_v \in \{0, 1\}$. The set of adopters and non-adopters is denoted as S and D respectively. For convenience, we denote the sub-matrix of A defining node connectivity from a set of nodes S to a set of nodes D as $A_{S,D}$, the sub-matrix of X containing features for a set of nodes S as X_S , and the vectors of ones and zeros as $\mathbf{1}$ and $\mathbf{0}$ respectively. We denote the weights on edge and feature perturbations as P_A and P_X respectively. The gradient scores on edge and feature perturbations are denoted by $\hat{P}_A$ and $\hat{P}_X$ respectively. For a dynamic network, the superscript t is used to indicate the variable at time t ; we drop the superscript if it's clear from the context.

We consider the large family of graph neural networks [16, 35] to construct layerwise hidden representations and finally output classifier logit scores $Z \in \mathbb{R}^{n \times 2}$. For an L-layer GNN,

$$H_l = \sigma(\hat{A}W_lH_{l-1}),$$

$$H_0 = X, \quad Z = H_L$$

where W_l refers to the learnable GNN parameters at layer l , σ is a nonlinear activation, and $\hat{A}$ is the GNN propagation matrix. For Graph Convolutional Networks (GCN) [16], $\hat{A} = \bar{\Delta}^{-\frac{1}{2}}\bar{A}\bar{\Delta}^{-\frac{1}{2}}$, where $\bar{A} = A + I$ and $\bar{\Delta}$ is the associated degree matrix. For GraphSAGE [35] with mean pooling, $\hat{A} = \bar{\Delta}^{-1}\bar{A}$. The predicted labels $y'_v \in \{0, 1\}$ for each node $v \in \mathcal{V}$ are given by the class with the maximum logit score. In typical semi-supervised learning for node classification, referred to as transductive learning, the GNN parameters W are learned by minimizing the cross-entropy classification loss,

$$\mathcal{L}_{tr}(v) = -\log \sigma(z_v)_{y_v} \quad (4.1)$$

where z_v denotes logit scores for node v and σ the softmax function.

4.4 Dynamic Viral Marketing Problem

In this section, we present the problem of viral marketing [20, 74] in the context of dynamic changes for accelerating the adoption of products by customers. Specifically, we consider a dynamic marketing scenario with the following salient properties:

1. *Referral marketing*: Companies incentivize people who are already using their product to refer it to others; adopters make new connections in the network to non-adopters.
2. *Co-marketing*: Companies partner with other companies or jointly market a group of products; people who adopt the target product likely adopt similar products and vice versa.

Formally, consider $G^t = (A^t, X^t, Y^t)$, a series of undirected, unweighted, and dynamic attributed networks, observed at time steps $t = 1, \dots, T$, where A^t represents the adjacency matrix defining node connectivity, X^t represents a binary node feature matrix containing adoption labels for related products, and Y^t represents the binary adoption labels for the target product. We assume that adoption labels are given by $Y^t = f(A^t, X^t, Y^0)$, where $f()$ is a general propagation model that governs diffusion of the initial labels Y^0 using network structure and attributes at time t . The sets of adopters and non-adopters at time t are denoted by S^t and D^t respectively,

$$S^t = \sum_v \mathbb{1}[y_v^t = 1], \quad D^t = \sum_v \mathbb{1}[y_v^t = 0] \quad (4.2)$$

The final spread, $\sigma()$ in the network at time T is given by,

$$\sigma(G^T) = |S^T|. \quad (4.3)$$

Due to the representation learning ability of graph neural networks (GNNs) [77, 16, 88, 89] through the propagation of feature and label information, we use them as our

propagation model $f()$. Specifically, a GNN f_θ is trained on the initial network G^0 and its parameters θ are then fixed. The GNN uses both structure and feature information to yield a decision boundary between adopters and non-adopters; the marketing objective is to flip nodes from non-adopters to adopters. Thereafter, we use *self-labeling*: the predictions from the GNN on G^t yield adoption states Y^t . Therefore, both our seed nodes, Y^0 , and our propagation model, $f()$, are data-driven.

The dynamic network transitions are constrained as follows:

- *Referral marketing*: Edge insertions can be made only between S^{t-1} and D^{t-1} at time $t - 1$; total cost of structural changes is $|A^t - A^{t-1}|$.
- *Co-marketing*: Features of nodes in S^{t-1} can flip from 0 to 1 at time $t - 1$; total cost of attribute changes is $|X^t - X^{t-1}|$.

As all the changes are made incrementally, the total cost incurred is given by $|A^T - A^0| + |X^T - X^0|$.

Dynamic Viral Marketing (DVM) : We now state the DVM optimization problem of finding the minimum budget, $\mu()$, and a minimal set of changes to reach a spread ϕ ,

$$\arg\min_{A^1, \dots, A^T, X^1, \dots, X^T} \sigma(G^T) \geq \phi, \quad (4.4)$$

$$\mu(\phi, G_0) = |A^T - A^0| + |X^T - X^0|. \quad (4.5)$$

Fig. 4.2 depicts the DVM problem schematically. Note that while the total budget is a function of the final adjacency and feature matrices, to solve the DVM problem a *sequence* of structural and attribute changes under the referral and co-marketing constraints are required. Furthermore, while we use a uniform cost model on edge and attribute perturbations, the problem can be extended to bespoke settings by using edge-specific and attribute-specific costs.

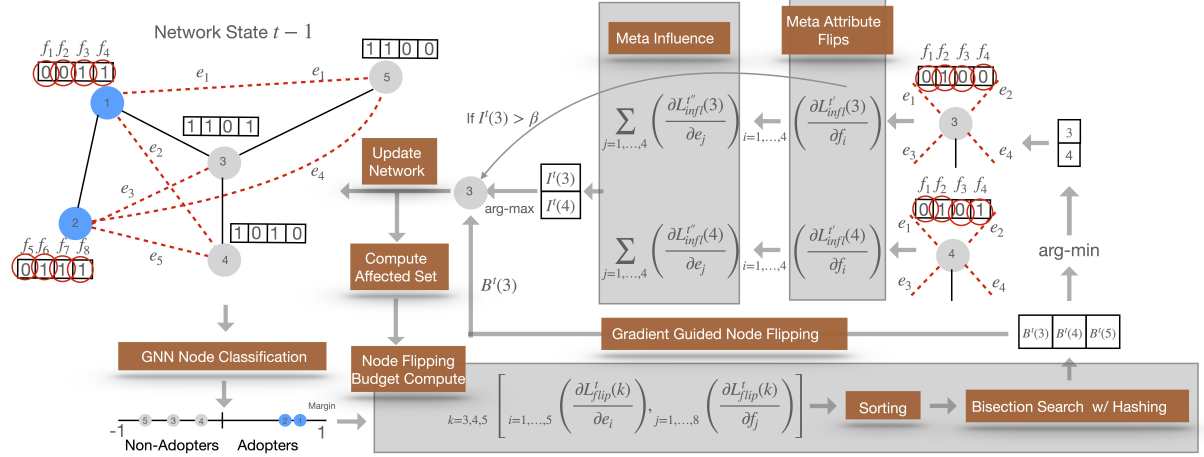


Figure 4.3: Overview of the Dynamic Gradient Influencing (DGI) framework. DGI picks candidate nodes to flip using Node Flipping Budget Compute, which involves gradient sorting along with bisection search, hashing and affected set estimation. The gradient-based Meta Influence Heuristic is used to tiebreak among least budget candidate nodes, as well as thresholding for Meta Attribute Flips that enhance node potency. Red lines and circles indicate candidate perturbations.

Theorem 4.1 *The decision version of the Dynamic Viral Marketing (DVM) problem is NP-Hard.*

Proof: Consider an instance of the NP-hard *Knapsack* problem, defined by a maximum value V , a maximum weight W , and a set of n items $X = \{(v_1, w_1), \dots, (v_n, w_n)\}$ where v_i and w_i denote the i^{th} item's value and weight respectively. The decision problem is whether there exists a subset of items $Z \subset X$ with total weight $\sum_{i \in Z} w_i \leq W$ and total value $\sum_{i \in Z} v_i \geq V$. We show that this problem reduces to the decision problem of DVM.

Given an arbitrary instance of the Knapsack problem, consider a star network with node t at the center which is connected to n nodes s_i and another n nodes t_i . Each s_i is in turn connected to two other nodes b_i and c_i . The initial labels on t, s_i, t_i are 0 and

that on b_i, c_i are 1. Each node has one feature that is initially $1 - w_i$ at b_i , $V - 1$ at c_i , v_i at s_i and V at t . The cost of changing the feature at a node b_i from a value of $1 - w_i$ to 1 is w_i . The feature values are held constant at all other nodes. The GNN classifier parameters are tuned such that the prediction on a node flips from 0 to 1 when the sum of its neighborhood features at nodes labeled 1 is at least V . The situation is depicted in Fig. 4.4.

Note that at a cost of w_i , the feature at node b_i can be set to 1. This causes the label at s_i to flip. If there is a subset Z of s_i nodes whose labels are 1 with $\sum_{i \in Z} v_i \geq V$ then the label on t will also flip to 1. This will then cascade to flips at all the n nodes t_i .

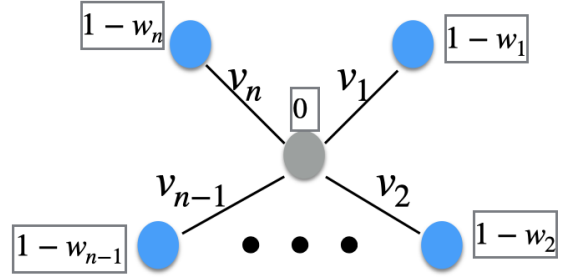


Figure 4.4: Star network used in the proof.

Thus, if there is a solution to the Knapsack problem then there is a spread of size $\geq n + 1$ with a cost $\leq W$. Similarly, if there is a spread of size $\geq n + 1$ with a cost $\leq W$ then node t must have flipped implying that there is a subset of s_i nodes with label 1 and whose features add up to at least V . This subset of s_i nodes must have flipped due to the features of the corresponding b_i nodes being incremented by w_i . The sum total of these increments is bounded by W , thus leading to a solution to the Knapsack problem. ■

4.4.1 Relating DVM to Influence Maximization

We draw an interesting connection between DVM and the related problem of influence maximization (IM) [76]. Consider the linear threshold propagation model [26], where nodes $i \in [n]$ randomly choose a threshold $\theta_i \in [0, 1]$ and incoming influence edge weights $I_{i,j}$ such that $\forall i, \sum_j I_{i,j} \leq 1$. The propagation unfolds in discrete time steps—if the set of

active nodes at any given step is S , then an inactive node becomes active if the following constraint is satisfied:

$$\sum_{j \in S} I_{i,j} \geq \theta_i. \quad (4.6)$$

While the objective in IM is to find a set of seed nodes for maximizing spread, we instead search for a sequence of dynamical changes to maximize spread. Despite the difference, given the set of adopters S^{t-1} at step $t-1$, the criterion for a node to flip in DVM has a similar form as Eq. 4.6. Suppose that the L -layer GNN f_θ has the associated L -step random walk propagation matrix M . Then the following theorem holds,

Theorem 4.2 *Let the vector $\varepsilon_j^t = x_j^t - x_j^{t-1}$ denote the change in the feature of node j from time $t-1$ to t . Further, let the matrix $\xi = M^t - M^{t-1}$ denote the change in the L -step random walk matrix M from time $t-1$ to t . Then the dynamic threshold and influence edge weights for node i to flip at time t according to the criterion in Eq. 4.6 are given by:*

$$\theta_i^t = z_{i,0}^{t-1} - z_{i,1}^{t-1} \quad (4.7)$$

$$I_{i,j}^t = M^t \alpha^T \varepsilon_j^t + \xi_{i,j}^t \alpha^T x_j^{t-1} \quad (4.8)$$

where α is a vector which depends on the parameters θ of the GNN.

Proof: Consider an L -layer GNN without non-linearities. Then the criterion to flip node i at time t can be expressed in terms of the L -step random walk matrix M and the weights W_l ,

$$\sum_{j \in N(i)} M_{ij}^t w_{L,1}^T \bar{W} x_j^t > \sum_{j \in N(i)} M_{ij}^t w_{L,0}^T \bar{W} x_j^t \quad (4.9)$$

where $w_{L,1}$ and $w_{L,0}$ represents the final layer classifier vectors for the two classes, and $\bar{W} = \prod_{l=1}^{L-1} W_l$ is the combined feature transform of the first $L-1$ layers.

The criterion can be equivalently written as,

$$\begin{aligned}
& \sum_{j \in N(i)} [M_{ij}^t w_{L,1}^T \bar{W}(x_j^t - x_j^{t-1}) + (M_{ij}^t - M_{ij}^{t-1}) w_{L,1}^T \bar{W} x_j^{t-1} \\
& \quad + M_{ij}^{t-1} w_{L,1}^T \bar{W} x_j^{t-1}] > \\
& \sum_{j \in N(i)} [M_{ij}^t w_{L,0}^T \bar{W}(x_j^t - x_j^{t-1}) + (M_{ij}^t - M_{ij}^{t-1}) w_{L,0}^T \bar{W} x_j^{t-1} \\
& \quad + M_{ij}^{t-1} w_{L,0}^T \bar{W} x_j^{t-1}] \quad (4.10)
\end{aligned}$$

Rearranging terms,

$$\begin{aligned}
& \sum_{j \in N(i)} [M_{ij}^t (w_{L,1}^T - w_{L,0}^T) \bar{W}(x_j^t - x_j^{t-1}) \\
& \quad + (M_{ij}^t - M_{ij}^{t-1}) (w_{L,1}^T - w_{L,0}^T) \bar{W} x_j^{t-1}] > \\
& \sum_{j \in N(i)} M_{ij}^{t-1} w_{L,0}^T \bar{W} x_j^{t-1} - \sum_{j \in N(i)} M_{ij}^{t-1} w_{L,1}^T \bar{W} x_j^{t-1} \quad (4.11)
\end{aligned}$$

Denoting $\alpha = \bar{W}^T(w_{L,1} - w_{L,0})$, and observing that the right hand side is the logit margin at time $t - 1$, the above simplifies to,

$$\sum_{j \in N(i)} [M_{ij}^t \alpha^T \epsilon_j^t + \xi_{i,j}^t \alpha^T x_j^{t-1}] > z_{i,0}^{t-1} - z_{i,1}^{t-1} \quad (4.12)$$

■

Intuitively, the dynamic node threshold depends on its logit margin, and the dynamic influence edge weights depend on both the feature change ϵ_j and the random walk propagation change $\xi_{i,j}$. Theorem 4.2 suggests that budget should be spent on changes which contribute most to the incoming influence weights and push the node just beyond its threshold.

4.5 Dynamic Gradient Influencing Framework

Motivated by the criterion for node flipping in Thm. 4.2, that the sum of the dynamic influence edge weights must exceed the dynamic node threshold, we develop the Dynamic Gradient Influencing (DGI) framework to solve the DVM problem. DGI uses Gradient-Guided Node Flipping (c.f. Sec. 4.5.1), to flip a particular candidate node in each step. At each step t , the candidate node to flip, v^t , is given by:

$$\arg\max_{v \in N} I^t(v), \text{ where } N = \arg\min_{w \in D} B^t(w) \quad (4.13)$$

where $B^t(w)$ denotes the budget required to flip node w , and $I^t(v)$ denotes the Meta Influence of node v . In other words, we choose the node with the least budget to flip and the highest Meta Influence. Using our novel Node Flipping Budget Computation algorithm (c.f. Sec. 4.5.2) candidate nodes are picked in order of lowest budget first. Further, we develop a novel Meta Influence heuristic (c.f. Sec. 4.5.3) for tiebreaking between equal budget candidates and thresholding for Meta Attribute Flips. We use Meta Attribute Flips to enhance a flipped node's downstream edge influence. Sec. 4.8.5 details DGI's asymptotic running time complexity. The complete DGI pipeline is depicted in Fig. 4.3, and the algorithm can be found in Alg. 3.

4.5.1 Gradient-Guided Node Flipping

In DGI, the core functionality for flipping nodes is accomplished through gradients on the restricted set of perturbations arising from the referral and co-marketing constraints of DVM. Specifically, the only changes that can happen to the adjacency matrix A and feature matrix X are restricted to the submatrices $A_{S,D}$ and X_S respectively. Therefore,

we define:

$$A_{S,D}^t = A_{S,D}^{t-1} + P_A^t \circ (\mathbf{1}\mathbf{1}^T - A_{S,D}^{t-1}), \quad A_{D,S}^t = (A_{S,D}^t)^T \quad (4.14)$$

$$X_S^t = X_S^{t-1} + P_X^t \circ (\mathbf{1}\mathbf{1}^T - X_S^{t-1}) \quad (4.15)$$

where P_A^t and P_X^t represent the weights on the edge and feature perturbations, $S = S^{t-1}$ and $D = D^{t-1}$, and $\circ$ denotes the Hadamard product. We initialize $P_A^t = \mathbf{0}\mathbf{0}^T$ and $P_X^t = \mathbf{0}\mathbf{0}^T$.

While it is an NP-Hard combinatorial optimization problem to find the minimal perturbation that flips a node, given that the adjacency and feature matrices are both discrete, first-order gradients work well enough in practice to find the required perturbations [82, 84]. We consider the negative cross-entropy loss as our flip loss for the chosen candidate node $v \in D$:

$$L_{flip}^t(v) = \log \sigma(z_v^{t-1})_0. \quad (4.16)$$

Note that $y_v = 0$ for non-adopters and $y_v = 1$ for adopters. We then compute non-negative gradient scores on edge perturbation weights, $\hat{P}_A^t$, and feature perturbation weights, $\hat{P}_X^t$, with respect to the flip loss:

$$\hat{P}_A^t = \max \left(\left[\frac{\partial L_{flip}^t(v)}{\partial P_A^t} \right], 0 \right), \quad \hat{P}_X^t = \max \left(\left[\frac{\partial L_{flip}^t(v)}{\partial P_X^t} \right], 0 \right). \quad (4.17)$$

We only compute the gradients once for all the perturbations. While it is possible to recompute gradients after every perturbation [82], we find that this is not that necessary to find the minimal set of perturbations. Moreover, as shown later, by merging and sorting the perturbations using the gradients, we can find the minimal perturbation set and minimum budget efficiently. Finally, suppose the minimum budget required to convert v is $B(v)$, then we find the top- $B(v)$ indices in the union of edge and feature

perturbations:

$$\hat{P}^t = \text{sort}(\text{merge}(\hat{P}_A^t, \hat{P}_X^t)) \quad (4.18)$$

$$i_A^t, i_X^t = \text{argtop-k}_{k=B(v)} \hat{P}. \quad (4.19)$$

Using the index sets i_A^t and i_X^t we can make the updates to the network to convert v :

$$[A_{S,D}^t]_{i_A^t} \leftarrow 1, \quad A_{D,S}^t \leftarrow (A_{S,D}^t)^T \quad (4.20)$$

$$[X_S^t]_{i_X^t} \leftarrow 1. \quad (4.21)$$

4.5.2 Node Flipping Budget Computation

The budget needed to flip a node v depends on both the logit margin in Eq. 4.7 and the node's degree $\deg(v)$ [90, 91]. In the adversarial attack framework, the practice is to set the node budget equal to its degree for local attacks [91], or choose a loss function which orders gradients in order of nodes closer to the decision boundary for global attacks [90]. However, due to the budget minimizing objective of DVM, we need to compute the budget precisely and pick candidate nodes that need the least budget.

Therefore, to compute the minimum budget $B^t(v)$ that converts node v , we use the bisection method [92, 93]. For each node, we compute the sorted gradients, $\hat{P}$ in Eq. 4.19 once and run bisection search over these gradients to find the minimal set of perturbations required to convert v . We initialize the lower and upper bound for search as 0 and $\deg(v)$, the degree of v , respectively. Thereafter, the upper bound is doubled until it is sufficient to convert v . We set the maximum upper bound equal to the maximum node degree of the network. After fixing the upper bound, bisection search repeatedly halves the search interval by checking feasibility of conversion at the midpoint of the interval and converges logarithmically.

We observe that other nodes also flip due to the same structure or attribute changes made for flipping a candidate node. Therefore, to choose the best candidate, we update

the budget:

$$B^t(v) \leftarrow B^t(v) + (|S^{t-1}| - |S_v^t|) \quad (4.22)$$

where S_v^t is the set of adopters at time t if node v is selected as the candidate node to flip. Thus, $B^t(v)$ represents both the node flipping budget and the "collateral damage" to other nodes from flipping it. Therefore, if a node with more budget causes high collateral damage it is preferable to a node with a lesser actual budget.

Since budgets need to be recomputed for all non-adopters at every step, we make the algorithm faster by hashing node budgets and only recomputing budgets for nodes whose budget has changed. The recompute set R^t is defined as the set of nodes whose logit scores changed in the previous time step:

$$R^t = \{v | v \in D^t, z_v^t \neq z_v^{t-1}\}. \quad (4.23)$$

For nodes whose logit scores are unchanged the actual budget might still change slightly, but for the purposes of picking the best candidate node we ignore these small changes. The entire algorithm for budget computation can be found in Alg. 2.

