Towards Effective Planning Strategies for Dynamic Opinion Networks

Bharath Muppasani, Protik Nag, Vignesh Narayanan, Biplav Srivastava, and Michael N. Huhns
Department of Computer Science
University of South Carolina, USA
bharath@email.sc.edu

Abstract

In this study, we investigate the under-explored *intervention planning* aimed at disseminating accurate information within *dynamic opinion networks* by leveraging learning strategies. Intervention planning involves *identifying key nodes* (search) and *exerting control* (e.g., disseminating accurate/official information through the nodes) to mitigate the influence of misinformation. However, as network size increases, the problem becomes computationally intractable. To address this, we first introduce a novel ranking algorithm (search) to identify key nodes for disseminating accurate information, which facilitates the training of neural network (NN) classifiers for scalable and generalized solutions. Second, we address the complexity of label generation (through search) by developing a Reinforcement Learning (RL)-based dynamic planning framework. We investigate NN-based RL planners tailored for dynamic opinion networks governed by two propagation models for the framework. Each model incorporates both binary and continuous opinion and trust representations. Our experimental results demonstrate that our ranking algorithm-based classifiers provide plans that enhance infection rate control, especially with increased action budgets. Moreover, reward strategies focusing on key metrics, such as the number of susceptible nodes and infection rates, outperform those prioritizing faster blocking strategies. Additionally, our findings reveal that Graph Convolutional Networks (GCNs)-based planners facilitate scalable centralized plans that achieve lower infection rates (higher control) across various network scenarios (e.g., Watts-Strogatz topology, varying action budgets, varying initial infected nodes, and varying degree of infected nodes).

1 Introduction

The spread of information across social networks profoundly impacts public opinion, collective behaviors, and societal outcomes [1]. Especially during crises such as disease outbreaks or disasters, there is often too much information coming from different sources. Sometimes, the resultant flood of information is unreliable or misleading, or spreads too quickly, which can have serious effects on society and health [6]. Online platforms such as Facebook, Twitter, and WeChat, while essential for communication, significantly contribute to the swift spread of misinformation. This has led to public confusion and panic in events ranging from the Fukushima disaster to the COVID-19 pandemic, demonstrating the need for effective information management on these platforms. Furthermore, the intentional dissemination of fabricated news, especially noted during significant political events such as U.S. presidential elections, underscores the influence of misinformation on democratic processes and highlights the urgent requirement for interventions to mitigate these impacts [2, 9].

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

The first step in combating misinformation propagation is to detect misinformation. However, detection alone is insufficient to effectively control its spread. Thus, it must be complemented with strategic planning. While numerous studies have focused on rumor detection [44, 39, 32, 16], comprehensive strategies for controlling misinformation are essential [12]. Existing research works emphasize three primary strategies: node removal [7, 37, 24], edge removal [18, 16, 36], and counter-rumor dissemination [3, 34, 10]. Node removal involves identifying and neutralizing key nodes using community detection methods, with dynamic models that adapt to changes in propagation. Edge removal focuses on disrupting misinformation pathways by strategically severing connections between nodes. Additionally, counter-rumor dissemination promotes the spread of factual information, leveraging user participation and 'positive cascades' to counteract misinformation. Authors in [27] have formulated the information spread in networks as a search problem and generated sequential plans using Automated Planning techniques for the targeted spread of information. These strategies underscore the necessity of a multifaceted approach to effectively manage misinformation in social networks. A review of relevant literature is presented in Appendix A.1.

Authors in [30] considered mitigating misinformation by identifying potential purveyors to block their posts. However, taking strong measures like censoring user posts may violate user rights. To address this, recent advancements have employed graph convolutional networks (GCNs) for both detection and mitigation, analyzing structural and content-based data to identify and target rumor patterns and key nodes. In our paper, we employ a strategy similar to [8, 11], which aims to reduce the impact of misinformation by intervening in the spread of accurate information.

Previous studies have found ranking algorithms to be critical in identifying influential nodes within complex networks, which can then be targeted to block, remove, or cascade information to reduce the overall spread of misinformation [42, 13]. These algorithms rank nodes based on various metrics, determining their importance or influence within the network.

We address the challenge of containing misinformation spread by effectively disseminating accurate information through two learning methodologies. In our study, we propose a novel ranking algorithm integrated with a supervised learning (SL) framework to identify influential nodes using a robust feature set of network nodes and evaluate their performance. While this approach is effective in a small-scale, we also develop reinforcement learning (RL)-based solution to this problem. In both the cases, we investigate three distinct initialization settings for our network, ranging from discrete to continuous representations of opinion and mutual trust. In a network of connected agents, we initially consider a set of nodes that possess misinformation, which they propagate to their neighbors. Our goal is to counteract this by disseminating accurate information to selected agents at each time step. The agents receiving accurate information update their opinion to a level where they no longer believe the misinformation and, consequently, cease to propagate it. The process ends when no agents are left to receive the misinformation.

Example: Consider a research community discussing the NeurIPS submission deadline. In this context, a *topic* is just a statement such as 'NeurIPS submission deadline is on May 22'. An *opinion* of an agent on this topic is defined as the belief of the agent in the truthfulness of the statement. A positive (respectively negative) opinion value represents that the agent believes the statement is true (respectively false). In our study, we consider the opinion value of an agent on a topic lies in $[-1, 1]$. Figure 1 illustrates such a scenario where the red nodes represent the agents not believing

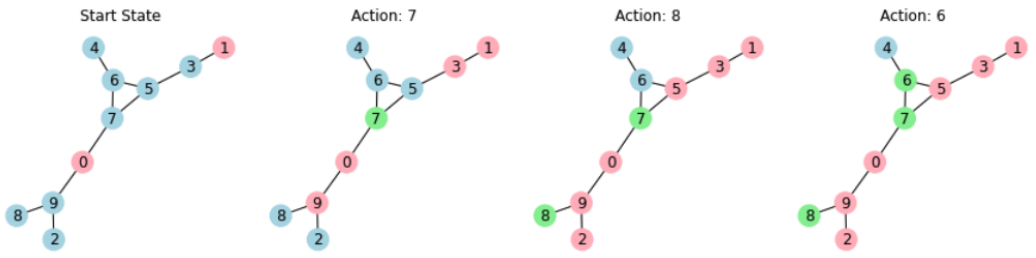

Figure 1: Example of misinformation propagation and control choices at each timestep. Blue nodes: neutral (opinion value 0), red nodes: possess misinformation (opinion value -1), green nodes: received accurate information (opinion value 1).

about the NeurIPS deadline being May 22. When they interact with other agents, they share this misinformation, leading to a spread of incorrect information throughout the network. The blue nodes represent neutral agents who are unaware of the deadline, and the green nodes represent agents who believe in the NeurIPS deadline being on May 22. Initially, agents $\{0, 1\}$ propagate misinformation to their neighbors. To control the spread of this misinformation, timely control actions are taken at each timestep to disseminate accurate information to selected agents. In the example, with an action budget of 1, agents 7, 8, and 6 are sequentially chosen to receive the accurate information, represented by the green nodes. Through these control actions, the example demonstrates how timely intervention at critical points can effectively mitigate misinformation spread, ensuring agents are accurately informed about the NeurIPS deadline. This mirrors real-world scenarios where strategic dissemination of accurate information prevents widespread misunderstanding within a community.

Our paper makes several significant **contributions** to dynamic planning for containing misinformation spread, focusing on the integration of more realistic and complex modeling approaches, label generation techniques, and training methodologies: (a) We utilize a continuous opinion scale to model the dynamics of misinformation spread, providing a more realistic representation of opinion changes over time. (b) We develop a novel ranking algorithm for generating labels in networks with discrete opinions, addressing a significant gap in efficient data preparation for SL algorithms in this domain. (c) We propose an innovative RL methodology. This allows for adaptive intervention strategies in response to evolving misinformation spread patterns, a critical improvement over traditional static approaches. (d) Utilizing GCNs with an enhanced feature set of opinion value, connectivity degree, and proximity to a misinformed node, we improve the training of models for selecting effective intervention strategies. This enhancement ensures scalability and generalizes well across various network structures, demonstrating the robust capabilities of GCNs in complex scenarios.