4.5.3 Meta Influence Using Meta Attribute Flips

Due to the dynamic sequence of changes involved in spreading product adoption in DVM, first-order gradients in Eq. 4.17 are insufficient to capture the long-range effects of a perturbation. While the flipping budget is minimized at each step, we need to characterize nodes that have high influence so that adoptions can cascade sequentially. Therefore, we develop the Meta Influence heuristic to model long-range effects and estimate downstream influence. Meta Influence uses Meta Attribute Flips which are feature perturbations that increase the potency of outgoing edge perturbations at an adopter node. Consequently, the Meta Influence is defined as the normalized gradient score on an adopter's outgoing edge perturbations post Meta Attribute Flips.

For Meta Attribute Flips, we restrict feature perturbations only to the features of node v and define perturbation weights P_X :

$$x_v^{t'} = x_v^t + P_X \circ (\mathbf{1} - x_v^t) \quad (4.24)$$

where t' indicates an auxilliary time step. We initialize $P_X = \mathbf{0}$. To capture the effect of Meta Attribute Flips, we consider an influence loss that is the sum on all non-adopters nodes of a CW-type loss [94] based on the logit margin, and compute gradient scores $\hat{P}_X$:

$$L_{infl}^{t'}(v) = \sum_{w \in D} (z_{w,1}^t - z_{w,0}^t) \quad (4.25)$$

$$\hat{P}_X^{t'} = \max \left(\left[\frac{\partial L_{infl}^{t'}(v)}{\partial P_X^{t'}} \right], 0 \right). \quad (4.26)$$

Thereafter, we update the features x_v using Meta Attribute Flips, which are the top- k ranked perturbations in $P_X^{t'}$, for the purposes of computing Meta Influence:

$$i_X^{t'} = \text{argtop-}k \hat{P}_X^{t'}, [x_v^{t'}]_{i_X^{t'}} \leftarrow 1 \quad (4.27)$$

where k is a hyper-parameter controlling the number of Meta Attribute Flips. From Thm. 4.2, Meta Attribute Flips find feature changes that align with the GNN’s classifier weights and increase the dynamic outgoing edge influence to other nodes in Eq. 4.8. Further, they also help to increase the margin from the GNN decision boundary and thus increase the node’s potency.

For finding the Meta Influence, we restrict the outgoing edge perturbations P_A from node v to the non-adopters D , and define corresponding edge perturbation weights P_A :

$$A_{v,D}^{t''} = A_{v,D}^{t'} + P_A^{t''} \circ (\mathbf{1} - A_{v,D}^{t'}) \quad (4.28)$$

where t'' indicates another auxilliary time step after t' and $P_A^{t''} = \mathbf{0}$. Now consider the same influence loss in Eq. 4.25 on all non-adopters *at time t''* , and compute non-negative

gradient scores $\hat{P}_A^{t''}$:

$$\hat{P}_A^{t''} = \max \left(\left[\frac{\partial L_{infl}^{t''}(v)}{\partial P_A^{t''}} \right], 0 \right). \quad (4.29)$$

Note that influence loss uses the discrete perturbed feature $x_v^{t'}$, which is computed using first order gradients, therefore Meta Influence can capture second order gradient effects. Finally, we denote the Meta Influence $I^t(v)$ of a node v as the normalized gradient score in $\hat{P}_A^{t''}$ averaged over the number of non-adopters:

$$I^t(v) = \frac{1^T \hat{P}_A^{t''}}{|D|}. \quad (4.30)$$

During candidate node selection, Meta Influence is used for tiebreaking between equal budget nodes (Eq. 4.13). Further, we threshold on Meta Influence to perform Meta Attribute Flips:

$$x_v^t \leftarrow \begin{cases} x_v^{t'} & \text{if } I^t(v) \geq \beta \\ x_v^t & \text{otherwise} \end{cases} \quad (4.31)$$

where β is a hyper-parameter controlling the threshold. Using Meta Influence, we can estimate which nodes will have high influence on their outgoing edges after Meta Attribute Flips, and thus judiciously allocate budget for Meta Attribute Flips. In Sec. 4.6.2, we validate that nodes with high Meta Influence indeed contribute more outgoing edge perturbations to non-adopters.

4.6 Experiments

We conducted experiments on real-world attributed network datasets to answer the following research questions:

- **RQ1:** How does DGI compare with baselines for DVM?

- **RQ2:** Does Meta Influence accelerate spread and capture a node’s actual dynamic influence?
- **RQ3:** What kinds of cascade patterns are created by DGI?
- **RQ4:** What are the properties of intermediary nodes in cascades created by DGI?

Datasets: We utilize three real-world attributed datasets to evaluate DGI and baseline approaches. Epinions and Ciao [95] are datasets collected from two popular product review sites, where each user can specify their trust relation in addition to rating products. Flixster [96] is a dataset collected from a popular movie rating website with an associated social graph. We create new small-scale and large-scale splits for these datasets using the provided user networks and product ratings from [95, 85]. For small-scale split generation, we sort the nodes using their degrees and take the subgraph corresponding to the lowest degree nodes. To generate features for each user, we pick a binary value for each product/movie based on whether they rated it or not. We choose the product/movie with the least number of seeds as the optimization goal for DVM. The analysis in the main paper is conducted on the small-scale splits and their statistics are depicted in Table 4.1. Additional details and results on large-scale splits using Fast-DGI can be found in Sec. 4.8.2.

Table 4.1: Dataset statistics. $|\mathcal{S}|$ denotes seed set size.

Dataset	$ \mathcal{V} $	$ \mathcal{E} $	Avg, Max Deg.	$ \mathcal{S} $	#Features
Flixster	1045	1488	2.8, 153	5	839
Epinions	1054	1214	2.3, 25	5	2999
Ciao	1057	1190	2.3, 36	7	2999

Evaluation metric: The efficacy of the proposed dynamic DGI and baselines is evaluated using the minimum budget needed to spread to $C = 500$ target nodes on the respective network. We use a fixed number of targets to spread to make the results across different datasets comparable. We also report the Area Under Curve (AUC) of the budget-spread curve, which gives an aggregate estimate of the budget required for different spread values. For calculating AUC, we normalize the budget by $(\sum(A)/2 + \sum(X))$, i.e., the sum of the number of edges and turned on features. Lower values of budget and AUC indicate better performance.

Variants: We consider three variants of DGI in our experiments:

- **Base** is DGI without Meta Attribute Flips. It uses Meta Influence only for tiebreaking in Eq. 4.13.
- **Fixed** is DGI with fixed Meta Attribute Flips. It is equivalent to using a threshold $\beta = 0$ in Eq. 4.7.
- **Dynamic** is DGI with optimally chosen threshold β .

Baselines: We compare with the following approaches:

- **Degree** selects target nodes in order of low-degree first. Until the target is flipped, Degree repeatedly spends a unit budget by first randomly picking a seeder node, then either adds a link from the seeder to the target if they aren't connected, else turns on a feature with high-correlation to the label.
- **Margin** selects target nodes based on the low-margin first heuristic, i.e., nodes that have a smaller margin to the decision boundary are picked first. Edits to the structure and features are made in the same way as Degree.

- **GradArgmax** [82] is a gradient based white-box adversarial attack on structure. Target nodes are selected in the order of lower losses first. We adapt GradArgmax to make edits to both structure and features.
- **MiBTack** [85] is another white-box adversarial attack that dynamically adjusts node budgets for topology-based PGD [84]. We adapt MiBTack to make edits to both structure and features while selecting target nodes same as GradArgmax.

4.6.1 Performance Comparison (RQ1)

We compare DGI variants to baselines using budget and AUC at $C = 500$ in Table 4.2 and Table 4.3 respectively. We report results with both GCN [16] and GraphSAGE [35] as the propagation models. DGI variants outperform the baselines in all the scenarios. Among the variants, Dynamic does the best, followed by Base and then Fixed. Fixed overspends budget on Meta Attribute Flips over Base by using a threshold of $\beta = 0$, and Dynamic spends the least budget by optimally selecting nodes with high Meta Influence (for spending additional budget) for Meta Attribute Flips. Further, we plot budget spread curves on Flixster and Epinions in Fig. 4.5. Dynamic DGI consistently achieves the minimum budget across all levels. To understand the time evolution of the spread for different variants, we plot spread as a function of time in Fig. 4.6a. Due to the accelerating effect of Meta Attribute Flips, Fixed spreads fastest, followed by the economical Dynamic and the conservative Base approach.

4.6.2 Strategy of Meta Influence (RQ2)

To validate the effectiveness of Meta Influence and Meta Attribute Flips, we plot the histogram of perturbations contributed by nodes with respect to their Meta Influence in Fig. 4.6b. For each dataset, we normalize the Meta Influence to lie in the interval $[0,1]$

Table 4.2: Comparison of DGI variants to baselines. Numbers indicate minimum budget to spread to 500 nodes. GCN and SAGE are used as the GNN propagation backbones. Dynamic DGI consistently achieves the minimum budget.

	Flixster		Epinions		Ciao	
	GCN	SAGE	GCN	SAGE	GCN	SAGE
Degree	2551	6573	22076	21125	25162	45291
Margin	8136	7655	26109	18550	20814	41077
GradArgmax	1012	625	5893	859	2620	972
MiBTack	843	583	2828	866	5111	1035
Base	791	543	3342	1099	3525	856
Fixed	831	571	1985	1297	2221	1094
Dynamic	667	494	1893	803	2096	821

Table 4.3: Comparison of DGI variants to baselines. Numbers indicate AUC of budget-spread curves. Dynamic DGI consistently achieves the minimum AUC.

	Flixster		Epinions		Ciao	
	GCN	SAGE	GCN	SAGE	GCN	SAGE
Degree	19.48	150.50	215.33	328.15	349.48	631.66
Margin	173.70	177.71	366.43	284.77	287.11	595.69
GradArgmax	23.32	13.72	112.84	16.36	54.46	18.52
MiBTack	21.78	16.77	50.70	21.03	80.95	21.91
Base	19.06	11.68	56.82	19.23	46.59	14.46
Fixed	20.78	12.53	36.41	23.75	36.20	19.82
Dynamic	18.27	10.17	31.93	15.18	33.84	13.33

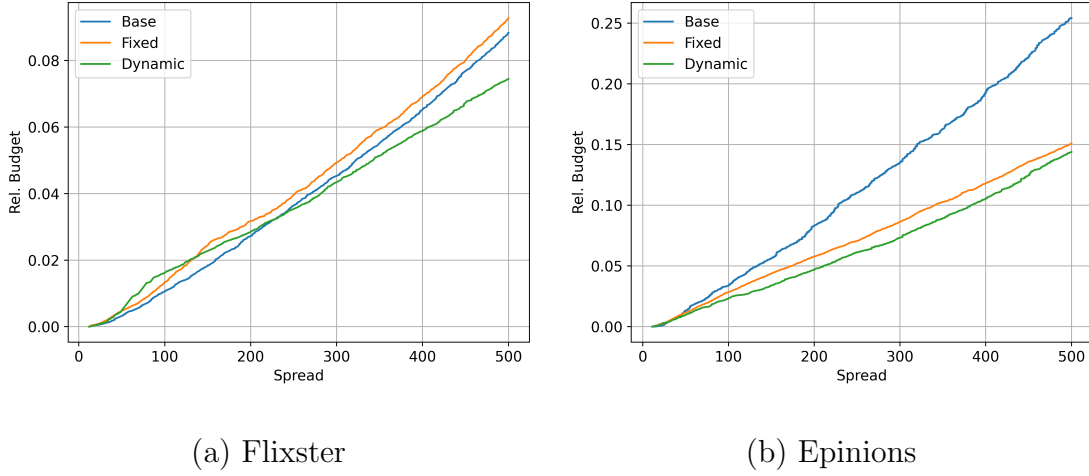
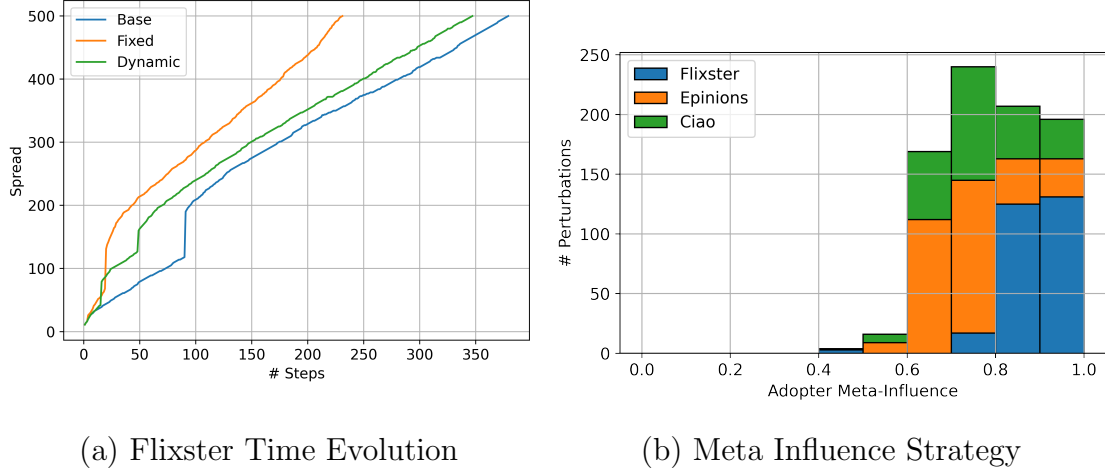


Figure 4.5: Budget spent as a function of increasing spread with GCN as the GNN propagation model. Dynamic requires consistently lower budgets across all spreads.

and divide uniformly into 10 sub-intervals. For each sub-interval, we count the number of perturbations contributed by nodes whose Meta Influence lies within it. We carry out the analysis on Fixed DGI, in which Meta Attribute Flips are made at each node. We observe a high correlation of Meta Influence to the number of contributed perturbations across all the datasets. Note that the Meta Influence is computed at the step when the node is flipped, but even then it provides a good signal of how important that node will be later. This shows that the Meta Influence is a close approximation of the actual node influence in terms of perturbations it makes. Further, by thresholding on the Meta Influence, we are able to save budget by not applying Meta Attribute Flips on low-influence nodes.

4.6.3 Cascades Created by DGI (RQ3)

To understand cascades created by the DGI spread, in Fig. 4.7a, we qualitatively visualize a subgraph spanned by the Dynamic DGI edges rooted at a single source node on Flixster, and color each node according to its cascade hop distance from the root. We



(a) Flixster Time Evolution

(b) Meta Influence Strategy

Figure 4.6: (a) Spread achieved with increasing time steps for Flixster with GCN backbone. Fixed and Dynamic spread faster than Base due to acceleration from Meta Attribute Flips. (b) Histogram of perturbations contributed by intermediary adopter nodes with increasing Meta Influence, where scaling is used for mapping Meta Influence to $[0,1]$. Higher Meta Influence adopter nodes contribute more perturbations.

define the cascade hop distance of a node from the seed set inductively:

$$hop(i) = 1 + \max_{j \in PN(i)} hop(j) \quad (4.32)$$

where $hop(i) = 0$ for nodes in the initial seed set, and $PN(i)$ denotes perturbed neighbors of node i at the step when it flips. Further, we use node size to indicate the number of perturbations the node contributes in the course of the multi-step spread. We clearly see a strong pattern of cascading flips, whereby a node flipped earlier later flips many more and so on inductively. Therefore, the DGI spread creates cascading flipping, similar to a chain of referrals in a social network, where each referral is an entirely new edge. We also see from the node sizes that a few nodes are dominant spreaders while others contribute very little.

To understand the cascades created by DGI quantitatively, in Fig. 4.7b we plot the

number of non-adopter flips with increasing hop distances for Flixster, Epinions and Ciao. Multi-hop cascade flips account for a sizable number of the total flips, which indicates that multi-hop path flips help in decreasing the budget required for the spread. Further, the cascade hop length can be considerably large. Particularly, for Flixster, we see nodes with cascade hop lengths up to 30, indicating how the added perturbations can percolate the adoption far from the seed set.

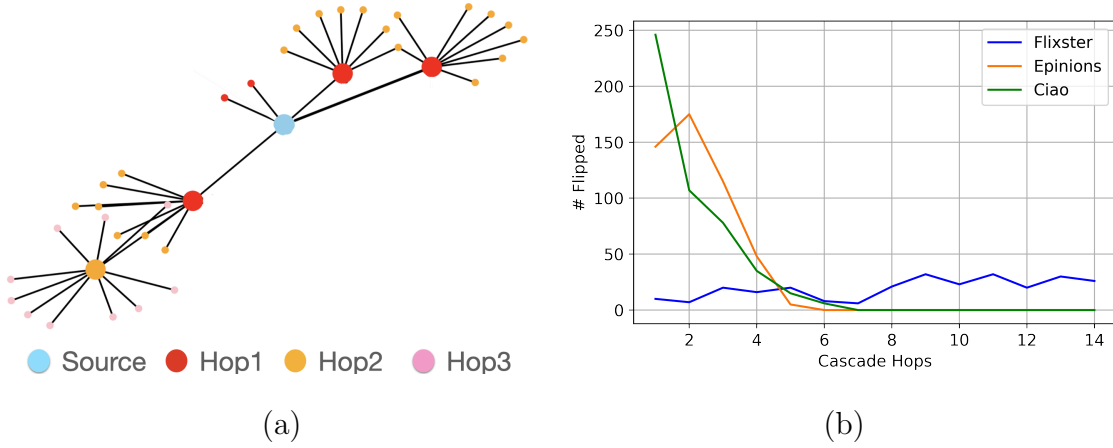


Figure 4.7: (a): Visualization of cascading dynamics of DGI. Node sizes and colors correspond to number of perturbations and cascade hops respectively. Only edges added by DGI are depicted. (b): Number of flipped nodes at different cascade hops. DGI creates long and staggered cascades for DVM.

4.6.4 Properties of Intermediary Spreaders (RQ4)

To understand the properties of intermediary spreader nodes, we plot the histogram of perturbations contributed by intermediary spreader nodes with respect to their degree and classification margin in Fig. 4.8a and Fig. 4.8b respectively. For each dataset, we count the number of perturbations arising from nodes with the degree and classification

margin lying within the same sub-interval. The degree and margin are considered at the moment the perturbation is made to account for dynamic changes. Due to the degree normalization in GNN message passing, low degree nodes have a higher influence edge weight, and we observe that nodes with low degree are highly correlated to higher number of perturbations. On the other hand, high classification margin indicates high feature and neighborhood alignment with the GNN classifier, therefore making outgoing edge or feature perturbations more potent. Thus, perturbations are made exclusively at nodes with the maximum possible margin.

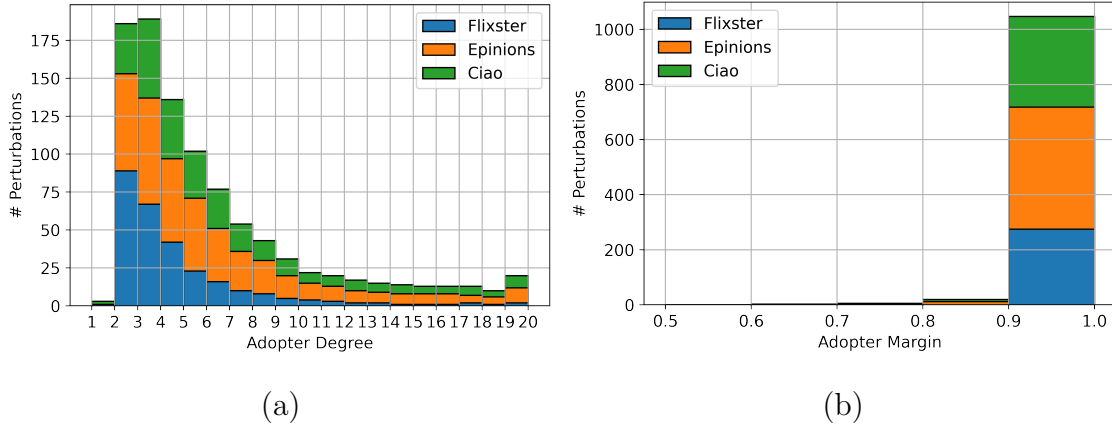


Figure 4.8: (a) Histogram of perturbations contributed by intermediary spreader nodes with increasing degree. (b) Histogram of perturbations contributed by intermediary spreader nodes with increasing GCN classification margin. Higher contributions are made by spreader nodes with low degrees and high margins.

4.7 Takeaways

In this chapter, we studied the novel Dynamic Viral Marketing (DVM) problem to find the minimum budget and minimal perturbation set to attain a spread goal, where the propagation model is a non-linear GNN and perturbations are restricted by refer-

ral and co-marketing constraints. We showed that DVM is NP-Hard and is related to influence maximization. We developed the Dynamic Gradient Influencing (DGI) framework, which targets non-adopters with low budget and high influence. DGI uses gradient ranking to order perturbations and utilizes an efficient budget computation approach, a novel Meta Influence heuristic, and Meta Attribute Flips to increase a node’s influence. We comprehensively evaluated DGI on three real world attributed networks and demonstrated gains on average of 24% on budget and 37% on AUC over multiple gradient and non-gradient baselines. We validated the efficacy of budget computation and the Meta Influence heuristic. We extensively analyzed the cascade patterns through intermediary adopter nodes discovered by DGI.

This work opens up a new research direction: data-driven models for network propagation as alternatives to postulated models such as Linear Threshold [26]. Data-driven models can incorporate attributes and long-range interactions. This research also motivates a number of future research questions, including model-based reinforcement learning [97] using such GNN models in unknown environments, and development of data-driven competitive strategies between groups, each trying to increase their spread.

4.8 Extended Analysis

4.8.1 Another Proof of NP-Hardness

For the proof in the main paper we assumed unweighted edges and weighted features, and directly relied upon the cascading flipping under the GNN propagation model to attain a desired spread. Below we provide an even simpler proof where we assume both weighted edges and weighted features.

Given an arbitrary instance of the Knapsack problem, consider a weighted star net-

work with target node t at the center which is connected to n source nodes. Each node has a single feature, whose value at node t is 0 and at node i is $x_i = 1 - w_i$. The weight on the edge between node i and t is set to $a_i = v_i$. The initial label on node t is 0 and the initial label on the other nodes is 1. The cost of changing the feature at a node from a value of $1 - w_i$ to 1 is w_i . The GNN classifier parameters are tuned such that the prediction on node t flips from 0 to 1 when the weighted sum of its neighborhood of n nodes with feature value $x_i = 1$ is at least V (spread $\phi = 1$). The Knapsack problem is solvable iff $\sum_{i \in [n]} a_i \mathbb{1}[x_i = 1] \geq V$ and $\sum_{i \in [n]} w_i \mathbb{1}[x_i = 1] \leq W$. Thus, if the DVM problem can find feature flips costing at most W at t 's neighbors whose edge weights sum to at least V then the Knapsack problem is solvable. $\square$

For the interested reader, we also point to prior work on certifiable adversarial robustness for neural networks with ReLU activation functions. The general problem of finding the minimum distortion of adversarial samples in NNs with ReLU activations is known to be an NP-Complete problem [98]. More generally, NNs are large, non-linear, and non-convex, and verifying even simple properties about them is an NP-complete problem, as can be shown using a reduction from the 3-SAT problem [99].

4.8.2 Implementation Details

For the propagation model, we use 2-layer GNN architectures, as stacking multiple layers can lead to oversmoothing in GNNs [100]. We report results with both GCN [16] and GraphSAGE [35] as the backbone propagation models. We set the hidden layer to size 64 and use ReLU as the intermediate non-linear function. We train models using cross entropy loss for 200 epochs, a patience of 50, learning rate 1e-2 with cosine annealing and weight decay regularization 5e-4. We use all nodes and edges during training to attain the best GNN decision boundary. All models achieve 100% accuracy on the seed

Algorithm 2: Bisection search to find minimum budget.

Input: $G = (A, X)$, GNN f_θ , $\hat{P}$, Target node v .

Output: $B(v)$, minimum budget to flip v .

```

1 Init bounds  $U = \deg(v)$  and  $L = 0$ 
2 do
3    $U = 2U$ 
4 while  $v$  is not flipped by top- $U$  perturbations in  $\hat{P}$ ;
5 do
6    $C = \lfloor \frac{U+L}{2} \rfloor$ 
7   if  $v$  is not flipped by top- $C$  perturbations in  $\hat{P}$  then
8      $L = C$ 
9   else
10     $U = C$ 
11  end
12 while  $U - L > 1$ ;
return:  $U$ 

```

Algorithm 3: DGI

Input: $G=(A,X)$, GNN f_θ , Adopters S , Non-Adopters D , k, β **Output:** Minimum budget required s.t. $|S| = 500$

```

1 Init budget  $\mathcal{B} = 0$ 
2 do
3   for  $v \in D$  do
4     Compute  $\hat{P}$  in Eq. 4.19
5     Call Algorithm 2 to compute  $B(v)$ 
6   end
7   Find  $N$ , the set of minimum budget nodes
8   for  $v \in N$  do
9     Compute the Meta Influence  $I(v)$ 
10  end
11   $v^* \leftarrow \max_{v \in N} I(v)$ 
12  Perturb the network to flip  $v^*$ 
13  if  $I(v^*) > \beta$  then
14    Make  $k$  Meta Attribute Flips at  $v^*$ 
15     $B(v^*) \leftarrow B(v^*) + k$ 
16  end
17   $\mathcal{B} \leftarrow \mathcal{B} + B(v^*)$ 
18   $S, D \leftarrow \{v | y'_v = 1\}, \{v | y'_v = 0\}$ 
19 while  $|S| < 500$ ;

return:  $\mathcal{B}$ 

```

set along with a small number of false positives that are in the vicinity of the seeds and are included in the initial seed set. For the spread approaches, the hyperparameter k for number of Meta Attribute Flips is set to a value in $[1, 2, 4, 8, 16]$ using grid search with Fixed DGI. The threshold hyperparameter β is set to a value within $[0, 1]$ of the maximum Meta Influence in Fixed DGI using grid search with Dynamic DGI. The maximum upper bound to convert a node is set to the maximum degree in the graph during bisection search. If the spread approach is unable to increase the size of the adopters for 40 steps, we halt and report failure. We use spread approaches with GCN backbones for all our analysis, and note that the SAGE backbone yields the same insights. DGI and baselines are implemented using the DeepRobust library in Pytorch [101] for adversarial attacks on GNNs. All GNN models and spread approaches are trained and executed on a single NVIDIA RTX2080 GPU with 8GM RAM.

4.8.3 Results on Large-Scale Data

The statistics for the larger scale datasets are presented in Table 4.4, and the results are presented in Table 4.5 and Table 4.6 respectively. It is interesting to note the distinction of the larger scale datasets from the smaller scale datasets in that the former are more tree like and the latter are more dense. This causes the propagation to happen much faster in the dense networks and with lesser budgets, therefore it is an easier problem when the attributed network is larger and more dense. This also matches our intuition that densely connected networks allow for faster information spread. This suggests that sparser networks require more effort for a network process such as DVM to achieve an equivalent amount of spread.

Table 4.4: Dataset statistics. $|\mathcal{S}|$ denotes seed set size.

Dataset	$ \mathcal{V} $	$ \mathcal{E} $	Avg, Max Deg.	$ \mathcal{S} $	Feats
Flixster	3000	29677	20, 1716	10	839
Epinions	15948	234438	29, 1443	11	2999
Ciao	6841	77404	22, 749	6	2999

Table 4.5: Comparison of DGI variants to baselines. Numbers indicate minimum budget to spread to 500 nodes. GCN and SAGE are used as the GNN propagation backbones.

	Flixster		Epinions		Ciao	
	GCN	SAGE	GCN	SAGE	GCN	SAGE
GradArgmax	459	241	489	425	1097	630
MiBTack	399	310	376	383	869	705
Base	466	92	332	318	1136	615
Fixed	222	106	623	615	737	576
Dynamic	195	87	287	291	501	509

4.8.4 Ablation Study

We study the effect of the different components of Dynamic DGI, i.e, budget compute (BC), tiebreaking (TB) and Meta Attribute Flips (MAF), in Table 4.7 and Table 4.8 respectively. We see that the different components indeed cause an additive increase in the performance of Dynamic DGI in all the scenarios, thus validating their utility.

Table 4.6: Comparison of DGI variants to baselines. Numbers indicate AUC of budget-spread curves.

	Flixster		Epinions		Ciao	
	GCN	SAGE	GCN	SAGE	GCN	SAGE
GradArgmax	2.36	1.23	0.27	0.24	1.18	0.82
MiBTack	1.87	1.45	0.16	0.17	0.98	0.90
Base	2.38	0.57	0.13	0.14	1.20	0.81
Fixed	1.17	0.61	0.29	0.28	0.85	0.77
Dynamic	1.08	0.48	0.11	0.12	0.71	0.70

Table 4.7: Ablation study for the effect of budget compute (BC), tiebreaking (TB) using Meta Influence, and Meta Attribute Flips (MAF).

BC	TB	MAF	Flixster		Epinions		Ciao	
			GCN	SAGE	GCN	SAGE	GCN	SAGE
			1012	625	5893	859	2620	972
✓			797	506	3235	831	4231	878
✓	✓		791	543	3342	1099	3525	856
✓	✓	✓	667	494	1893	803	2096	821

4.8.5 Complexity

In each step, for each target, DGI makes 1 backward pass to compute gradients and then sorts the gradients in $O(E' \log E')$ time, where E' is the number of non-edges in the graph. This is followed by $O(\log \Delta)$ forward passes in bisection search, where Δ is the maximum allowed budget, which we set as the maximum degree of the graph. For the set of minimum budget nodes, the Meta Influence makes two forward and two backward

Table 4.8: Ablation study for the effect of budget compute (BC), tiebreaking (TB) using Meta Influence, and Meta Attribute Flips (MAF).

BC	TB	MAF	Flixster		Epinions		Ciao	
			GCN	SAGE	GCN	SAGE	GCN	SAGE
			23.32	13.72	112.84	16.36	54.46	18.52
✓			19.44	10.54	54.01	15.44	66.02	14.66
✓	✓		19.06	11.68	56.82	19.23	46.59	14.46
✓	✓	✓	18.27	10.17	31.93	15.18	33.84	13.33

passes. The time taken for a forward or backward pass in a 2-layer GNN on a GPU is $\mathcal{O}(1)$ assuming small input, hidden, and output sizes. Thus, in the first step of DGI, we make $O(|D|)$ inner steps of gradient-guided node flipping, budget computation, meta attribute flips and meta influence, which takes a total of $O(|D|(E' \log E' + \log \Delta))$ time. In later steps, due to budget hashing, we only recompute budgets for $O(1)$ nodes, which takes $O(E' \log E' + \log \Delta)$. Assuming that every step of DGI flips $O(1)$ non-adopter nodes, the total runtime complexity of DGI to achieve a spread of ϕ can be determined as:

$$O(\phi(E' \log E' + \log \Delta)) + O(|D|(E' \log E' + \log \Delta)). \quad (4.33)$$

Since real world networks are mostly sparse, E' is usually quadratic and DGI is difficult to scale up for larger networks. Therefore, we also use a faster variant of DGI, Fast DGI, for large scale network splits, where we recompute budgets after every $\frac{\phi}{10}$ steps, therefore bringing down the time complexity to:

$$O(E' \log E' + \log \Delta) + O(|D|(E' \log E' + \log \Delta)). \quad (4.34)$$

4.8.6 Sensitivity Analysis

Table 4.9 reports the hyperparameters used in this work. The sensitivity of Dynamic DGI to the hyperparameter β for threshold is presented in Fig. 4.9a and Fig. 4.9b respectively. Likewise, sensitivity of Dynamic DGI to hyperparameter k for number of Meta Attribute Flips is presented in Fig. 4.10a and Fig. 4.10b respectively. We see that larger values of β and smaller values of k generally work better.

Table 4.9: Hyperparameter values used for Dynamic DGI on various datasets and GNN backbones. k denotes number of Meta Attribute Flips and β (quantile) denotes the quantile of the threshold β applied to the Meta Influence.

	Flixster		Epinions		Ciao	
	GCN	SAGE	GCN	SAGE	GCN	SAGE
k	2	2	4	2	4	1
β (quantile)	0.75	0.94	0.65	0.98	0.30	0.90

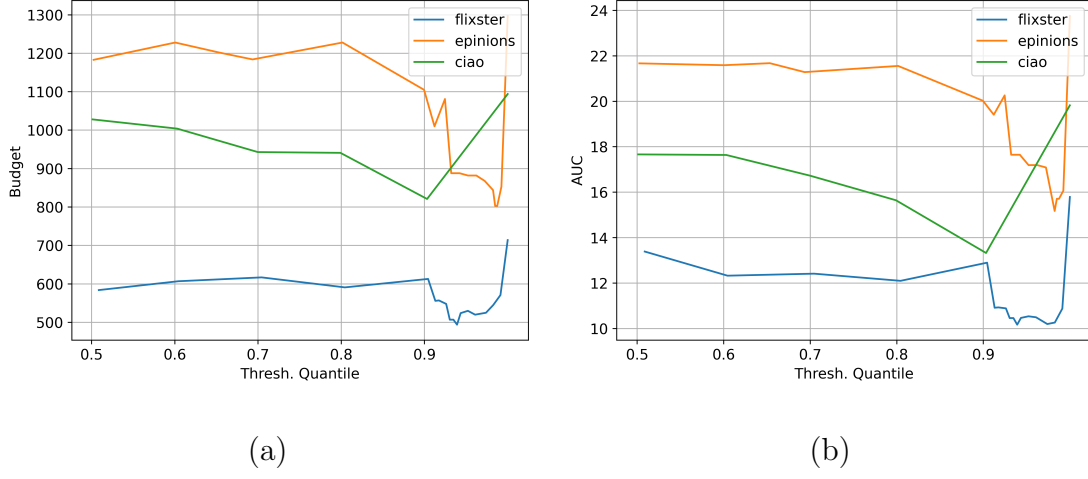


Figure 4.9: Sensitivity of Dynamic DGI to the influence threshold parameter β . The x-axis represents the quantile of the threshold β applied to the Meta Influence for each dataset. (a) represents budget and (b) represents AUC of budget-spread curve for a spread of 500.

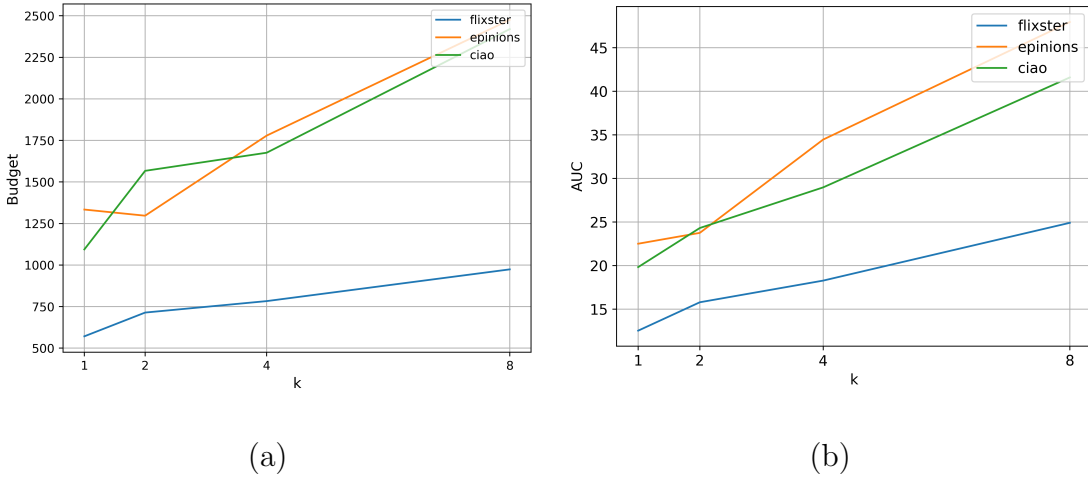


Figure 4.10: Sensitivity of Dynamic DGI to the parameter k controlling the number of Meta Attribute Flips. (a) represents budget and (b) represents AUC of budget-spread curve for a spread of 500.

4.8.7 Cascades in Adversarial Perturbations

In this section, we analyze cascade patterns in injected adversarial perturbations. Adversarial attacks on graph neural networks have recently been studied due to the increasing use of graph neural networks for problems such as recommendation systems, fraud detection, molecular property prediction, and social network analysis. Through access to model parameters (white-box attacks) and even with access only to model predictions (black-box attacks), GNN models are known to suffer a severe degradation in performance from a few perturbations to the graph adjacency and node attributes. While numerous adversarial attacks and defenses have been proposed in the literature, there is still a lack of understanding about the kinds of noise injected into the graph by these adversarial attacks. In the real world, networked systems are known to be vulnerable to cascading reactions from a few perturbations, such as in the context of epidemics, information spread and marketing. Therefore, the perturbations that're introduced by attacks induce a similar effect to infection spread in the real world.

More formally, we describe the kinds of cascade patterns that are created by various adversarial attacks on different kinds of graph neural network architectures. In a typical adversarial attack (for instance, Nettack, Fast Gradient Attack, Projected Gradient Descent, Mettack), perturbations are added sequentially into the network in order of their importance. Therefore, this describes a similar procedure to recommendations within a social network where users recommend things to each other and that're consequently spread through the network. We can now map these kinds of cascade patterns to the adversarial noise, by understanding the cascade patterns. Fig. 4.11 demonstrates the kinds of cascade patterns created from Mettack. Further, we visualize the most frequent cascade patterns among these using approximate graph isomorphism in Fig. 4.12.

Our finding that injected perturbations form cascade subgraphs suggests that adver-

sarial attacks exploit synergy between local interactions and therefore the perturbations work in a concerted way. Therefore, this opens up directions for future research into defenses against adversarial attacks by preventing the triggering of cascading perturbations.

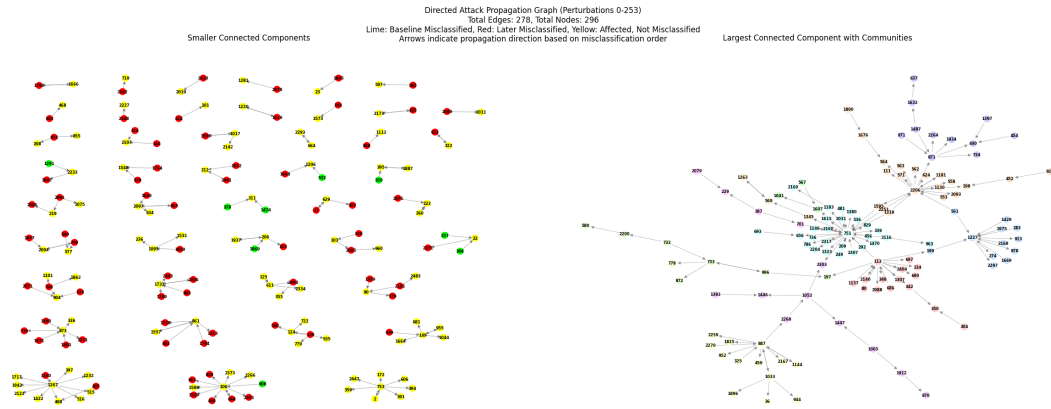


Figure 4.11: Visualization of adversarial attack cascades demonstrates most of the injected perturbations falls into a large component and small scattered trees. The perturbations are obtained from Mettack [102] on a GCN trained on the Cora network.

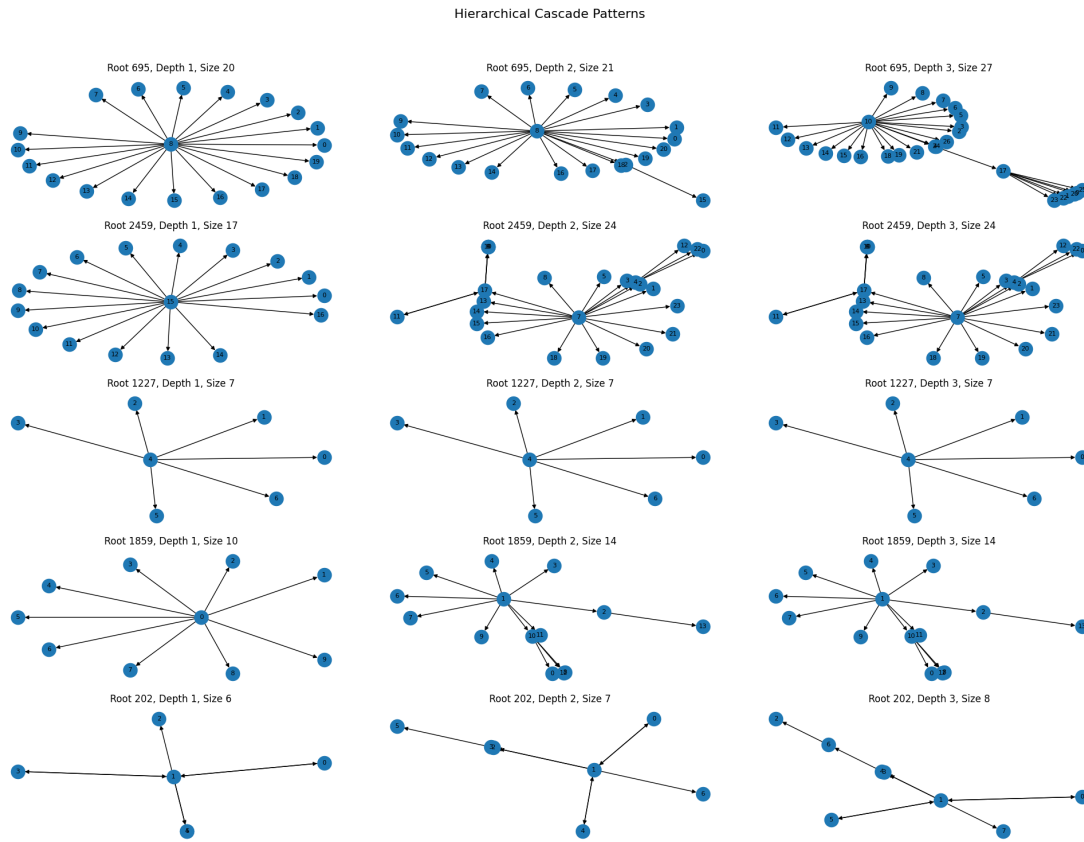


Figure 4.12: Visualization of most frequent cascade patterns obtained using approximated graph isomorphism on the cascade graphs.