2 Problem Formulation

In this section, we discuss our approach to modeling the social network environment, the dynamics of information propagation, and our strategy for containing misinformation.

2.1 Environment Description

A social network is formally represented as a directed graph $G = (V, E)$, where V denotes the set of nodes (agents), and E denotes the set of edges (relationships or connections) between agents [12]. The graph structure we consider for our study is undirected, indicating that relationships between agents are bi-directional. Each node within the graph represents an individual agent, and each agent holds a specific opinion on a given topic. An edge between any two nodes signifies a direct connection or relationship between those agents, facilitating the exchange of opinions.

Opinion values are quantified within the range $[-1, 1]$, representing different levels of sentiment, and the weight assigned to each edge quantifies the mutual trust level between connected agents, scaled within the interval $[0, 1]$. In our simulations, we explore three distinct cases of opinion and trust values. While existing literature works have explored only binary opinion and trust models, computational social science often models the opinion and trust values as continuous variables. Investigation of planning strategies in continuous models remains under-explored. `Case-1` involves binary opinion values with binary trust, simplifying the network dynamics into discrete states. In `Case-2`, we use floating-point opinion values while maintaining binary trust, allowing for a more granular assessment of opinions while still simplifying trust dynamics. Finally, `Case-3` features both floating-point opinion values and floating-point trust, representing more realistic opinion and trust relationships within the network capturing continuous variations.

2.2 Propagation Model

In the analysis of opinion networks, it is essential to understand how opinions form and evolve, guided by the dynamics of trust among agents. In our analysis, the evolution and propagation of opinions within opinion networks are modeled using a linear adjustment mechanism (*discrete linear maps*), as described by the following transition function

$$x_i(t+1) = x_i(t) + \mu_{ik}(x_k(t) - x_i(t)), \quad t = 0, 1, \ldots. \tag{1}$$

Equation 1 models the dynamics of opinion evolution, where the opinion of agent i at time $t+1$ depends on its current opinion and the influence exerted by a connected agent k, who is actively sharing some information with agent i, moderated by the trust factor μ_{ik}. This model adapts differently across various experimental setups as detailed below.

In `Case-1` and `Case-2`, where mutual trust values are discrete $\{0, 1\}$, the application of Equation 1 results in immediate shifts in opinion. For example, if an agent i with a current opinion value of 0.5 on some topic is influenced by a connected neighboring agent k with an opinion value of -1 on the same topic, agent i's opinion immediately shifts to -1 in the next timestep, reflecting a discrete transition. Conversely, in `Case-3`, which involves a continuous range of opinion and trust values, changes are more gradual. Here, if agent i holds an opinion of 0.5 and is influenced by a neighbor k with an opinion of -1 and a moderate trust factor, the opinion of agent i incrementally moves closer to -1 in subsequent timesteps. This reflects a gradual shift towards a consensus opinion, depending on the magnitude of the trust level between agents i and k.

To further enhance our understanding of opinion dynamics in networks with continuous trust relationships, we have also used the DeGroot propagation model [4] in `Case-3`. The propagation of opinions in this model is governed by the following equation:

$$x_i(t+1) = \sum_{k=1}^{n} \mu_{ik} x_k(t), \quad t = 0, 1, \ldots. \tag{2}$$

Equation 2 describes the opinion of agent i at time $t+1$ as a weighted average of the opinions of all the neighboring agents at time t, where the weights μ_{ik} represent the trust agent i has in agent k. Often, in the DeGroot model, the summation in 2 is a *convex sum*, i.e., the trust values add to one so that we have $\sum_{k=1}^{n} \mu_{ik} = 1$ for each $i = 1, \ldots, n$. This normalization allows the DeGroot model to exhibit stable asymptotic behaviour.