Chapter 5

Unified Propagation Model for Complex Contagion

5.1 Introduction

Complex contagion [10, 103, 104, 11] is a popular model for the spread of behaviours in a networked system where nodes adopt the propagating behaviour if and only if the cumulative local influence exceeds a threshold. In contrast to phenomena such as information and disease spread, the spreading of behaviours is slower and requires critical mass to gather momentum. However, most prior models of complex contagion rely solely on the number of infected neighbors as a measure of local influence. This is oversimplified and does not easily fit into the world of online social networks (OSN) where nodes keep public profiles and adoption depends not only on a single summary statistic, but on the complex interaction between three factors (1) the propagating notion itself, (2) the properties of the local influencing neighbors, and (3) the global sentiment towards adoption (bullish/bearish) that is made available by the OSN provider. Further, viral memes usually demonstrate a phase transition phenomenon where the meme initially

grows slowly and then rapidly accelerates towards virality. This phase transition is insufficiently understood by prior work that concentrates on the time to convergence but not on the "tipping point" itself. In this work, we go a step further and propose a model that accounts for both the complex OSN interplays and the phase transition phenomenon.

To model OSN dynamics, we postulate a Unified Propagation (UP) model that has the following salient properties, (1) Propagations are modelled as vectors in the same high-dimensional space where node features exist, (2) Propagations spread in a synchronous stepwise manner whereby network states are updated simultaneously at discrete time intervals, (3) Nodes activate based on a shared decision function with shared thresholds that accounts for propagation affinity, local influence and global influence, and (4) The underlying network is based on the Preferential Attachment (PA) model [3] which yields a core-periphery structure and a scale-free degree distribution observed in real-world networks. Our model affords sampling ease from the propagation distribution which is similar to a gated, time-varying random walk on a network.

The design of our propagation model is motivated by recent work at the intersection of computation and social science [11] that shows small topic-oriented peripheral groups can have a large influence through network propagation pathways from periphery to core. Moreover, such groups are essential for incubating new concepts in a controlled environment, from where they can spread to large portions of the population through core-periphery pathways. Further, they report cases studies such as Google Glass, where propagations from a core node die out quickly when initial adoption is slow, which signals a population wide unpopularity that impedes adoption. We illustrate these scenarios in Fig. 5.1. Overall, both the initial incubation and global influence factors are necessary for determining the spread of the complex contagion. Finally, as we show in our experiments, we can observe the emergence of phase transitions in complex networked systems [105, 106] by running simulations on our proposed model.

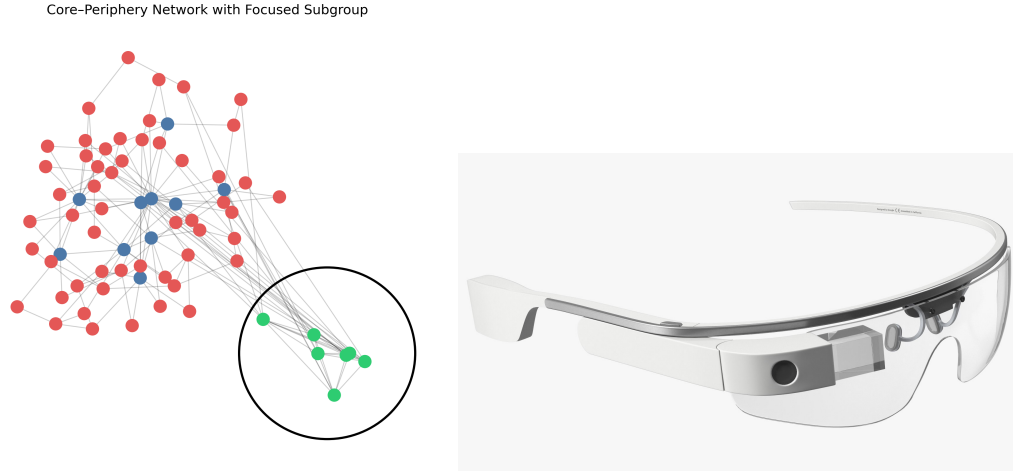


Figure 5.1: L-R: (a) Small topic-focused groups incubate new ideas [11]. Repeated exposure enables complex contagion. Propagation pathways move ideas from periphery to core. Once reaching the core, diffusion scales to the population. (b) Google Glass adoption stalled despite visibility. Core influence alone cannot create complex contagion. Highly connected core nodes expose many users immediately. If early adoption is weak, cascades die out quickly.

Our framework leverages propagation sampling from the high-dimensional node feature space to yield a simple personal affinity score. Prior works in psycho-social belief dynamics [107] usually incorporate latent personal factors with publicly visible attributes to determine individual and consequently collective behaviour; this process is illustrated in Fig. 5.2. Our approach is closest to [108] who model node opinions as unit-length vectors and propagate randomly sampled opinion vectors based on correlation scores; this process is depicted in Fig. 5.3. However, we differentiate ourselves by applying this approach to network cascades, which are distinct from opinion dynamics models in that we’re stochastically spreading activations on a network. To the best of our knowledge, we’re the first to model propagations as vectors for network cascades.

Our main contributions are summarized as follows:

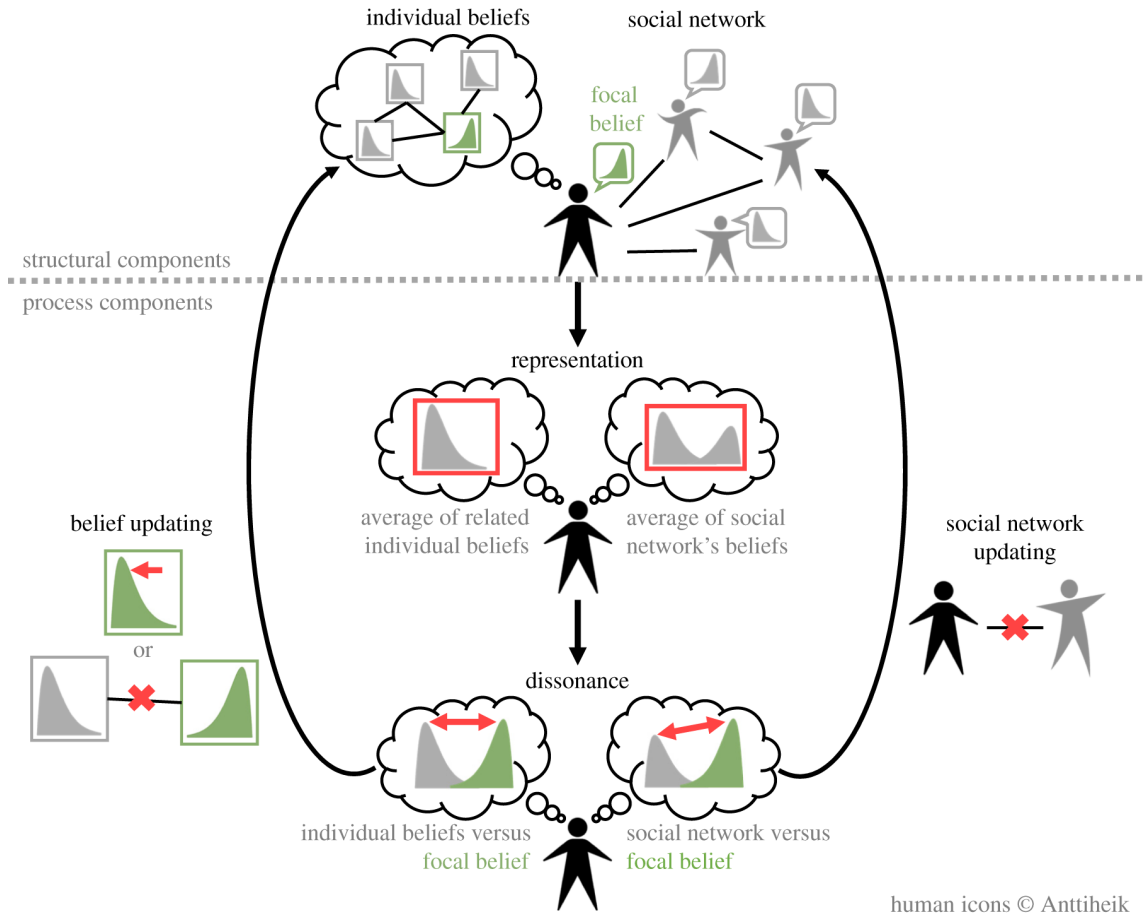


Figure 5.2: A unified framework of belief dynamics linking cognitive representations and social interactions: individuals form representations by aggregating personal and social beliefs, experience dissonance relative to focal beliefs, and update both beliefs and network ties through coupled cognitive and social processes.

- We introduce a general propagation-based model of complex contagion in which propagating notions and node attributes are represented in a shared high dimensional feature space.
- We propose the Unified Propagation (UP) model that integrates propagation affinity, local influence, and global influence, yielding a stochastic Markovian cascade

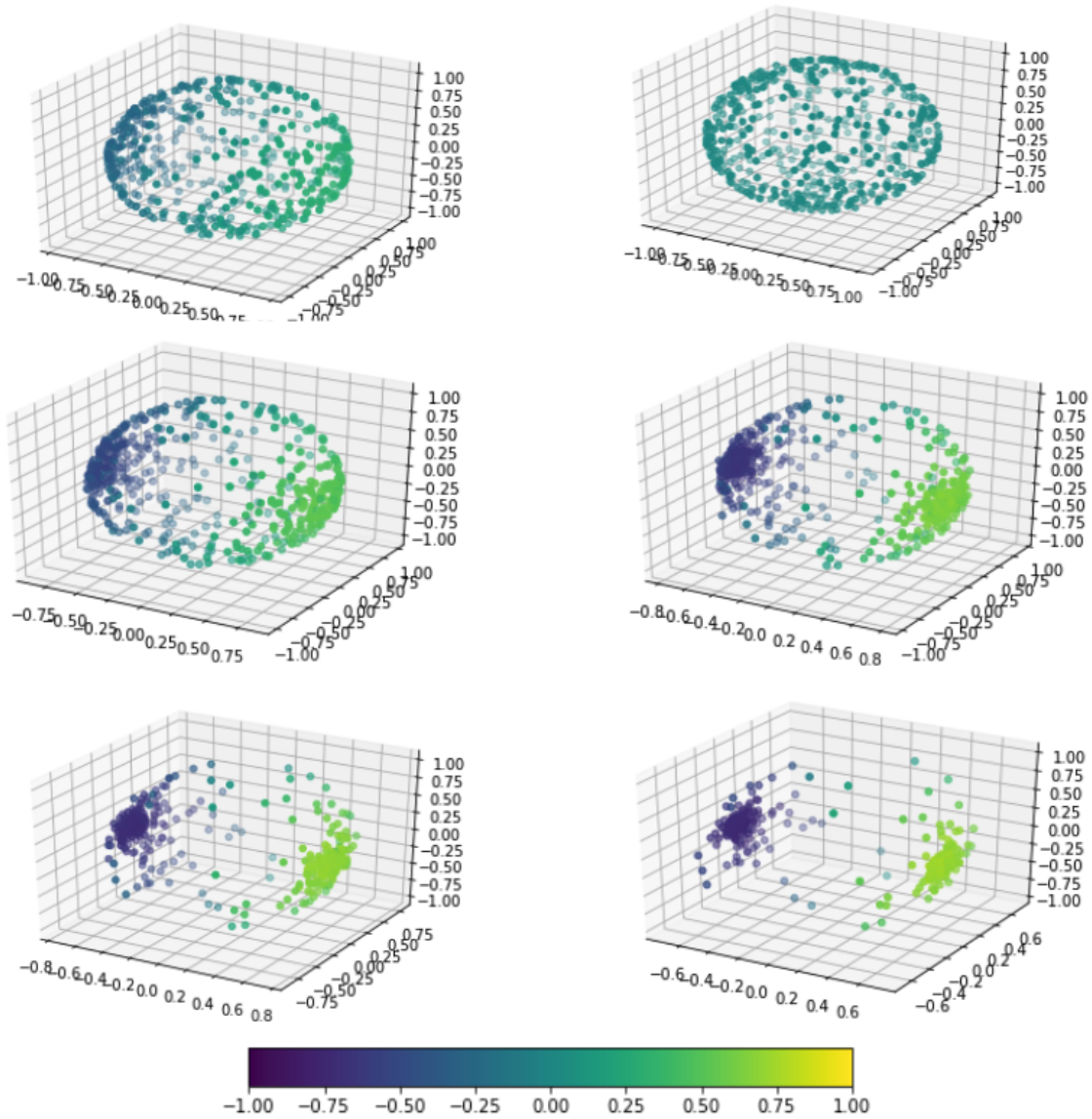


Figure 5.3: Illustration of geometric evolution of opinions towards polarization in [108], with the first three dimensions as spatial positions and the fourth dimension with a color scale.

process.

- We develop an MCMC-based simulation framework for sampling from the distri-

bution of cascade realizations induced by the model.

- We analyze incubation dynamics and derive conditions under which local reinforcement can lead to global diffusion through core–periphery pathways.
- We empirically demonstrate phase transition behavior and tipping points in cascade growth and show how early-stage dynamics can be used to anticipate virality.

5.2 Related Work

Diffusion of innovations. The study of social diffusion has a long tradition in sociology, where adoption is understood as a socially embedded process shaped by interpersonal influence, heterogeneity in individual thresholds, and the structure of communication networks. Early work on diffusion of innovations emphasizes the role of opinion leaders, social systems, and communication channels in shaping adoption trajectories [109], which typically follow a S-shaped curve as shown in Fig. 5.4. Complementary threshold-based models describe how collective behavior can emerge from heterogeneous individual decision rules driven by local social exposure [26]. Empirical and theoretical studies in this line highlight that adoption is rarely driven by single exposures alone, but instead depends on cumulative social reinforcement, normative pressures, and perceived legitimacy within social groups. This sociological perspective motivates the need for models that explicitly account for local context and heterogeneous influence pathways, rather than relying solely on aggregate exposure counts.

Complex contagion and computational models. Building on these foundations, the notion of complex contagion formalizes diffusion processes in which multiple reinforcing contacts are required before adoption occurs [10]. Computational and algorithmic

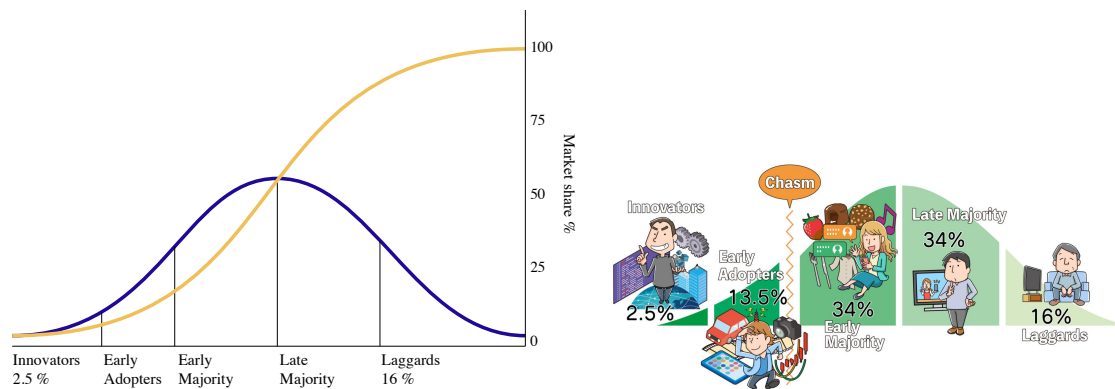


Figure 5.4: Diffusion of innovations [109] follows a characteristic S-shaped adoption pattern. A small number of innovators and early adopters initiate adoption. As social reinforcement increases, diffusion accelerates through the early and late majority, before slowing as the remaining laggards adopt. The bell-shaped curve represents the rate of adoption over time, while the cumulative adoption follows the S-curve.

mic models have investigated how network topology, clustering, and community structure affect the emergence of large-scale cascades under such reinforcement mechanisms [2, 110, 6]. Subsequent work has studied algorithmic and structural properties of complex contagions, including the role of strong ties and wide bridges, the effect of network interventions, and the computational aspects of triggering global cascades under threshold dynamics [9, 111]. Our work further emphasizes that reinforcement-based spreading processes exhibit sharp changes in behavior depending on network structure and initial conditions, suggesting the presence of critical regimes that separate small, localized cascades from system-wide diffusion.

Network propagation, label propagation, and diffusion processes. A complementary line of work models propagation through linear diffusion mechanisms such as random walks, label propagation, and heat or diffusion kernels. Seminal work by Zhu

and Ghahramani formulates semi-supervised learning as a harmonic extension problem on graphs, in which labels propagate via random walks with absorbing boundary conditions [112, 113]. Related formulations based on graph Laplacians and heat kernels characterize propagation through continuous-time diffusion processes and spectral filtering on networks [114, 115]. These methods yield closed-form solutions and strong convergence guarantees under mild connectivity assumptions. While such models have primarily been used for inference and representation learning, their random-walk interpretation provides a natural foundation for sampling-based propagation mechanisms. Our work builds on this perspective by coupling propagation in a high-dimensional feature space with stochastic activation dynamics, thereby extending linear diffusion models toward the sampling of network cascades.

Network formation and structural models. The behavior of diffusion and cascade processes is tightly linked to the generative mechanisms underlying network structure. Classical models of network formation include the small-world model, which captures short path lengths and high clustering through random rewiring [1], and preferential attachment models that produce scale-free degree distributions and pronounced core-periphery structure [3]. In addition, Kleinberg’s navigable small-world model introduces a geometric formulation that explains how long-range links enable efficient decentralized routing and shape information flow patterns [5]. These models have been widely used as benchmarks for studying diffusion, contagion, and search processes, and they provide principled abstractions for the heterogeneous connectivity and structural bottlenecks observed in online social networks. In this work, we adopt a preferential attachment framework to explicitly capture the emergence of core-periphery pathways that play a central role in our propagation dynamics.

Phase transitions and virality in networked systems. Diffusion and contagion processes on networks are known to exhibit phase transitions in which small changes in parameters or initial conditions lead to abrupt changes in macroscopic behavior. Early studies in complex systems and statistical physics have characterized critical points and tipping behavior in interacting systems [106]. Likewise, models of spreading activation networks in psychological theories of cognition [105] have demonstrated phase transitions by changing network topology or the activation parameters across phase boundaries. In the context of social and information networks, recent work has investigated how viral cascades transition from slow growth to rapid, large-scale spread, and how structural properties of cascade trees relate to different notions of virality [116]. Related studies on epidemic and information spreading further show that network heterogeneity and degree distributions fundamentally affect critical thresholds and outbreak dynamics [117]. These results motivate our focus on the transient evolution of cascades and the emergence of tipping points, rather than solely on asymptotic convergence behavior.

5.3 Method

5.3.1 Preliminaries

Consider a graph $G = (A, X)$, with a weighted adjacency matrix $A \in [0, 1]^{n \times n}$ and node attribute matrix $X \in [0, 1]^{n \times d}$ respectively, and node activations $Y \in \{0, 1\}^n$. We refer to the associated node-ids as $\mathcal{V} = \{1, \dots, n\}$. We denote the node feature $x_v \in [0, 1]^d$, and the node activation $y_v \in \{0, 1\}$. The active set of nodes is denoted as $S = \{v | y_v = 1\}$. Further, consider a propagation $C \in [0, 1]^d$ initiated at seed set S_0 , creating a cascade of network activations Y^t over time steps $t = 1, \dots, T_{P, S_0}$, with T_{C, S_0} denoting the convergence time. The spreading dynamics defined by the propagation and

activation rules form a discrete-time Markov process, since the network state at time t depends only on the state at time $t - 1$ and C .

The networks we use in the paper are based on the Preferential Attachment (PA) model [3] of network growth and link formation. The PA model constructs networks incrementally, unlike the configuration model [4] or the Watts-Strogatz small world model [1], and matches the power-law degree distribution of real world networks by construction. Formally, a PA graph $G(n, r)$ is a network of n nodes where each incoming node attaches to r existing nodes according to a probability proportional to their degree. Therefore, this creates an early mover advantage, causing initial nodes to have high degree centrality and later nodes to be on the periphery. The resulting core-periphery network structure is ideal for our analysis of the spread of complex contagions. Note that PA networks have been used by prior work [103], but our contagion model exploits the core-periphery structure more explicitly and gives power to the periphery as expected from sociological analysis [11].

Node features are constructed using Laplacian node embeddings, in other words, by taking the k smallest eigenvectors of the graph Laplacian. Formally, the graph Laplacian $L = D - A$, where D is the degree matrix of the graph, with its eigenvector decomposition given by $L = Q\Delta Q^T$ as usual. Therefore, the node feature matrix is the submatrix $X = Q_{:,k}$. The graph Laplacian appears frequently in graph signal processing, with the eigenvalues-eigenvector pairs yielding frequencies in the graph Fourier domain. The smallest eigenvectors correspond to low frequencies that vary smoothly over the graph and carry structural information as well. Moreover, certain eigenvectors carry important information about the connectivity of the graph [118] and are used in graph cuts, community detection and clustering [119]. Therefore, while there exist other node embedding approaches [120, 121] that have shown promise in problems such as node classification and link detection, we use Laplacian embeddings in this work due to their efficient

construction using eigenvector factorization.

Since the PA model yields an unweighted network, we assign edge weights using the Laplacian node embeddings,

$$A[i, j] = x_i^T x_j \quad (5.1)$$

The edge weights then correspond to influences coming from the neighboring nodes and can be aggregated.