At each timestep, the following processes occur: Nodes with opinion values lower than -0.95 are identified as sources of misinformation and transmit the misinformation to their immediate neighbors (referred to as 'candidate nodes') according to one of the propagation model detailed in Equations 1 and 2. Concurrently, an intervention strategy is applied where a subset of these neighbors—constrained by *an action budget*—is selected to receive credible information from a trusted source. This source is characterized by an opinion value of 1 and we vary trust parameter among 1, 0.8, and 0.75. The process includes a blocking mechanism where a node that exceeds a positive opinion threshold of 0.95 is considered 'blocked', ceasing to interact with the misinformation spread or disseminate positive influence further. The simulation concludes when there are no viable `'candidate nodes'` left to propagate misinformation. **Our primary objective** is to devise a learning mechanism that efficiently identifies and selects key nodes within the network to disseminate accurate information at each time step.

3 Methods

In this section, we will explain our methodologies, presenting an overview of the network architectures employed, including the GCN and ResNet frameworks. Additional details about the neural network architecture utilized for our experiments can be found in Appendix A.2. We detail our proposed ranking algorithm utilized in the SL process. Additionally, we elaborate on the implementation of the Deep Q-Network with experience replay for RL. Furthermore, we provide an explanation of the various reward functions designed for our RL setup.

3.1 Ranking Algorithm based Supervised Learning

In this section, we propose a ranking algorithm based SL model to classify the key nodes at each time step to disseminate accurate information. Our SL method utilizes a GCN architecture.

Ranking Algorithm: We pose the ranking algorithm as a search problem where the objective is to find the optimal set of nodes that when blocked minimizes the overall infection rate. The network is represented as a graph G, where nodes can be infected, blocked, or possess opinion values within the range $[-0.95, 0.95]$. Initially, a simulation network S is created by setting the opinion values of infected nodes to -1 and removing blocked nodes. Let M denote the number of nodes in S that are

neither infected nor blocked. Given an action budget K, we select K nodes from M in $\binom{M}{K}$ possible ways, forming the set C of all possible combinations. For each subset $c \in C$, we temporarily block the nodes in c by setting their opinion values to 1 and simulate the spread of misinformation within S. The resulting infection rates for each subset c are stored in the set R. We identify the subset $c^* \in C$ that yields the minimal infection rate, denoted as r^*. This subset c^* is our target set. We then construct a target matrix $T \in \mathbb{R}^{N\times 1}$, where N is the total number of nodes in the original network G. All entries of T are initialized to 0, and for each node $i \in c^*$, the i-th entry of T is set to 1. This target matrix T is subsequently used to train the GCN-based model. A pseudocode for this ranking algorithm is presented in Algorithm 1 in Appendix A.5.

Overall Training Procedure: The training of our GCN-based model leverages the labels defined in the target matrix $T \in \mathbb{R}^{N\times 1}$. This matrix is compared with the model's output matrix $O \in \mathbb{R}^{N\times 1}$, which estimates the blocking probability of each node. We evaluate training efficacy using the binary cross-entropy loss between T and O, which quantifies prediction errors. Model weight adjustments are implemented via standard backpropagation [20] based on this loss.

Each training iteration consists of several episodes, starting with the generation of a random graph state G containing initially infected nodes. The GCN then processes this state to output matrix O using the graph's features and structure. Labels are generated, as detailed above using the ranking algorithm, generating the target matrix T, and the binary cross-entropy loss between O and T is calculated for backpropagation. The environment updates by blocking predicted nodes, allowing infection spread, and adjusting node attributes. The process repeats until misinformation spread is halted, with each episode refining the graph's state for subsequent iterations. While the ranking algorithm employs a brute force approach to identify optimal nodes, which becomes increasingly complex in the case of continuous opinion models, its integration within the SL framework to generate synthetic labels for the network represents the novel aspect of our methodology.

3.2 Reinforcement Learning-based Centralized Dynamic Planners

In SL, the process of generating labels can be costly and impractical as network size increases. This is evident while considering mitigating misinformation propagation in large networks, where identifying the optimal set of nodes for blocking requires a combinatorial search that is computationally infeasible. Thus, RL emerges as a viable alternative.

We employ the Deep Q-Network (DQN) [26] framework using random exploration combined with experience replay. Unlike the classical DQN, where the network outputs a Q function corresponding to each possible action, we have modified our approach to develop a Deep Value Network (DVN) as the number of available actions in each time-step in our problem setup need not be fixed. In this modified version, the network outputs the value for a given input state. Consequently, the output layer consists of a single neuron instead of one neuron per action. The agent's experiences at each time step are stored in a replay memory for the neural network parameter updates. The loss function for training is given by

$$\mathcal{L}(s_t, s_{t+1}|\theta) = \left(r_t + \max_a \hat{V}_{\theta^-}(s_{t+1}) - V_\theta(s_t)\right), \tag{3}$$

where s_t represents the current state, s_{t+1} denotes the subsequent state after action a_t is taken, and r_t is the reward received for taking a_t in s_t. The specific reward functions used in this study are discussed later in the section.

Algorithm 2, in Appendix A.5, provides a detailed implementation of our DVN with experience replay, demonstrating the use of random exploration and network synchronization.

3.2.1 Reward Functions for RL setup

The reward function is designed to encourage policies that effectively mitigate the spread of misinformation. Specifically, the reward functions modeled for our study are: (1) $R_0 = -(\Delta\text{infection rate})$, where Δinfection rate is defined as the change in infection rate resulting from taking action a_t. Specifically, Δinfection rate = infection rate at s_{t+1} − infection rate at s_t, with s_{t+1} being the state after action a_t is applied at state s_t. This reward structure encourages the model to reduce the rate at which misinformation spreads by penalizing increases in the infection rate. (2) $R_1 = -(\#\text{ candidate nodes})$,

targets the immediate neighbors of infected nodes that are susceptible to becoming infected in the next timestep, thereby promoting strategies that minimize the potential for misinformation to spread. (3) $R_2 = -(\#\text{ candidate nodes}) - (\Delta\text{infection rate})$, merges these concepts, balancing the need to control both the number of susceptible nodes and the overall infection rate. (4) $R_3 = 1 - (\frac{\#\text{ time steps}}{\text{Total time steps}})$, rewards quicker resolutions, providing higher rewards for strategies that contain misinformation rapidly and evaluating the effectiveness only at the end of each episode. Furthermore, (5) $R_4 = -(\text{infection rate})$, directly penalizes the current infection rate, thus favoring actions that achieve lower overall infection rates. In addition to these individual rewards, our setup also includes a combined reward that incorporates elements of both R_3 and R_1. Throughout the simulation, the agent continually receives rewards based on the number of candidate nodes, fostering strategies that limit the expansion of the infection network. As the simulation concludes, the agent receives an episodic reward calculated as (6) $R_5 = -(\#\text{ candidate nodes}) - \frac{\#\text{ time steps}}{\text{Total time steps}}$, thereby reinforcing the importance of quick and efficient resolution of misinformation spread.

3.3 Network Architectures

In our experiments, we utilized a GCN to model node features within a network. Each node was characterized by three key features: opinion value, connectivity degree, and proximity to a misinformed node. These features were represented in a matrix $F \in \mathbb{R}^{N\times 3}$, where N denotes the total number of nodes. The feature matrix is dynamic and evolves to reflect changes in the network. It includes the opinion value, the connectivity degree, which identifies nodes potentially susceptible to misinformation while excluding those already blocked or misinformed, and the proximity to a misinformed node, which is calculated as the shortest path to the nearest infected node, assigning a distance of infinity to unreachable nodes.

We have also considered using Residual Network (ResNet) Architecture. The ResNet model implemented in our study is a variant of the conventional ResNet architecture. The core component of our ResNet model is the **ResidualBlock**, which allows for the training of deeper networks by addressing the vanishing gradient problem through skip connections. Each **ResidualBlock** consists of two sequences of convolutional layers (`Conv2d`), batch normalization (`BatchNorm2d`), and sigmoid activations. Complete details about the model architectures used in our study are provided in Appendix A.2.