5.3.2 Unified Propagation of Complex Contagion

There are many variations of the complex contagion model [10], among which the most commonly used is the k -complex contagion: a node activates when at least k of its neighbors have been activated. In contrast, simple contagion models [26, 27] such as linear threshold, independent cascade and SIR can cause infections from just a single neighbor and enjoy submodularity properties. The submodularity property endows the propagation with diminishing returns, and therefore is amenable to a fast greedy approximation for downstream tasks such as influence maximization (IM). However, complex contagion is not submodular in its original form, and variants have been proposed to achieve submodularity for the ease of analysis [8]. While these models are very expressive and have been used extensively for influence maximization, they don't account for (1) node features, and (2) cannot give rise to the characteristic S-curve in technology adoption [122] or exhibit phase transition behavior [7]. In this work, we propose a complex contagion model that addresses the aforementioned issues and combines 3 sources of influence in a threshold based model. In the following subsections we define each influence source separately and show how they're aggregated to run propagation simulations.

Propagation Affinity For a propagation $C \in [0, 1]^d$ from the space of node features, we can construct a normalized vector $\hat{C} = \frac{C}{\|C\|}$ in the d -dimensional hypersphere to denote the direction of propagation. Inspired by prior work on opinion dynamics [108], we define the propagation affinity for a node v as the dot-product similarity between the propagation and the node feature vectors.

$$F(\hat{C}, v) = \hat{C}^T x_v \quad (5.2)$$

Note that by construction the dot-product $\hat{C}^T v \in [-1, 1]$ since both $\hat{C}$ and x_v are normalized vectors. However, to keep influences calibrated across the 3 sources, we scale the score so that it lies in $[0, 1]$ as follows,

$$\hat{F}(\hat{C}, v) = \frac{1 + \hat{C}^T x_v}{2} \quad (5.3)$$

Local Influence The local influence corresponds to the neighborhood effects and/or the peer pressure, as also modelled by prior models of complex contagion. While most of the focus has been on the *number of activating neighbors*, we argue that a better measure is the *density of activating neighbors*. This has the effect, for instance, of allowing a low-degree peripheral node to activate even if the actual number of activating neighbors is very low. Explicitly,

$$LI(v, Nbr(v)) = \frac{\sum_{w \in M(v)} A_{v,w} - \sum_{w \in Nbr(v) \setminus M(v)} A_{v,w}}{d_v} \quad (5.4)$$

where $M(v)$ denotes the set of activating neighbors of v and $d_v = \sum_{w \in Nbr(v)} A_{v,w}$ is the degree of node v . This local influence term accounts for the competitive influence weights for activation from the neighbors and their density. Similar to the propagation affinity, we scale the value so that it lies in $[0, 1]$.

$$\hat{LI}(v, Nbr(v)) = \frac{1 + LI(v, Nbr(v))}{2} \quad (5.5)$$

The local influence term is similar in spirit to the coordination games model [110, 123] where nodes get payoffs from neighboring nodes for aligning their behaviors.

Global Influence Besides the network influences, population level influences also account for the spread of contagions [123]. This is especially true in OSNs, where the population level adoption statistics are provided directly to the user. Therefore we use the rate of adoption as a measure of global influence,

$$GI(\hat{C}, G) = \frac{|S|}{n} \quad (5.6)$$

where S is the dynamically varying set of activating neighbors. Clearly, $GI(\hat{C}, G) \sim 0$ initially and increases to 1 as S grows.

Aggregating Influences and Simulating Contagion Cumulatively, the 3 influence scores can be aggregated to yield the probability of activation for v ,

$$\begin{aligned} \text{Prob}(v|Nbr(v), \hat{C}, G) = & \gamma(\alpha \hat{F}(\hat{C}, v) + \beta LI(v, Nbr(v)) \\ & + (1 - \alpha - \beta)GI(\hat{C}, G)) \end{aligned} \quad (5.7)$$

where $0 \leq \alpha \leq 1, 0 \leq \beta \leq 1, \alpha + \beta \leq 1$ and γ is a parameter for modulating the simulation temperature.

We use $\text{Prob}(v|Nbr(v), \hat{C})$ to perform Markov Chain Monte Carlo (MCMC) simulation of cascades and to draw samples from the distribution of possible propagation outcomes. In particular, we construct a Markov chain over network activation states and use a Metropolis-style update rule in which candidate activation updates are proposed according to the Bernoulli probabilities induced by $\text{Prob}(v|Nbr(v), \hat{C})$ and accepted deterministically in the irreversible activation setting.

Specifically, at each time step t , we propose activation updates for all currently inactive nodes independently according to their Bernoulli probabilities and accept all suc-

cessful proposals.

$$Y_v^t = \begin{cases} 1 & : Y_v^{t-1} = 1, \\ X \sim \text{Bern}(\text{Prob}(v|Nbr(v), \hat{C}, G)) & : \text{otherwise} \end{cases}$$

Note that once a node gets activated it stays activated for the rest of the simulation. The time steps of the simulation can be of the order of days, weeks or months in a real-world scenario, whereby nodes activate asynchronously because of multiple contacts with neighboring nodes in the network. We keep redrawing node samples until the network state hasn't changed for a minimum cooling period of ϵ steps, and this yields the steady state. Since activations are irreversible in our model, the resulting Markov chain is monotone and converges to a stationary distribution over final cascade states. This yields a valid MCMC sampler over cascade realizations induced by the propagation dynamics.

5.3.3 Conditions for Local Incubation

In this section, we derive conditions for local incubation for the foregoing propagation model, primarily focusing on the second adopter.; the analysis for subsequent adopters is similar. Let the seed node v_0 be connected to a clique $\{v_1, v_2, \dots, v_k\}$ of size k . Without loss of generality, let v_0 attach to the clique at v_1 . Then, from Eq. 5.7, the activation probability for v_1 is,

$$\text{Prob}(v_1|Nbr(v), \hat{C}) = \gamma(\alpha \hat{F}(\hat{C}, v_1) + \beta \hat{L}I(v, Nbr(v))) \quad (5.8)$$

$$= \gamma(\alpha \frac{1 + x_{v_1}^T \hat{C}}{2} + \beta \frac{1 + \frac{1-(k-1)}{k}}{2}) \quad (5.9)$$

$$= \gamma(\alpha \frac{1 + x_{v_1}^T \hat{C}}{2} + \beta \frac{1}{k}) \quad (5.10)$$

where we have assumed $GI(\hat{C}, G) \sim 0$ for a large network during local incubation and taken an unweighted network; the extension to the weighted case is straightforward.

Given a cooling period of ϵ steps, the likelihood of node v_1 activating is the probability of success in ϵ trials of $\text{Bern}(\text{Prob}(v_1|Nbr(v), \hat{C}))$,

$$\text{Prob}^\epsilon(v_1) = 1 - (1 - \gamma(\alpha \frac{1 + x_{v_1}^T \hat{C}}{2} + \beta \frac{1}{k}))^\epsilon \quad (5.11)$$

For instance, taking $\alpha = \beta = \frac{1}{2}$, $\gamma = 1$ and $\epsilon = 1$,

$$\text{Prob}^\epsilon(v_1) = \frac{k(1 + x_{v_1}^T \hat{C}) + 2}{4k} \quad (5.12)$$

Since $x_{v_1}^T \hat{C} = 1$, and assuming $k \gg 2$,

$$\text{Prob}^\epsilon(v_1) \sim \frac{1}{2} \quad (5.13)$$

We can see here that even with complete alignment to the propagating notion, the probability of activation is squarely in the middle due to the singleton local influence initially. On the other hand, letting $x_{v_1}^T \hat{C} \sim -1$, clearly $\text{Prob}^\epsilon(v_1) \sim 0$. Increasing ϵ has the effect of increasing the the number of trials, allowing lower-probability evens to happen. For the per-step probability, we can see that when k is small the second term contributes more local influence. Therefore, a smaller group favors chances of local incubation. In a larger group, the propagation affinity score needs to be higher since local influence for the bridge node will be very small.

5.3.4 Connection to Random Walks

The proposed contagion model is connected to existing random walk diffusion models on networks. However, the proposed contagion model is more like a stochastic growth process that can be seen as a gated, time-varying random walk. The propagation can be written in matrix-vector form precisely as,

$$Y^{t+1} \sim \gamma(\alpha \hat{F}(\hat{C}, V) + \beta(2Y^t - 1)D^{-1}A + (1 - \alpha - \beta)GI(\hat{C}^t, G)I_n)1_n \quad (5.14)$$

where $\hat{F}(\hat{C}, V) = \text{Diag}(\hat{F}(\hat{C}, v_1), \dots, \hat{F}(\hat{C}, v_n))$ is the diagonal matrix of node affinity scores, I_n is the $n \times n$ identity matrix and 1_n is the vector of all ones. The matrix modulating the propagation is very much like a random walk matrix [112, 112, 113], however it's dynamic and depends on the activation states of the nodes. Moreover, the probabilistic node activations function as a gate that changes the transition matrix. Moreover, unlike a classical random walk, once activated the nodes act as constant sources of flow. Finally, there is no closed form analytical solution in terms of the system's eigenvalues for the rate of convergence due to stochasticity. Instead, we run simulations to estimate asymptotic convergence rates.

5.3.5 Phase Transition Behavior

Understanding phase transitions in networked systems is an important area of research in psychology, ecology and dynamical systems [105, 106]. In the context of social contagion, this has not been studied significantly, and is especially hard because complex contagions have a distribution skewed towards small spreads, and viral events lie in the tail end of that distribution. While large-scale datasets have recently become available for this purpose [116], they account more for information diffusion and not behavioural diffusion. However, it's well known from studies in management science that technological adoption follows an S-curve [122], where early adopters incubate an invention and then after a slow initial period of adoption the invention reaches critical mass and propagates rapidly to a large population, with late stage adopters forming the top part of the S-curve. This can be further linked to a recent sociological study [11] that shows how peripheral institutional groups can propagate large spreads through core-periphery adoption pathways. Our proposed contagion model captures this phase transition behavior and achieves periphery-core spreading. Intuitively, the contagion accelerates once

the initial nodes in the PA network, which happen to have the highest centrality, get infected. Therefore, in the initial stages spreading is slow due to lower branching, and gets accelerated from both high branching through high centrality nodes and critical mass effects. Notably, our contagion model explicitly accounts for global influence and drives the propagation towards virality. This phase transition behavior is more subtle than previous works where mere changes in connectedness lead to the formation of a giant component [124].

5.4 Experiments

We evaluate the proposed propagation-driven cascade model on PA networks in order to analyze how stochastic propagation, reinforcement effects, and network structure jointly shape diffusion outcomes. In all experiments, we simulate influence cascades generated from the propagation dynamics described in Section 5.3.2, while varying the location of initial seeds, the spread control parameters, and the structural properties of the underlying networks.

Specifically, we wish to answer the following research questions:

- **RQ1:** What kind of spread distribution is attained?
- **RQ2:** What’re the dynamics of spread evolution to attain virality?

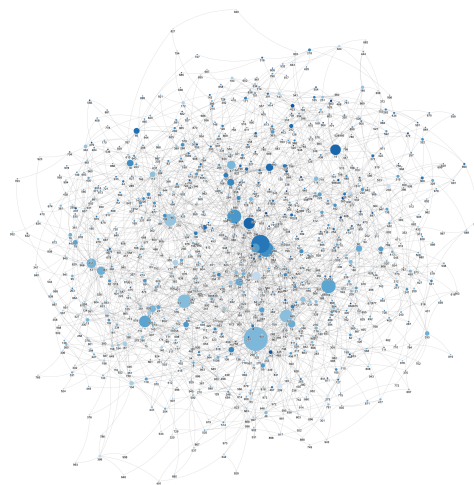


Figure 5.5: Qualitative illustration of a PA network, $G(1000, 2)$ used in our simulations. Node size and color opacity indicate degree and the component along the first spectral embedding vector respectively.

- **RQ3:** How does the time to attain virality vary with network structure?
- **RQ4:** How does the simulation vary with the parameters?
- **RQ5:** How do misaligned ideas get propagated?
- **RQ6:** How does complex contagion spread on real-world networks?

Implementation details. Unless stated otherwise, all experiments use the same simulation configuration. For each node, we draw 20 stochastic samples of propagation trajectories. The propagation parameters are fixed to $\alpha = 1/3$ and $\beta = 1/3$, and the simulation temperature set to $\gamma = 0.05$. The cooling down period is set to $\epsilon = 10$ number of steps. For every node, the initial propagation vector is initialized to be the node’s own feature vector; consequently, all experiments evaluate *self-propagation* dynamics unless explicitly noted otherwise. The underlying network is generated using a preferential attachment (PA) model with 1000 nodes, where each newly arriving node attaches to 2 existing nodes. We construct node features by taking the first $k = 10$ eigenvectors of the graph Laplacian. A qualitative visualization of the resulting network structure is shown in Figure 5.5.

We categorize nodes into three structural segments based on degree centrality: *core* nodes correspond to the top 10% of nodes ranked by degree, *peripheral* nodes correspond to the bottom 10%, and the remaining nodes are treated as *intermediate*. Throughout the experiments, we report and compare trends separately for these three node segments in order to analyze how core–periphery structure shapes propagation and cascade dynamics.

5.4.1 RQ1: Spread Distributions.

In Figure 5.6a we illustrate the distribution of final cascade sizes across repeated simulations, together with the relationship between the average degree of seed nodes and

the resulting spread in Figure 5.6b.

The distribution in Figure 5.6a exhibits a clearly bimodal structure. Most realizations terminate quickly after affecting only a small fraction of the network, while a smaller but significant fraction of runs lead to large-scale cascades that reach a substantial portion of the graph. This behavior is consistent with the stochastic sampling nature of our propagation model and reflects the presence of two dominant regimes: localized diffusion confined to a neighborhood of the initial seeds, and global diffusion driven by reinforcement and repeated exposure. We note not all propagations from core nodes reach virality, such as high-profile failures (eg. Google Glass) where slow initial growth leads to extinctions. Also, some propagations from peripheral nodes do reach virality, corresponding to focused groups that have core-periphery pathways to adoption [11].

The degree–virality relationship shown in Figure 5.6b reveals a strong positive correlation between the structural centrality of the seed nodes and the probability of large-scale spread. Seeds located in the core of the PA network, characterized by high degree and dense connectivity, are substantially more likely to trigger global cascades. In contrast, cascades initiated from peripheral nodes typically remain small and short-lived. This observation aligns with the core–periphery pathways emphasized by our model: propagation originating in the core rapidly encounters multiple reinforcing paths, whereas peripheral seeds lack sufficient redundant exposure to overcome activation thresholds.

5.4.2 RQ2: Spread Evolution toward Virality.

Amongst the simulations that achieve virality, we want to understand the propagation dynamics that drive it and whether we observe a phase transition with a tipping point.

Figure 5.7a shows the cumulative number of activated nodes as a function of simulation time for representative viral simulations from the three node segments. The curves

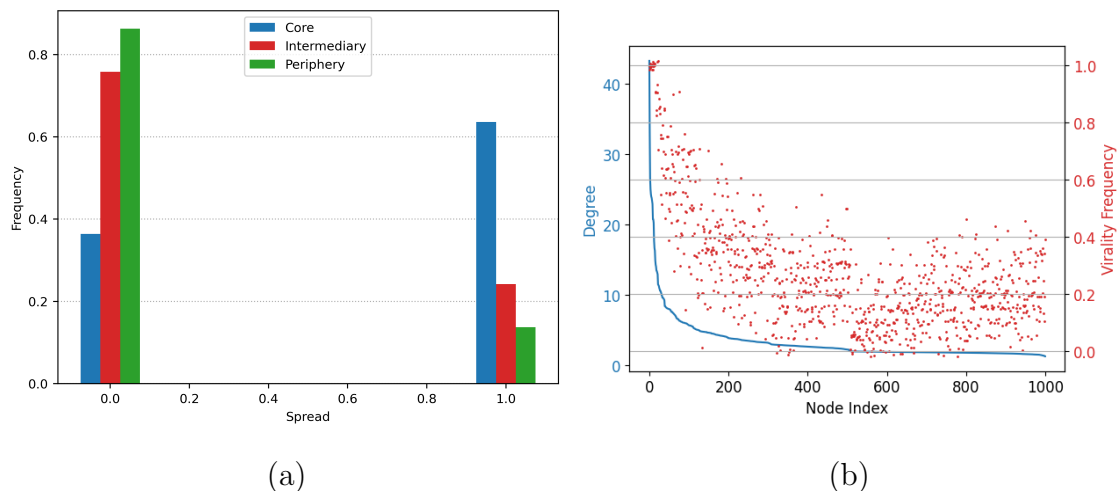


Figure 5.6: (a): Distribution of final cascade sizes across simulation runs, showing a bimodal separation between localized and large-scale cascades. (b): Correlation between average seed-node degree and cascade size, illustrating the influence of core versus peripheral seeding.

reveal a two-stage diffusion pattern. In the early phase, growth is slow and sub-linear, reflecting the incubation period where branching remains low and both local and global influence remain low as well. Once the cascade reaches a sufficiently connected region of the network, typically within the core, the growth rate increases sharply. This transition corresponds to a tipping point at which there is high branching that propels the propagation towards virality.

The same phenomenon is visible in Figure 5.7b, which plots the number of newly activated nodes at each time step. The spike in new adopters marks the onset of the supercritical regime of the cascade process. After this point, activation becomes self-sustaining due to the abundance of overlapping propagation paths and accumulated exposure. These results demonstrate that virality in our model is not driven by a single influential event but emerges from the interaction between random-walk propagation and nonlinear activation dynamics.

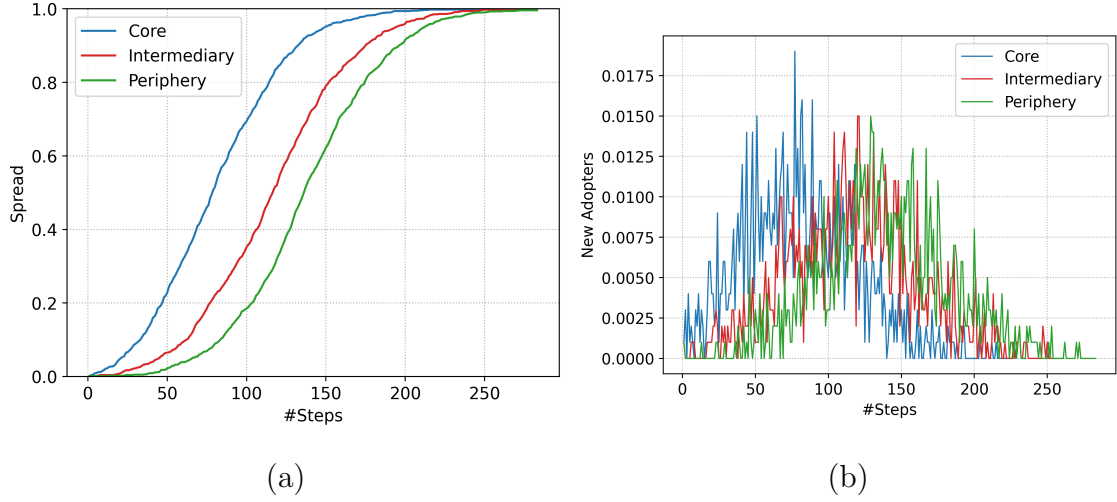


Figure 5.7: (a) Temporal evolution of cumulative activated nodes during cascades. (b) Number of newly activated nodes at each time step, highlighting the non-linear phase transition into virality and the tipping point into rapid diffusion.

5.4.3 RQ3: Time to Virality

Figure 5.8 reports the average time required for a cascade to reach the viral regime as a function of network size, together with the corresponding network diameter measured on the same preferential attachment graphs. A salient observation is that the time to virality grows very slowly as the size of the network increases. Even as the number of nodes increases by a large factor, the increase in the time required to reach the viral regime remains modest, indicating a highly sublinear scaling with network size.

More importantly, the plot shows a strong correlation between the time to virality and the diameter of the underlying network. Across all network sizes, the two quantities closely track each other: instances with larger diameters consistently exhibit longer times to virality, while networks with smaller diameters reach the viral regime earlier. This tight coupling suggests that the dominant factor governing the onset of large-scale diffusion in our model is the characteristic shortest-path scale of the graph, rather than the total

number of nodes. Since cascades must propagate through a sequence of intermediate activations before reaching highly connected regions that enable rapid reinforcement, the effective depth of the network—as captured by its diameter—directly constrains how quickly the process can enter the supercritical growth phase.

The observed slow growth of the time to virality is therefore consistent with known structural properties of preferential attachment graphs. In particular, the diameter of random preferential attachment networks is known to grow extremely slowly with the number of nodes, on the order of $\Theta(\log n / \log \log n)$ [125]. Since propagation trajectories in our model rapidly funnel toward the network core through short paths, the number of steps required before sufficient reinforcement becomes possible remains closely tied to this slowly growing diameter. This provides a structural explanation for why the time to virality closely follows the diameter curve in Figure 5.8, and only weakly depends on the overall network size.

5.4.4 RQ4: Effect of the Model Parameters.

We analyze how the model parameters affect both the *virality frequency*, defined as the fraction of simulation runs that enter the viral regime, and the *average time to virality* among the runs that do become viral. Figures 5.9, 5.10, and 5.11 report these trends for the feature influence weight, the local influence parameter, and the global influence parameter, respectively.

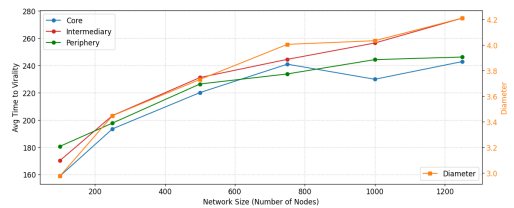


Figure 5.8: Average time required to reach the viral regime and the network diameter as a function of network size.

Figure 5.9 shows a monotonic increase in virality frequency together with a monotonic

decrease in the average time to virality as the feature influence weight increases. This behavior is a direct consequence of the geometry of the spectral embeddings used in our experiments. The embeddings exhibit high feature similarity across large portions of the network, and as a result, self-propagating simulations are consistently reinforced by neighboring nodes with similar representations. Increasing the feature influence weight therefore strengthens early reinforcement, making it both more likely that cascades enter the viral regime and faster for successful cascades to do so.

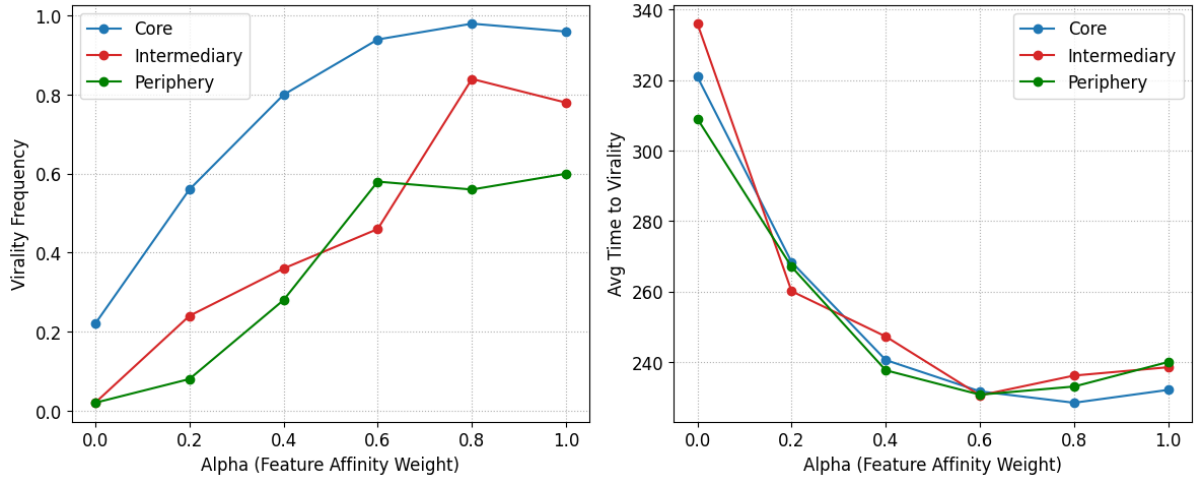


Figure 5.9: Effect of the feature affinity weight α on the virality frequency and time to virality.

In contrast, Figure 5.10 demonstrates that increasing the weight placed on local influence leads to a systematic decrease in virality frequency and smaller cascade sizes, together with an increase in the average time to virality. Strong local influence overemphasizes reinforcement within immediate neighborhoods and suppresses propagation across structurally diverse regions of the graph. This induces a network inhibition effect in which cascades become confined to locally homogeneous areas and fail to access the wide bridges required for global diffusion. This behavior is consistent with the conditions for local incubation discussed in Section 5.3.3: while local reinforcement is necessary to

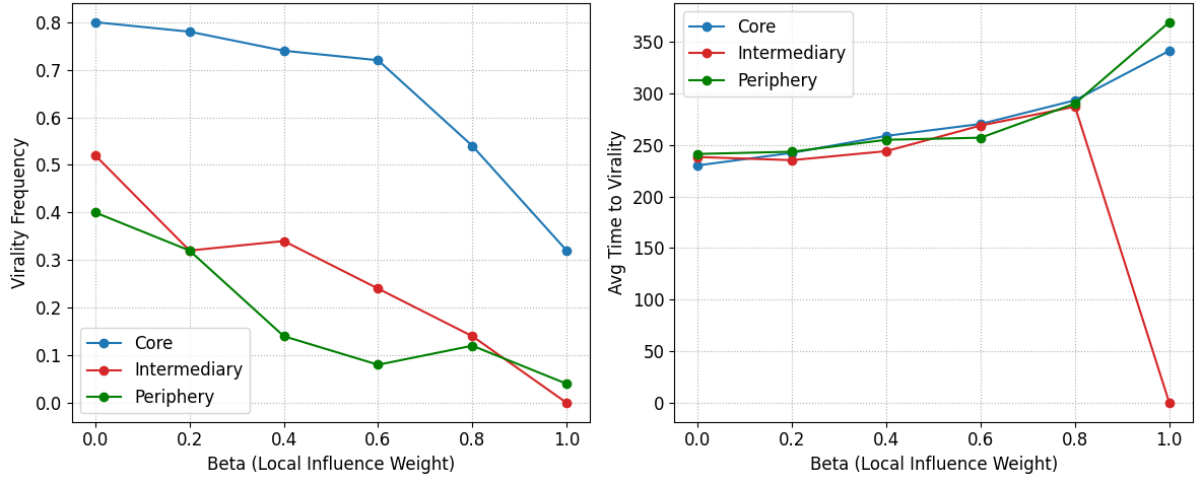


Figure 5.10: Effect of the local influence weight β on the virality frequency and time to virality.

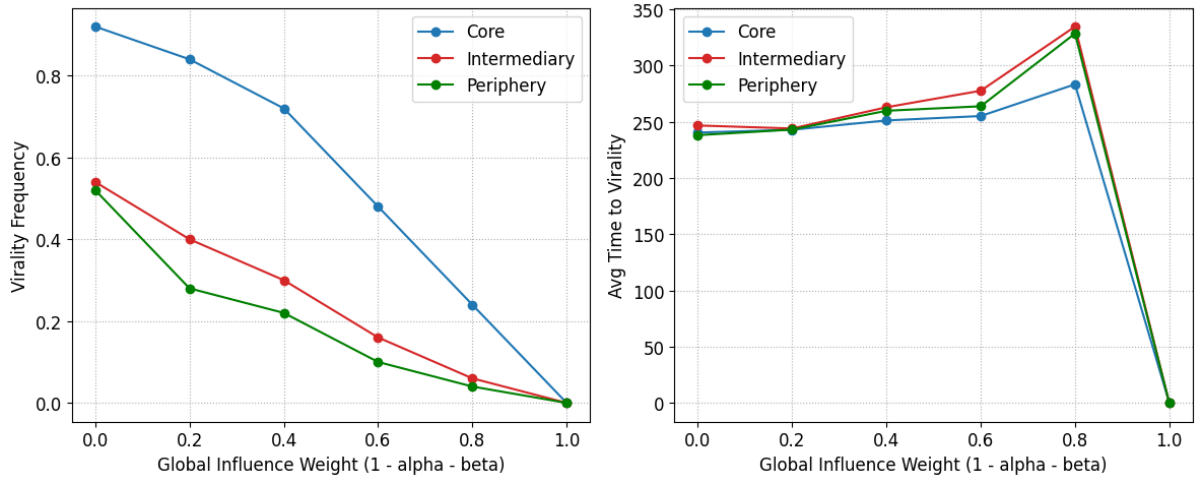


Figure 5.11: Effect of the global influence weight $(1 - \alpha - \beta)$ on the virality frequency and time to virality.

sustain early growth, excessive reliance on purely local influence prevents the transition from the incubation stage to large-scale diffusion.

Finally, Figure 5.11 shows that increasing the global influence weight also produces a clear inhibition effect. Higher global influence reduces the virality frequency and increases

the average time to virality. Strong global influence amplifies activation based on limited early exposure, effectively destabilizing the incubation stage by triggering premature activation in peripheral or weakly connected regions of the network. Because these early activations lack sufficient structural support and reinforcement pathways, many cascades fail to develop into sustained diffusion processes. Together, these results highlight that both excessive local and excessive global influence hinder the formation of successful cascades, and that a balanced interaction between local incubation and global activation is required for high virality frequency and rapid onset of virality.

5.4.5 RQ5: Propagation of Misaligned Ideas.

Finally, we study how ideas that are initially misaligned with the feature distributions of the population propagate in the network.

Figure 5.12 reports the relationship between the similarity of the initial propagation vector to the feature profiles of the seed nodes and the resulting virality frequency and the average time to virality. When the propagated idea is well aligned with the local neighborhood of the seeds, diffusion proceeds smoothly and often reaches the core, enabling large and fast cascades. In contrast, ideas that are initially dissimilar to the surrounding population experience substantially lower early activation rates and times to virality, and are more likely to die before reaching structurally influential regions.

Nevertheless, a non-negligible fraction of misaligned ideas still achieve wide spread. These cases correspond to rare stochastic trajectories in which early propagation reaches highly connected nodes that provide sufficient reinforcement to overcome initial resistance. This result illustrates how structurally central actors can act as bridges for the dissemination of novel or unpopular ideas, even when local compatibility is low.

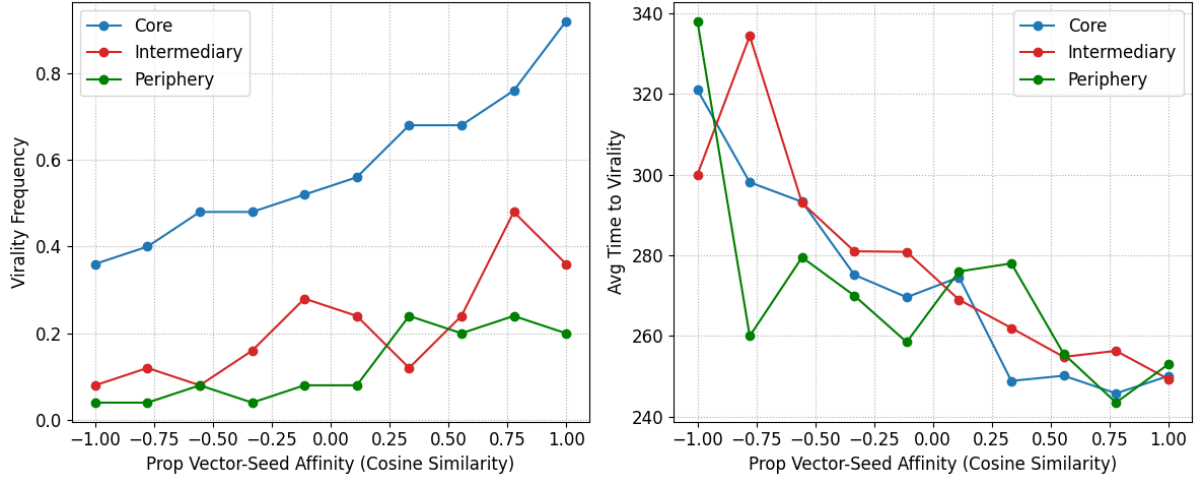


Figure 5.12: Effect of the propagated idea affinity to seed-node feature profiles to the virality frequency and the time to virality, illustrating the diffusion of initially misaligned ideas.

5.4.6 RQ6: Real-world Complex Contagion.

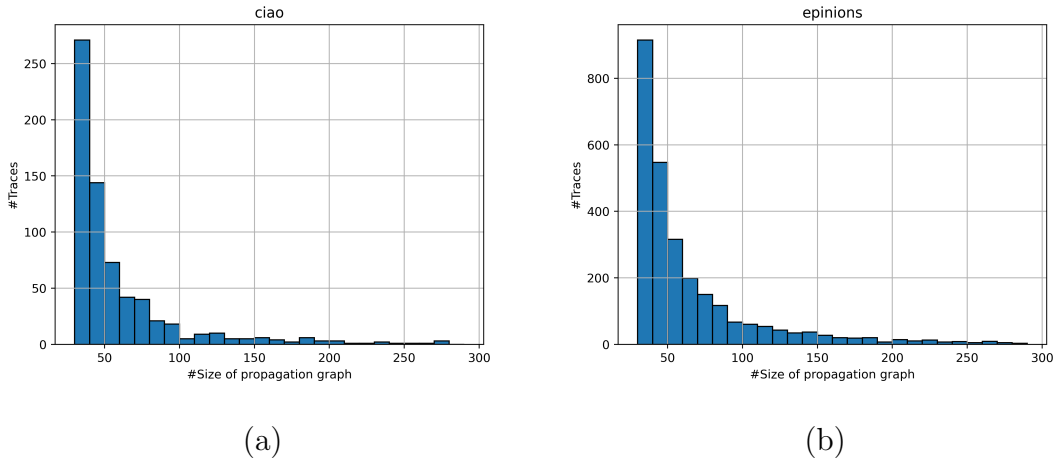


Figure 5.13: Distribution of size of cascade traces on the (a) Ciao and (b) Epinions datasets indicates a power-law scaling where most cascades are small and there are very few big cascades.

We evaluate real-world complex contagion traces using two widely used datasets, Epinions and Ciao, which are datasets collected from two popular product review sites, where each user can specify their trust relation in addition to rating products [126, 19]. Both contain temporally ordered user interactions and trust relations, from which cascades can be reconstructed. Specifically, if u trusts v and v rated product p before u then we take it that u was influenced by v .

A key challenge in modeling real-world complex contagion is that the empirical cascade size distributions are extremely skewed. As shown in Figure 5.13, both the Ciao and Epinions datasets exhibit heavy-tailed distributions in which the vast majority of cascade traces are very small, and only a tiny fraction of cascades grow large. This strong imbalance makes learning and evaluating models of virality particularly difficult, since large cascades—which are the most informative for understanding phase transitions and global diffusion—are severely underrepresented in the data.

Moreover, real-world propagation data is substantially noisier than controlled simulations. Nodes continuously enter and leave, interaction patterns evolve over time, and multiple overlapping cascades spread simultaneously, often interfering with one another. These effects obscure the underlying propagation mechanisms and make it difficult to isolate the role of local reinforcement, global influence, and network structure. In contrast, our model provides a controlled and interpretable environment in which individual propagation trajectories can be sampled independently and the influence of specific structural and behavioral factors can be analyzed in isolation. As a result, propagation dynamics and phase transition behavior that are difficult to observe directly in real-world data become substantially clearer in the synthetic setting.

5.5 Discussion

Generality and feature-driven adaptability. A key strength of the proposed propagation cascade framework is its generality with respect to the choice of node features and their underlying distributions. In our experiments, features are derived from spectral embeddings of the network, which induces a smooth and highly correlated feature landscape. However, the model does not rely on any specific embedding method or feature construction. In practical applications, node features may arise from specialized and heterogeneous data sources, including multimodal representations such as text embeddings, image-derived features, behavioral profiles, or contextual signals. The propagation mechanism operates directly on feature vectors and therefore naturally extends to such settings. Importantly, changing the feature distribution alters the geometry of the propagation space and the similarity structure that governs reinforcement during diffusion. As a result, different feature modalities and embedding choices can substantially modify local incubation dynamics, the accessibility of structurally diverse neighborhoods, and the likelihood of global breakout. This flexibility allows the model to be adapted to a wide range of real-world influence and information diffusion scenarios without changing the underlying propagation mechanism.

Early detection of phase transitions. Our results also suggest that the onset of large-scale diffusion can be anticipated from early-stage cascade dynamics. In particular, the growth trajectories observed during the initial incubation phase already contain strong signals of whether a cascade is likely to enter the supercritical regime. Consistent with the phase-transition behavior identified in our parameter sweeps, early growth rates and the accumulation of reinforcement paths provide a reliable proxy for the proximity of the system to its tipping point. This indicates that phase transitions in propagation

behavior can be predicted at an early stage by monitoring short-term growth patterns, such as the rate of newly activated nodes and the expansion of the active frontier across structurally diverse regions. From a practical perspective, this opens the possibility of forecasting viral outbreaks or intervention opportunities before large-scale diffusion has fully unfolded.

5.6 Takeaways

In this chapter, we presented a propagation-driven framework for modeling complex contagion in networks that explicitly integrates feature-based affinity, local reinforcement, and global influence into a unified stochastic activation process. By representing propagating notions in the same feature space as network nodes, our model naturally captures heterogeneity in individual susceptibility and enables propagation to be interpreted as a gated, time-varying random walk.

Through extensive simulations on preferential attachment networks, we demonstrated how local incubation, core-periphery pathways, and parameter interactions jointly shape the emergence of large-scale cascades. Our results highlight that both excessive local reinforcement and excessive global influence can inhibit diffusion, and that successful cascades arise from a delicate balance between incubation and structural exploration. We further showed that phase transitions and tipping points are reflected in early-stage growth dynamics, providing opportunities for early prediction of viral outbreaks. Finally, by offering a controlled and interpretable generative framework, our model provides a complementary tool for understanding propagation mechanisms that are often obscured in noisy empirical data.

5.7 Extended Analysis

5.7.1 Comparisons to Prior Propagation Models

Table 5.1 summarizes the key differences between classical contagion models and our unified propagation framework. The remainder of this appendix discusses these models in greater detail and highlights how their propagation assumptions influence cascade dynamics.

Model	Complex	Prop. Affinity	Tipping
IC	No	No	Weak
LT	Yes (local)	No	Limited
k-Complex	Yes (hard threshold)	No	Sensitive to k
Data-driven	Implicit	No	Not modeled
UP (ours)	Yes	Yes	Strong

Table 5.1: Comparison of propagation models discussed in this paper. The unified propagation model integrates propagation affinity with local and global influence, enabling incubation dynamics and tipping-point behavior that are difficult to capture in classical contagion models.

Independent Cascade (IC) model. Figure 5.14 compares the cascade size distributions produced by the Independent Cascade (IC) model with activation probabilities $p = 0.25$ and $p = 0.5$. In both cases the IC model exhibits broad distributions with significant mass at intermediate cascade sizes. Particularly, while increasing the activation probability increases branching and shifts the distribution toward larger spreads, the resulting cascades remain concentrated around moderate spreads rather than producing the sharp tipping-point behavior observed in our unified propagation model.

This behavior arises from the single-contact activation constraint in the IC model:

each active node has exactly one chance to activate each neighbor. As a result, propagation proceeds as a branching process whose growth is limited by early stochastic failures. Increasing p increases the branching factor and therefore the expected cascade size, but does not fundamentally change the qualitative shape of the distribution. Even at $p = 0.5$, where branching is substantially higher, cascades do not exhibit the abrupt transition between localized diffusion and system-wide virality that characterizes our model. In contrast, the unified propagation model allows repeated reinforcement through propagation affinity and accumulated local and global influences, enabling incubation phases that can eventually trigger rapid viral growth through sufficient reinforcement.

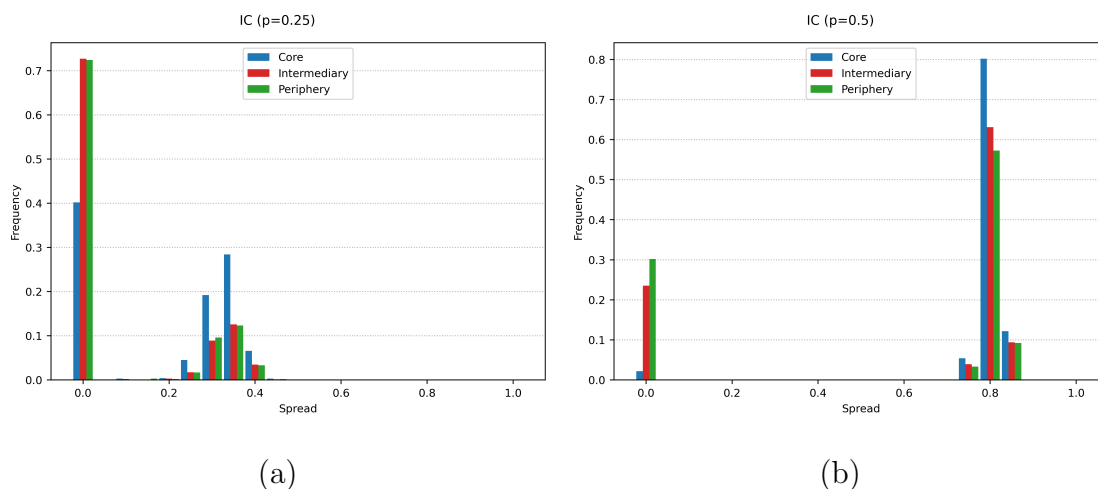


Figure 5.14: Distribution of size of cascade traces on a 1000-node PA network for the Independent Cascade (IC) model, with activation probability (a) $p = 0.25$ and (b) $p = 0.5$.