4 Experiments

In this section, we present the details about training data generation and configurations chosen for our SL and RL methodologies. We also explain the test data used for evaluating the trained models.

4.1 Training Setup

In the SL setup, we experimented with three distinct graph structures: Watts-Strogatz, nearest neighbors $k = 3$ and a rewiring probability $p = 0.4$, Erdos-Renyi, with branching factor of 4, and Tree graphs, with branching factors randomly selected from the range $[1, 4]$. Each graph type facilitated training models to evaluate the influence of various structural dynamics on performance. We used the GCN model for the SL method. Due to consistent performance of the trained models on the different graph topologies, we chose the small-world topology to present all the subsequent analysis and summarize the results in Appendix A.6.1. We also trained centralized RL planners using both ResNet and GCN network architectures. Each trained configuration is represented as $model$-n-x-y, where $model \in$ {ResNet, GCN}, $n \in$ {10,25,50} represents the network length, $x \in$ {1,2,3} represents the number of initial infected nodes and $y \in$ {1,2,3} represents the action budget.

4.2 Test Data Generation

The datasets used in related works [14] are just network structures, while we did not find any real-time opinion propagation data. To evaluate our intervention strategies, we generate two synthetic datasets using the Watts-Strogatz model with the training dataset's configurations. This approach allows us to simulate complex networks and control the structure, connectivity, and initial infected nodes to assess our models effectively.

Dataset v1 examines the effects of network size and the initial count of infected nodes on misinformation spread. We explored network sizes of 10, 25, and 50 nodes with 1, 2, and 3 initially infected nodes, respectively, creating 9 unique datasets. Each configuration has 1000 random network states with the opinion values of non-infected nodes uniformly distributed between -0.5 and 0.6.

Dataset v2 examines how the initial connections of infected nodes affect the spread of misinformation. Like Dataset v1, it uses networks of 10, 25, and 50 nodes. However, it is varied in the initial number of connections (degrees of connectivity) for the infected nodes from 1 to 4. Here by degree of connectivity, we mean the number of `candidate nodes` present at the start of the simulation. This variation results in a total of 12 datasets for each configuration, with each dataset containing 1000 states. In these configurations, the number of initially infected nodes is randomly chosen between 1 to 3. For instance, in a scenario with three initially infected nodes, the network might still have a degree of connectivity as 1 if all the infected nodes are connected to the same 1 uninfected node.

5 Results and Discussion

In this section, we evaluate the models, using the `infection rate` metric, trained using our ranking-based SL and RL algorithms with various reward functions. We discuss the efficiency of these models using `Dataset v2`, particularly on a network of 50 nodes with a connectivity degree of 4, as it represents the most complex test dataset we generated. Similar evaluation results for other datasets can be found in the Appendix A.6. The details of the hardware used for our experiments are provided in the Appendix A.4. The code and the datasets generated in our study are submitted as a `zip` file. Our empirical investigation yielded insightful results regarding the performance of our trained models under various training conditions. With a comprehensive experimental evaluations, we seek to verify the following research questions while meeting the desired objective.

Objective and Research Questions: O1: Identify the optimal combination of initially infected nodes and action budget parameters for training models to effectively control the spread of misinformation. **RQ1**: Among the reward functions considered, which is the most effective one? **RQ2**: For reward functions that focus on the time of blocking, does adding any other factor lead to better results? If yes, which factor? **RQ3**: Do reward functions that look at global graph information perform better than those considering local, neighboring information? **RQ4**: Does GCN offer better scalability and performance when compared with ResNet.