Linear Threshold (LT) model. The Linear Threshold (LT) model represents another canonical contagion framework and is most closely related to the high local-influence regime of our unified propagation model. In the LT model, each node activates once the weighted influence from its neighbors exceeds a threshold drawn from a specified distribu-

tion. When these thresholds are repeatedly sampled through MCMC-style simulations, the resulting cascade realizations resemble Bernoulli sampling over activation events in our unified propagation model. However, the LT mechanism strongly emphasizes local influence and does not explicitly incorporate propagation affinity or global sentiment. Consequently, when cascades originate from a single seed, the spread distribution is typically heavily concentrated near zero. The absence of a propagation mechanism that can accumulate influence from both propagation affinity as well as local/global mechanisms prevents the formation of incubation regions capable of triggering large-scale diffusion.

k -complex contagion. The k -complex contagion model captures reinforcement dynamics by requiring that a node observe at least k active neighbors before adopting. While this mechanism formalizes the notion of social reinforcement, it imposes strict structural constraints on diffusion. When $k = 1$, the model reduces to a simple contagion and virality becomes trivial, as a single exposure is sufficient for activation. For $k > 1$, however, diffusion cannot propagate from a single seed by definition, since no neighboring node can accumulate the required k active contacts. As a result, the model requires either multiple simultaneous seeds or highly clustered initial conditions for diffusion to begin. In core-periphery networks this restriction makes spreading from peripheral regions particularly difficult, as peripheral nodes rarely participate in sufficiently dense local structures. In contrast, the unified propagation model allows propagation affinity in the feature space to complement structural reinforcement, enabling cascades to incubate locally and later expand through structurally advantageous pathways. This interaction between feature-driven propagation and network structure allows our model to capture tipping-point behavior that is not accessible in standard k -complex contagion formulations.

5.7.2 Data-Driven Propagation Models

While the main body of the paper focuses on analyzing propagation dynamics through controlled simulations, an important question is whether the proposed propagation framework can be learned directly from real-world cascade data. In this appendix, we explore a preliminary data-driven formulation of the unified propagation model using cascade traces from real networks. Our goal is not to build a state-of-the-art prediction model, but rather to investigate whether the propagation and activation components of the model can be estimated from empirical diffusion data and whether the resulting learned model captures meaningful propagation patterns.

5.7.3 Choice of Propagation Function

To learn propagation dynamics from observed cascade traces, we parameterize the propagation function using a simple threshold-based architecture inspired by the Linear Threshold (LT) model.

$$P(v|\text{Nbr}(v)) = \sigma\left(\sum_{w \in \text{Nbr}_A(v)} I_{vw} - \sum_{w \in \text{Nbr}_I(v)} I_{vw} + b_v\right), \quad (5.15)$$

$$0 \leq I_{vw}, b_v \leq 0.1 \quad (5.16)$$

where $\text{Nbr}_A(v)$ and $\text{Nbr}_I(v)$ are the sets of active and inactive incoming neighbors of v respectively. Here $I \in [0, 0.1]^{|E|}$ and $b \in [0, 0.1]^{|V|}$ are learnable parameters of the model.

Alternately, we can let the influence weights and biases be positive and otherwise be unconstrained. The initialization of the parameters is done using "small" Gaussian weights. Since the sigmoid function saturates quickly, i.e., $\sigma(5) > 0.99$, it is advisable to choose mean aggregation of edge influences rather than sum aggregation as below,

$$P(v|\text{Nbr}(v)) = \sigma\left(\frac{\sum_{w \in \text{Nbr}_A(v)} I_{vw} - \sum_{w \in \text{Nbr}_I(v)} I_{vw}}{d_v} + b_v\right), \quad (5.17)$$

$$0 \leq I_{vw}, b_v \quad (5.18)$$

where d_v denotes the degree of node v .

While other model architectures are possible for learning propagation dynamics from data, we adopt this formulation because it provides the closest analog to the classical Linear Threshold (LT) model. In particular, the propagation function plays a role analogous to the influence weights in LT models, while the learned activation thresholds capture heterogeneous susceptibility across nodes. This choice therefore provides a natural bridge between traditional threshold-based contagion models and data-driven learning approaches. At the same time, more expressive architectures—such as graph neural networks or attention-based propagation models—could be used to parameterize the propagation function. Our goal here, however, is to retain interpretability and maintain a clear connection with the theoretical contagion models studied in the main paper.

5.7.4 Maximum Likelihood Formulation

From Eq. 4 above, we can compute the local conditional probability $P(v|Nbr(v))$, and so compute the likelihood of a cascade C , which is a directed subgraph of the global graph G as,

$$P(C|G, I, b) = \prod_{v \in C} P(v|Nbr(v)) \prod_{v \in B_G(C)} (1 - P(v|Nbr(v))) \quad (5.19)$$

where $B_G(C)$ is the boundary of the cascade C in graph G , that is, the nodes in $G \setminus C$ having at least one neighbor in C .

We then formulate our loss function as the negative log-likelihood of all the cascades and optimize for the influence parameters I and b . Here, it is important to set the weighting ratio for the in-cascade nodes and the boundary nodes in the loss appropriately,

since $|B_G(C)| \gg |C|$ generally. We assign equal weight to both in-cascade nodes and the boundary nodes.

It is also possible to differentially weigh nodes in the loss, since the early stage spreaders that're close to the seed are more important for spread prediction. Thus, we can consider subcascades within the cascade to augment our data. This can be easily implemented within the loss function.

5.7.5 Prediction of Activation States

A key question in learning propagation models from data is whether the learned model can correctly predict the intermediate activation states that arise during cascade evolution. Table 5.2 reports the accuracy of predicting node activation states across different node categories. We observe that the predictive accuracy for boundary nodes is consistently lower than for active non-seed nodes that lie fully inside cascade regions.

From our deep dives into the training process we hypothesize that this behavior is due to the presence of overlapping cascades in real-world diffusion data. In datasets such as Epinions, Ciao, Twitter and APS, multiple cascades propagate simultaneously and interact with the same nodes over time. As a result, a node that lies on the boundary of one cascade may already be active due to another cascade. This makes it difficult for a single learned propagation model to accurately capture the activation behavior using deterministic thresholds and influence weights. In effect, the activation labels observed in the data may reflect the superposition of multiple propagation processes.

One possible way to address this issue would be to train models using soft labels that represent activation probabilities rather than binary states. Such labels could capture uncertainty arising from overlapping cascades and competing diffusion processes. However, learning reliable soft-label representations requires substantially larger datasets

with dense temporal observations of cascade activity. In the current setting, the available data is insufficient to reliably estimate these distributions. This creates a chicken-and-egg problem: richer data would allow us to learn more expressive propagation models, but generating such data would itself require accurate generative models of cascade dynamics.

Overall, these results highlight the challenges of learning propagation dynamics directly from noisy cascade datasets and motivate the use of generative propagation models that allow controlled study of contagion mechanisms.

Table 5.2: Accuracy on prediction of activation state for active non-seeds and boundary nodes.

Dataset	Split	Active NonSeeds	Boundary
Epinions			
	Train	0.93	0.70
	Test	0.62	0.70
Ciao			
	Train	0.91	0.71
	Test	0.66	0.68
Twitter			
	Train	0.64	0.77
	Test	0.54	0.76
APS			
	Train	0.46	0.81
	Test	0.34	0.79

5.7.6 Limitations of Prior Data-Driven Propagation Models

Recent work on data-driven cascade modeling has focused primarily on predicting aggregate cascade outcomes, such as the final spread or popularity of a piece of content [15, 12, 13]. These models are typically trained to predict the eventual cascade size or influence spread and are often used in downstream optimization tasks such as influence maximization, where the learned models are reverse differentiated to identify optimal seed nodes.

While these approaches have achieved strong performance on spread prediction benchmarks, they do not explicitly model the intermediate activation states that define the cascade trace itself. In particular, the temporal and structural evolution of cascades—namely, which nodes activate at which stages and how propagation unfolds through the network—is not directly predicted by these models.

However, the structure of cascade traces contains important information about the underlying diffusion dynamics. As illustrated by the qualitative visualizations of large cascade traces in the Epinions and Ciao datasets 5.15, the shapes of these cascades often exhibit rich structural patterns such as branching, incubation regions, and bursts of rapid expansion. Capturing these structural patterns requires models that explicitly represent the propagation process rather than only predicting final cascade sizes.

Our analysis therefore suggests that evaluation metrics for learned propagation models should incorporate structural similarity between cascade traces. In particular, graph edit distance or related structural metrics could provide a principled way to measure how closely a generated cascade matches the topology of observed diffusion traces. Such metrics would encourage models to reproduce not only the final spread but also the intermediate propagation dynamics that give rise to complex contagion behavior.

These observations further motivate the propagation-driven generative framework

proposed in the main paper, which enables controlled exploration of cascade dynamics and phase transitions that are difficult to isolate in real-world diffusion datasets.

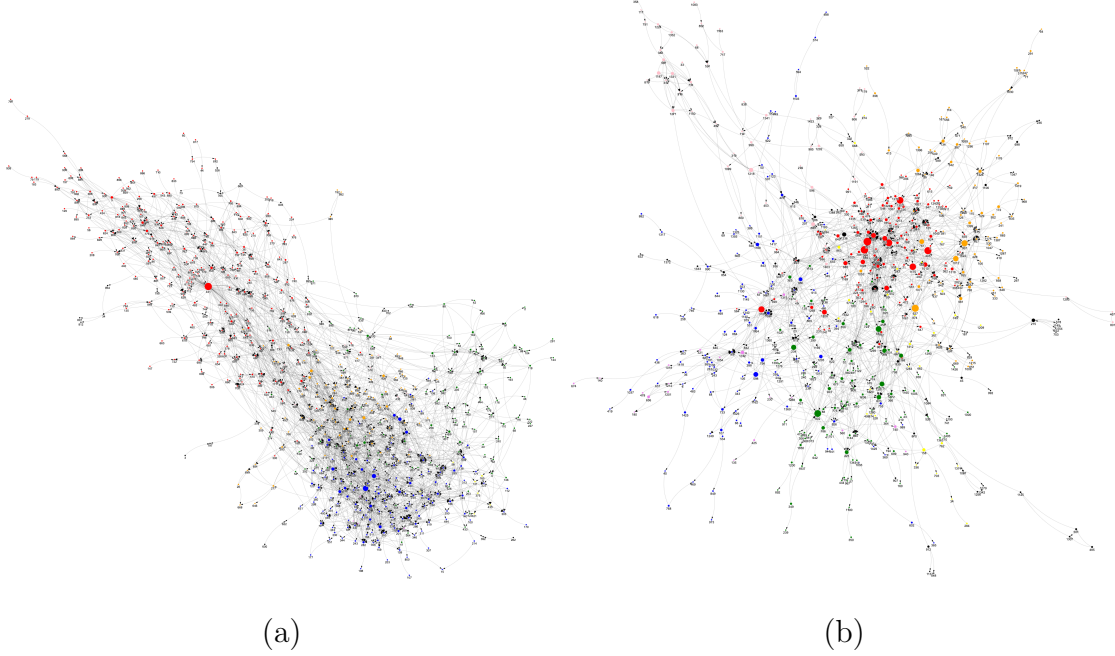


Figure 5.15: Qualitative visualization of big cascades on the (a) Ciao and (b) Epinions datasets.

5.7.7 Extended Propagation Model

In the main formulation of the unified propagation model, node features remain fixed throughout the cascade process, while the propagation vector evolves over time through network interactions. In many real-world settings, however, activation itself changes the latent state of a node. For instance, once a user adopts an opinion, engages with a product, or participates in a social behavior, subsequent susceptibility and influence may shift accordingly. To capture this effect, we consider an extension of the model in which node features are updated after activation.

Let $\mathbf{x}_v(t) \in \mathbb{R}^d$ denote the feature vector of node v at time t , and let $\hat{C} \in \mathbb{R}^d$ denote

the propagation vector at time t . Suppose node v activates at time t . We first define the unnormalized post-activation update

$$\tilde{\mathbf{x}}_v(t+1) = (1 - \lambda)\mathbf{x}_v(t) + \lambda\hat{C}, \quad (5.20)$$

where $\lambda \in [0, 1]$ is a tunable adaptation parameter. Since the feature vectors in our model are constrained to lie on the d -dimensional hypersphere, we then renormalize:

$$\mathbf{x}_v(t+1) = \frac{\tilde{\mathbf{x}}_v(t+1)}{\|\tilde{\mathbf{x}}_v(t+1)\|_2}. \quad (5.21)$$

If node v does not activate at time t , we leave its feature unchanged:

$$\mathbf{x}_v(t+1) = \mathbf{x}_v(t). \quad (5.22)$$

The parameter λ controls the extent to which activation shifts the latent state of a node toward the propagated notion. When $\lambda = 0$, the model reduces to the static-feature setting studied in the main paper. Larger values of λ produce stronger post-activation adaptation, moving the node more aggressively toward $\hat{C}$. Because of the normalization step in (5.21), the updated feature remains a point on the unit hypersphere, preserving the geometric interpretation of feature similarity used throughout the model.

This update rule induces a coupled propagation–adaptation process. Since future activations depend on node features through propagation affinity, and node features themselves evolve in response to past activations, the network state becomes path-dependent over time. In particular, repeated cascades can reshape the feature geometry of the network, altering which future propagations are likely to succeed. This provides a natural mechanism for modeling *concept drift* in evolving networks: the effective distribution of node states changes as a consequence of prior contagion events.

An important consequence of feature drift is that the network edge weights also evolve over time. In our setting, edge weights are derived from cosine similarity between node

features. Hence, if

$$w_{uv}(t) = \cos(\mathbf{x}_u(t), \mathbf{x}_v(t)) = \mathbf{x}_u(t)^\top \mathbf{x}_v(t), \quad (5.23)$$

where the second equality uses the fact that the features are normalized, then any activation-driven update to $\mathbf{x}_u(t)$ or $\mathbf{x}_v(t)$ changes the corresponding edge weight. As a result, repeated cascades do not merely alter node susceptibilities; they also modify the effective geometry of the network itself. *This yields a dynamic weighted graph in which both node states and edge weights co-evolve over time.*

This extension therefore enriches the unified propagation model in two ways. First, it allows node states to evolve endogenously as a result of activation, rather than remaining fixed throughout the process. Second, it provides a principled explanation for how repeated cascades can induce concept drift in the network over time, thereby changing both the feature geometry and the edge-weight structure under which future cascades incubate, break out, and become viral.

5.7.8 Sampling Prop. Vectors to Maximize Spread

We now consider the inverse problem associated with the unified propagation model: given a seed node and a network, can we identify or sample a propagation vector that is likely to induce maximal spread? Let $G = (V, E)$ be a graph with adjacency matrix A , node feature matrix $X \in \mathbb{R}^{n \times d}$, and a single seed node $v \in V$. The goal is to produce a propagation vector $\hat{C} \in \mathbb{R}^d$ that maximizes the expected cascade spread under the stochastic dynamics of the model.

Let $\sigma(\hat{C}; v, A, X)$ denote the expected spread size of a cascade generated by initializing the unified propagation model with seed node v and propagation vector $\hat{C}$. Since propagation vectors are normalized in our framework, we restrict the feasible set to the

unit hypersphere. The resulting optimization problem is

$$\hat{C}^\star = \arg \max_{\|\hat{C}\|_2=1} \sigma(\hat{C}; v, A, X). \quad (5.24)$$

This optimization problem is generally intractable. The search space is continuous and high-dimensional, and evaluating $\sigma(\hat{C})$ requires MCMC simulation of the stochastic cascade process. Furthermore, the objective function is highly nonconvex because small changes in $\hat{C}$ can induce activation threshold crossings that significantly alter cascade trajectories. We therefore consider approximate sampling-based strategies for identifying high-quality propagation vectors.

5.7.9 Candidate Generation

A natural heuristic is to restrict the search to propagation vectors aligned with node features that lie along structurally favorable diffusion pathways. In preferential attachment networks and other core-periphery structures, viral cascades often occur when propagation reaches highly connected core nodes or traverses wide bridges between communities.

Let $\mathcal{N}_K(v)$ denote a candidate node pool constructed from

1. the seed node v ,
2. nodes within K hops of v ,
3. high-degree nodes reachable from v , and
4. nodes lying on short paths from v to the network core.

From this pool, we construct candidate propagation vectors by normalizing node feature vectors or local feature averages:

$$\hat{C}_u = \frac{\mathbf{x}_u}{\|\mathbf{x}_u\|_2}, \quad \hat{C}_u = \frac{\mathbf{x}_u + \sum_{w \in N(u)} \mathbf{x}_w}{\left\| \mathbf{x}_u + \sum_{w \in N(u)} \mathbf{x}_w \right\|_2}, \quad (5.25)$$

for $u \in \mathcal{N}_K(v)$.

This reduces the search space from the continuous hypersphere to a finite set of feature-informed directions that likely align with the feature geometry of nodes capable of sustaining propagation.

5.7.10 Monte Carlo Estimation of Spread

For a candidate propagation vector $\hat{C}$, we estimate the expected spread by repeated cascade simulations. Let $\sigma^{(m)}(\hat{C})$ denote the spread size observed in the m -th cascade simulation. The empirical estimate of expected spread is

$$\hat{\sigma}(\hat{C}) = \frac{1}{M} \sum_{m=1}^M \sigma^{(m)}(\hat{C}). \quad (5.26)$$

Candidate propagation vectors can be ranked according to $\hat{\sigma}(\hat{C})$.

5.7.11 Beam Search Refinement

To improve over simple enumeration, we perform local refinement around high-scoring propagation vectors. Starting from a candidate $\hat{C}$, we generate perturbed candidates of the form

$$\hat{C}' = \frac{\hat{C} + \epsilon \mathbf{z}}{\|\hat{C} + \epsilon \mathbf{z}\|_2}, \quad (5.27)$$

where $\mathbf{z}$ is sampled from an isotropic noise distribution and ϵ is a small perturbation parameter. Each perturbed vector is evaluated using the Monte Carlo estimator in (5.26), and the top B candidates are retained for further refinement. Repeating this process for T rounds yields a beam-search approximation over the hypersphere of propagation vectors.

5.7.12 Dynamic Programming Approximation

Although exact dynamic programming over propagation vectors is infeasible, a coarse approximation can be constructed by discretizing the propagation space into a finite codebook $\{\hat{C}_1, \dots, \hat{C}_R\}$ and grouping nodes into structural states such as *core*, *intermediate*, and *periphery*.

Let $V_t(r, s)$ denote the estimated future spread obtainable at time t when the propagation vector corresponds to codebook entry r and the current cascade frontier lies primarily in structural region s . An approximate Bellman recursion can then be written as

$$V_t(r, s) = \max_{r' \in [R]} \left\{ \text{Reward}(r, s, r') + \sum_{s'} P(s' | s, r, r') V_{t+1}(r', s') \right\}, \quad (5.28)$$

where $\text{Reward}(r, s, r')$ estimates the immediate increase in spread and $P(s' | s, r, r')$ represents transition probabilities between structural regions estimated from simulation.

5.7.13 Approximate Algorithm

In practice, the following approximate procedure can be used:

1. Construct candidate propagation vectors from the features of nodes near the seed v and along short paths to core nodes.
2. Estimate the expected spread $\hat{\sigma}(\hat{C})$ for each candidate using Monte Carlo cascade simulations.
3. Refine the highest-scoring candidates using beam search with normalized perturbations.
4. Use the coarse dynamic programming approximation to prioritize candidates likely to move the cascade toward structurally advantageous regions of the graph.

This procedure yields an approximate solver

$$\text{SAMPLEPROPAGATIONVECTOR}(v, A, X) \rightarrow \hat{C}, \quad (5.29)$$

which outputs a normalized propagation vector predicted to achieve large expected spread. By combining structural heuristics with Monte Carlo evaluation, the algorithm exploits the key mechanisms of the unified propagation model: propagation affinity in feature space and structural pathways that enable incubation and virality.

Chapter 6

Conclusion

This dissertation examined models of propagation in real-world networks, with a particular emphasis on network perturbations for accelerating spread and representation-based propagations that unify local and global influences. Across three works, we addressed complementary aspects of this problem, ranging from classifier learning under imbalance to data-driven modeling of propagation and stochastic generative processes for complex contagion.

In the first work, we proposed Prototype classifiers to address the biased softmax problem for long-tailed recognition. Prototype classifiers do not suffer from the bias of correlating classifier norm to class size, as they rely on distances to learnable prototypes in representation space rather than dot-products with learnable weights, which couple weight and bias terms. Our theoretical analysis showed that Euclidean distance is stable for gradient-based optimization, and this was confirmed empirically. We further enabled independent distance scales in the prototype distance function using channel-dependent temperatures. Experimental results on four benchmark long-tailed datasets showed that the Prototype classifier outperforms or is comparable to prior state-of-the-art methods. Moreover, the learned prototypes were equinorm and exhibited improved separation in

representation space compared to centroids.

In the second work, we introduced the novel Dynamic Viral Marketing (DVM) problem, which seeks to find the minimum budget and minimal perturbation set required to achieve a target spread, where the propagation model is a non-linear GNN and perturbations are constrained by referral and co-marketing mechanisms. We showed that DVM is NP-hard and is closely related to influence maximization. To address this problem, we developed the Dynamic Gradient Influencing (DGI) framework, which targets non-adopters with low budget and high influence. DGI uses gradient-based ranking to order perturbations and incorporates an efficient budget computation method, a Meta Influence heuristic, and Meta Attribute Flips to increase node influence. We evaluated DGI on three real-world attributed networks and demonstrated average improvements of 24% in budget efficiency and 37% in AUC over multiple baselines. We also validated the effectiveness of the budget computation and Meta Influence heuristic, and analyzed cascade patterns through intermediary adopter nodes identified by DGI.

This work opens up a new research direction: data-driven models for network propagation as alternatives to postulated models such as Linear Threshold [26]. Data-driven approaches can incorporate node attributes and long-range interactions, enabling more expressive propagation models. This perspective motivates future directions including model-based reinforcement learning [97] in unknown environments and the development of competitive strategies among interacting groups seeking to maximize spread.

In the third work, we presented a propagation-driven framework for modeling complex contagion in networks that explicitly integrates feature-based affinity, local reinforcement, and global influence into a unified stochastic activation process. By representing propagating notions in the same feature space as network nodes, the model captures heterogeneity in susceptibility and interprets propagation as a gated, time-varying random walk. Through simulations on preferential attachment networks, we demonstrated

how local incubation, core–periphery pathways, and parameter interactions jointly shape cascade formation.

Our results showed that both excessive local reinforcement and excessive global influence can inhibit diffusion, and that successful cascades arise from a balance between incubation and structural exploration. We further observed that phase transitions and tipping points are reflected in early-stage growth dynamics, providing opportunities for early prediction of viral outbreaks. By offering a controlled and interpretable generative framework, this model serves as a complementary tool for understanding propagation mechanisms that are often difficult to isolate in empirical data.

Taken together, these works contribute to a broader understanding of propagation in networked systems. They highlight the importance of representation, temporal, and modeling assumptions in shaping both learning outcomes and propagation dynamics. The results demonstrate that both data-driven approaches and principled modeling frameworks are essential for capturing the complexity of real-world propagation processes.

Acknowledgments

Most of the work in this thesis is based upon work supported by the National Science Foundation under grant no. 2229876 and is supported in part by funds provided by the National Science Foundation, by the Department of Homeland Security, and by IBM.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or its federal agency and industry partners.

Bibliography

- [1] D. J. Watts and S. H. Strogatz, *Collective dynamics of ‘small-world’ networks*, *nature* **393** (1998), no. 6684 440–442.
- [2] D. J. Watts, *A simple model of global cascades on random networks*, *Proceedings of the National Academy of Sciences* **99** (2002), no. 9 5766–5771.
- [3] A.-L. Barabási and R. Albert, *Emergence of scaling in random networks*, *science* **286** (1999), no. 5439 509–512.
- [4] M. Newman, *Networks*. Oxford university press, 2018.
- [5] J. Kleinberg, *Navigation in a small world*, *Nature* **406** (2000), no. 6798 845.
- [6] J. Kleinberg, *Networks, Crowds, and Markets*. Cambridge University Press, 2010.
- [7] Jon Kleinberg, ITA, *Cascading behavior in complex networks*, 2010.
<https://www.youtube.com/watch?v=GBuekJkQR3o>,.
- [8] J. Gao, G. Ghasemiesfeh, G. Schoenebeck, and F.-Y. Yu, *General threshold model for social cascades: Analysis and simulations*, in *Proceedings of the 2016 ACM Conference on Economics and Computation*, pp. 617–634, 2016.
- [9] G. Ghasemiesfeh, R. Ebrahimi, and J. Gao, *Complex contagion and the weakness of long ties in social networks: revisited*, in *Proceedings of the fourteenth ACM conference on Electronic Commerce*, pp. 507–524, 2013.
- [10] D. Centola and M. Macy, *Complex contagions and the weakness of long ties*, *American journal of Sociology* **113** (2007), no. 3 702–734.
- [11] D. Centola, *Influencers, backfire effects, and the power of the periphery*, *Personal Networks: Classic Readings and New Directions in Egocentric Analysis* **51** (2021) 73.
- [12] C. Ling, J. Jiang, J. Wang, M. T. Thai, R. Xue, J. Song, M. Qiu, and L. Zhao, *Deep graph representation learning and optimization for influence maximization*, in *International conference on machine learning*, pp. 21350–21361, PMLR, 2023.

- [13] S. Kumar, A. Mallik, A. Khetarpal, and B. S. Panda, *Influence maximization in social networks using graph embedding and graph neural network*, *Information Sciences* **607** (2022) 1617–1636.
- [14] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, *A data-based approach to social influence maximization*, *Proc. VLDB Endow.* **5** (2011), no. 1 73–84.
- [15] X. Xu, F. Zhou, K. Zhang, S. Liu, and G. Trajcevski, *Casflow: Exploring hierarchical structures and propagation uncertainty for cascade prediction*, *IEEE Transactions on Knowledge and Data Engineering* **35** (2021), no. 4 3484–3499.
- [16] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, in *International Conference on Learning Representations*, 2016.
- [17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.
- [18] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, *Simplifying graph convolutional networks*, in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871, PMLR, 2019.
- [19] J. Tang, H. Gao, and H. Liu, *mtrust: Discerning multi-faceted trust in a connected world*, in *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 93–102, 2012.
- [20] M. Richardson and P. M. Domingos, *Mining knowledge-sharing sites for viral marketing*, in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pp. 61–70, ACM, 2002.
- [21] J. Leskovec, A. Singh, and J. Kleinberg, *Patterns of influence in a recommendation network*, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 380–389, Springer, 2006.
- [22] M. Andre, K. Ijaz, J. D. Tillinghast, V. E. Krebs, L. A. Diem, B. Metchock, T. Crisp, and P. D. McElroy, *Transmission network analysis to complement routine tuberculosis contact investigations*, *American journal of public health* **97** (2007), no. 3 470–477.
- [23] P. Erdős, A. Rényi, *et. al.*, *On the evolution of random graphs*, *Publications of the* (1960).

- [24] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, *The inaturalist species classification and detection dataset*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.
- [25] J. Travers and S. Milgram, *An experimental study of the small world problem*, in *Social networks*, pp. 179–197. Elsevier, 1977.
- [26] M. Granovetter, *Threshold models of collective behavior*, *American journal of sociology* **83** (1978), no. 6 1420–1443.
- [27] D. Kempe, J. Kleinberg, and É. Tardos, *Maximizing the spread of influence through a social network*, in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, 2003.
- [28] J. Goldenberg, B. Libai, and E. Muller, *Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata*, *Academy of Marketing Science Review* **9** (2001), no. 3 1–18.
- [29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, *nature* **323** (1986), no. 6088 533–536.
- [30] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, *Class-balanced loss based on effective number of samples*, in *CVPR*, 2019.
- [31] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, *Decoupling representation and classifier for long-tailed recognition*, in *Eighth International Conference on Learning Representations (ICLR)*, 2020.
- [32] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, *Large-scale long-tailed recognition in an open world*, in *CVPR*, 2019.
- [33] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, *arXiv preprint arXiv:1412.6572* (2014).
- [34] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains*, *IEEE signal processing magazine* **30** (2013), no. 3 83–98.
- [35] W. Hamilton, Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs*, *Advances in neural information processing systems* **30** (2017).
- [36] D. Chen, Y. Lin, G. Zhao, X. Ren, P. Li, J. Zhou, and X. Sun, *Topology-imbalance learning for semi-supervised node classification*, *Advances in Neural Information Processing Systems* **34** (2021) 29885–29897.

- [37] K. Ding, J. Li, R. Bhanushali, and H. Liu, *Deep anomaly detection on attributed networks*, in *Proceedings of the 2019 SIAM international conference on data mining*, pp. 594–602, SIAM, 2019.
- [38] D. Zügner, A. Akbarnejad, and S. Günnemann, *Adversarial attacks on neural networks for graph data*, in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2847–2856, 2018.
- [39] G. Van Horn and P. Perona, *The devil is in the tails: Fine-grained classification in the wild*, *arXiv preprint arXiv:1709.01450* (2017).
- [40] A. Estabrooks, T. Jo, and N. Japkowicz, *A multiple resampling method for learning from imbalanced data sets*, *Computational intelligence* (2004).
- [41] H. He and E. A. Garcia, *Learning from imbalanced data*, *TKDE* (2009).
- [42] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, *Learning imbalanced datasets with label-distribution-aware margin loss*, in *NeurIPS*, 2019.
- [43] J. Ren, C. Yu, X. Ma, H. Zhao, S. Yi, *et. al.*, *Balanced meta-softmax for long-tailed visual recognition*, *Advances in Neural Information Processing Systems* **33** (2020) 4175–4186.
- [44] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar, *Long-tail learning via logit adjustment*, in *International Conference on Learning Representations*, 2021.
- [45] D. Samuel and G. Chechik, *Distributional robustness loss for long-tail learning*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.
- [46] L. Xiang, G. Ding, and J. Han, *Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification*, in *European Conference on Computer Vision*, Springer, 2020.
- [47] X. Wang, L. Lian, Z. Miao, Z. Liu, and S. Yu, *Long-tailed recognition by routing diverse distribution-aware experts*, in *International Conference on Learning Representations*, 2020.
- [48] B. Zhou, Q. Cui, X.-S. Wei, and Z.-M. Chen, *Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition*, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9719–9728, 2020.
- [49] B. Liu, H. Li, H. Kang, G. Hua, and N. Vasconcelos, *Gistnet: a geometric structure transfer network for long-tailed recognition*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8209–8218, 2021.

- [50] S. Alshammari, Y.-X. Wang, D. Ramanan, and S. Kong, *Long-tailed recognition via weight balancing*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6897–6907, 2022.
- [51] V. Pappas, X. Han, and D. L. Donoho, *Prevalence of neural collapse during the terminal phase of deep learning training*, *Proceedings of the National Academy of Sciences* **117** (2020), no. 40 24652–24663.
- [52] C. Thrampoulidis, G. R. Kini, V. Vakilian, and T. Behnia, *Imbalance trouble: Revisiting neural-collapse geometry*, in *Advances in Neural Information Processing Systems*, 2022.
- [53] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, *Distance-based image classification: Generalizing to new classes at near-zero cost*, *TPAMI* (2013).
- [54] J. Snell, K. Swersky, and R. Zemel, *Prototypical networks for few-shot learning*, in *NeurIPS*, 2017.
- [55] S. Guerriero, B. Caputo, and T. Mensink, *Deepncm: Deep nearest class mean classifiers*, in *International Conference on Learning Representations Workshop*, 2018.
- [56] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, *icarl: Incremental classifier and representation learning*, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.
- [57] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, *Smote: synthetic minority over-sampling technique*, *JAIR* (2002).
- [58] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, *Smoteboost: Improving prediction of the minority class in boosting*, in *European conference on principles of data mining and knowledge discovery*, 2003.
- [59] C. Drummond, R. C. Holte, et. al., *C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling*, in *Workshop on learning from imbalanced datasets II*, 2003.
- [60] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, *Meta-transfer learning for few-shot learning*, in *CVPR*, 2019.
- [61] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, *f-vaegan-d2: A feature generating framework for any-shot learning*, in *CVPR*, 2019.
- [62] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, *Focal loss for dense object detection*, in *ICCV*, 2017.

- [63] S. Sharma, N. Yu, M. Fritz, and B. Schiele, *Long-tailed recognition using class-balanced experts*, in *DAGM German Conference on Pattern Recognition*, Springer, 2020.
- [64] J. Cui, Z. Zhong, S. Liu, B. Yu, and J. Jia, *Parametric contrastive learning*, in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 715–724, 2021.
- [65] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in *CVPR*, 2009.
- [66] I. Loshchilov and F. Hutter, *Sgdr: Stochastic gradient descent with warm restarts*, in *International Conference on Learning Representations*, 2017.
- [67] F. Nogueira et. al., *Bayesian optimization: Open source constrained global optimization tool for python*, URL <https://github.com/fmfn/BayesianOptimization> (2014).
- [68] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, *Autoaugment: Learning augmentation policies from data*, *arXiv preprint arXiv:1805.09501* (2018).
- [69] T. DeVries and G. W. Taylor, *Improved regularization of convolutional neural networks with cutout*, *arXiv preprint arXiv:1708.04552* (2017).
- [70] C. Fang, H. He, Q. Long, and W. J. Su, *Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training*, *Proceedings of the National Academy of Sciences* **118** (2021), no. 43 e2103091118.
- [71] S. Zhang, Z. Li, S. Yan, X. He, and J. Sun, *Distribution alignment: A unified framework for long-tail visual recognition*, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2361–2370, 2021.
- [72] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, *Randaugment: Practical automated data augmentation with a reduced search space*, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.
- [73] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, *Momentum contrast for unsupervised visual representation learning*, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.
- [74] P. M. Domingos and M. Richardson, *Mining the network value of customers*, in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001* (D. Lee, M. Schkolnick, F. J. Provost, and R. Srikant, eds.), pp. 57–66, ACM, 2001.

- [75] S. Morris, *Contagion*, *The Review of Economic Studies* **67** (2000), no. 1 57–78.
- [76] D. Kempe, J. M. Kleinberg, and É. Tardos, *Maximizing the spread of influence through a social network*, *Theory Comput.* **11** (2015) 105–147.
- [77] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering*, *Advances in neural information processing systems* **29** (2016).
- [78] F. A. Buttle, *Word of mouth: understanding and managing referral marketing*, *Journal of strategic marketing* **6** (1998), no. 3 241–254.
- [79] Ironclad Journal, “Joint marketing agreements.” <https://ironcladapp.com/journal/contracts/joint-marketing-agreement/>, 2024. Accessed: 2024-08-29.
- [80] Y. Wang and Y. Wang, *Opinion-aware influence maximization in online social networks*, in *6th International Conference on Data Science and Information Technology, DSIT 2023, Shanghai, China, July 28-30, 2023*, pp. 214–221, IEEE, 2023.
- [81] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, *A comprehensive survey on graph neural networks*, *IEEE Trans. Neural Networks Learn. Syst.* **32** (2021), no. 1 4–24.
- [82] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, *Adversarial attack on graph structured data*, in *International conference on machine learning*, pp. 1115–1124, PMLR, 2018.
- [83] D. Zügner and S. Günnemann, *Adversarial attacks on graph neural networks via meta learning*, in *International Conference on Learning Representations (ICLR)*, 2019.
- [84] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, *Topology attack and defense for graph neural networks: an optimization perspective*, in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3961–3967, 2019.
- [85] M. Zhang, X. Wang, C. Shi, L. Lyu, T. Yang, and J. Du, *Minimum topology attacks for graph neural networks*, in *Proceedings of the ACM Web Conference 2023*, pp. 630–640, 2023.
- [86] S. Manchanda, A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. K. Singh, *GCOMB: learning budget-constrained combinatorial algorithms over billion-sized graphs*, in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*,

- December 6-12, 2020, virtual (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.
- [87] S. Kumar, A. Mallik, A. Khetarpal, and B. S. Panda, *Influence maximization in social networks using graph embedding and graph neural network*, *Inf. Sci.* **607** (2022) 1617–1636.
 - [88] J. Klicpera, A. Bojchevski, and S. Günnemann, *Predict then propagate: Graph neural networks meet personalized pagerank*, in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.
 - [89] H. Wang and J. Leskovec, *Unifying graph convolutional neural networks and label propagation*, *CoRR* **abs/2002.06755** (2020) [arXiv:2002.06755].
 - [90] S. Geisler, T. Schmidt, H. Şirin, D. Zügner, A. Bojchevski, and S. Günnemann, *Robustness of graph neural networks at scale*, *Advances in Neural Information Processing Systems* **34** (2021) 7637–7649.
 - [91] F. Mujkanovic, S. Geisler, S. Günnemann, and A. Bojchevski, *Are defenses for graph neural networks robust?*, *Advances in Neural Information Processing Systems* **35** (2022) 8954–8968.
 - [92] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
 - [93] P. Tabacof and E. Valle, *Exploring the space of adversarial images*, in *2016 international joint conference on neural networks (IJCNN)*, pp. 426–433, IEEE, 2016.
 - [94] N. Carlini and D. Wagner, *Towards evaluating the robustness of neural networks*, in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, Ieee, 2017.
 - [95] J. Tang, H. Gao, and H. Liu, *mtrust: discerning multi-faceted trust in a connected world*, in *Proceedings of the Fifth International Conference on Web Search and Web Data Mining, WSDM 2012, Seattle, WA, USA, February 8-12, 2012* (E. Adar, J. Teevan, E. Agichtein, and Y. Maarek, eds.), pp. 93–102, ACM, 2012.
 - [96] M. Jamali and M. Ester, *A matrix factorization technique with trust propagation for recommendation in social networks*, in *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010* (X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, eds.), pp. 135–142, ACM, 2010.
 - [97] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, et. al., *Model-based reinforcement learning: A survey*, *Foundations and Trends® in Machine Learning* **16** (2023), no. 1 1–118.

- [98] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, *Efficient neural network robustness certification with general activation functions*, *Advances in neural information processing systems* **31** (2018).
- [99] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, *Reluplex: An efficient smt solver for verifying deep neural networks*, in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I* 30, pp. 97–117, Springer, 2017.
- [100] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, *Measuring and relieving the over-smoothing problem for graph neural networks from the topological view*, in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, pp. 3438–3445, AAAI Press, 2020.
- [101] Y. Li, W. Jin, H. Xu, and J. Tang, *Deepprobust: A pytorch library for adversarial attacks and defenses*, *arXiv preprint arXiv:2005.06149* (2020).
- [102] D. Zügner, O. Borchert, A. Akbarnejad, and S. Günnemann, *Adversarial attacks on graph neural networks: Perturbations and their patterns*, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **14** (2020), no. 5 1–31.
- [103] R. Ebrahimi, J. Gao, G. Ghasemiesfeh, and G. Schoenbeck, *How complex contagions spread quickly in preferential attachment models and other time-evolving networks*, *IEEE Transactions on Network Science and Engineering* **4** (2017), no. 4 201–214.
- [104] D. Guilbeault, J. Becker, and D. Centola, *Complex contagions: A decade in review*, *Complex spreading phenomena in social systems: Influence and contagion in real-world social networks* (2018) 3–25.
- [105] J. Shrager, T. Hogg, and B. A. Huberman, *Observation of phase transitions in spreading activation networks*, *Science* **236** (1987), no. 4805 1092–1094.
- [106] M. Scheffer, S. R. Carpenter, T. M. Lenton, J. Bascompte, W. Brock, V. Dakos, J. Van de Koppel, I. A. Van de Leemput, S. A. Levin, E. H. Van Nes, *et. al.*, *Anticipating critical transitions*, *science* **338** (2012), no. 6105 344–348.
- [107] M. Galesic, H. Olsson, J. Dalege, T. Van Der Does, and D. L. Stein, *Integrating social and cognitive aspects of belief dynamics: towards a unifying framework*, *Journal of the Royal Society Interface* **18** (2021), no. 176.
- [108] J. Gaitonde, J. Kleinberg, and É. Tardos, *Polarization in geometric opinion dynamics*, in *Proceedings of the 22nd ACM Conference on Economics and Computation*, pp. 499–519, 2021.
- [109] E. M. Rogers, *Diffusion of Innovations*. Free Press, 5 ed., 2003.

- [110] J. Kleinberg *et. al.*, *Cascading behavior in networks: Algorithmic and economic issues*, *Algorithmic game theory* **24** (2007) 613–632.
- [111] R. Ebrahimi, J. Gao, G. Ghasemiefteh, and G. Schoenebeck, *Complex contagions in kleinberg’s small world model*, in *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pp. 63–72, 2015.
- [112] X. Zhu and Z. Ghahramani, *Learning from labeled and unlabeled data with label propagation*, in *Technical Report, Carnegie Mellon University*, 2002.
- [113] X. Zhu, Z. Ghahramani, and J. Lafferty, *Semi-supervised learning using gaussian fields and harmonic functions*, in *Proceedings of the 20th International Conference on Machine Learning*, 2003.
- [114] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, *Learning with local and global consistency*, *Advances in Neural Information Processing Systems* **16** (2004).
- [115] R. Kondor and J. Lafferty, *Diffusion kernels on graphs and other discrete structures*, *Proceedings of the International Conference on Machine Learning* (2002).
- [116] S. Goel, A. Anderson, J. Hofman, and D. J. Watts, *The structural virality of online diffusion*, *Management Science* **62** (2015), no. 1 180–196.
- [117] R. Pastor-Satorras and A. Vespignani, *Epidemic spreading in scale-free networks*, *Physical Review Letters* **86** (2001), no. 14 3200–3203.
- [118] D. Spielman, *Spectral graph theory*, *Combinatorial scientific computing* **18** (2012), no. 18.
- [119] J. Shi and J. Malik, *Normalized cuts and image segmentation*, *IEEE Transactions on pattern analysis and machine intelligence* **22** (2000), no. 8 888–905.
- [120] B. Perozzi, R. Al-Rfou, and S. Skiena, *Deepwalk: Online learning of social representations*, in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.
- [121] A. Grover and J. Leskovec, *node2vec: Scalable feature learning for networks*, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.
- [122] C. M. Christensen, *Exploring the limits of the technology s-curve. part i: component technologies*, *Production and operations management* **1** (1992), no. 4 334–357.

- [123] D. Easley, J. Kleinberg, *et. al.*, *Networks, crowds, and markets: Reasoning about a highly connected world*, vol. 1. Cambridge university press Cambridge, 2010.
- [124] S. Janson, D. E. Knuth, T. Łuczak, and B. Pittel, *The birth of the giant component*, *Random Structures & Algorithms* **4** (1993), no. 3 233–358.
- [125] B. Bollobás* and O. Riordan, *The diameter of a scale-free random graph*, *Combinatorica* **24** (2004), no. 1 5–34.
- [126] J. Tang, H. Gao, H. Liu, and A. Das Sarma, *etrust: Understanding trust evolution in an online world*, in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 253–261, 2012.