O1: What is the best combination of initially infected nodes and action budget parameters for training the models to control the misinformation spread

To examine this we focused our analysis on the Mean Squared Error (MSE) loss plots obtained during the training phase. Figure 7 in Appendix A.6 illustrates the comparison of training loss across various network parameter settings for all considered reward types in `Case-1`, employing a ResNet model trained on a network of 50 nodes. The trend in loss convergence across episodes was found to be consistent for both the ResNet and GCN models across all cases examined. The analysis revealed that reward functions exhibiting lower and more stable loss values correlate with improved model learning performance. Our findings highlight that increasing the number of initially infected nodes typically elevates the stabilization point of MSE loss, indicating a more challenging learning environment. Additionally, a higher action budget contributes to increased MSE variability, reflecting the added complexity and generally poorer performance during training. Based on this analysis, we find ResNet-n-1-1 and GCN-n-1-1, $n \in \{10,25,50\}$, to be the best training configurations.

RQ1: Among the reward functions considered, which is the most effective one? Answer: R_4
To determine the best reward function for controlling the misinformation spread we compare the average infection rates using different reward functions. Table 1 presents the average infection rate values across different cases considered for `Dataset v2` with a degree of connectivity 4, featuring a network of 50 nodes, detailing the average infection rates. It compares the performance of the ResNet model, trained on a network of 50 nodes, with the GCN model, trained on a network of 10 nodes, using the reinforcement learning training algorithm across the different reward types, and the GCN model trained using SL on a network of 25 nodes. Results on the additional datasets are provided in Appendix A.6. It can be observed that reward function R_4 consistently provides lower infection rate values across different cases.

Table 1: Inference results on Dataset v2, with a degree of connectivity 4, featuring a network of 50 nodes. This table presents the average infection rates for different models, with ResNet trained on a network of 50 nodes and GCN trained on networks of 10 nodes, under various methods (`M.`) tested with varying action budgets (`A.`) across different cases considered.

A.	M.	Case-1		Case-2		Case-3	
		ResNet(50)	**GCN(10)**	**ResNet(50)**	**GCN(10)**	**ResNet(50)**	**GCN(10)**
1	RL+R_0	0.2334	0.2481	0.2496	0.2454	0.0449	0.0461
	RL+R_1	0.1917	0.1608	0.1965	0.1607	0.0449	**0.0435**
	RL+R_2	0.2427	0.1608	0.2148	0.1608	0.0451	0.0438
	RL+R_3	0.2331	0.2958	0.2281	0.3381	0.0444	0.046
	RL+R_4	0.199	**0.1593**	0.2513	**0.1596**	0.045	0.0442
	RL+R_5	-	0.1607	-	0.1605	-	0.0439
	SL+GCN(25)	0.304		0.2889		0.3715	
2	RL+R_0	0.0974	0.1012	0.0992	0.1007	**0.0398**	0.04
	RL+R_1	0.0886	**0.0842**	0.0901	0.0843	**0.0398**	**0.0398**
	RL+R_2	0.097	**0.0842**	0.0957	0.0843	0.0399	**0.0398**
	RL+R_3	0.0959	0.0969	0.0962	0.1032	**0.0398**	0.04
	RL+R_4	0.0898	**0.0842**	0.1005	**0.0842**	0.0399	**0.0398**
	RL+R_5	-	**0.0842**	-	**0.0842**	-	**0.0398**
	SL+GCN(25)	0.1464		0.1032		0.3491	
3	RL+R_0	0.0599	0.0599	0.0599	0.06	**0.0397**	0.0399
	RL+R_1	0.0599	**0.0597**	0.0598	**0.0597**	0.0398	**0.0397**
	RL+R_2	0.0598	**0.0597**	0.0599	**0.0597**	0.0398	**0.0397**
	RL+R_3	0.06	0.0598	0.0601	0.0602	**0.0397**	0.0399
	RL+R_4	0.0598	**0.0597**	0.06	**0.0597**	0.0398	0.0398
	RL+R_5	-	**0.0597**	-	**0.0597**	-	0.0398
	SL+GCN(25)	0.0488		0.0559		0.2526	

RQ2: For reward functions that focus on time of blocking, does adding any other factor lead to better result? If yes, which factor? Answer: Yes. `#candidate nodes`.

Reward function R_3, which is formulated to minimize the number of time steps required to halt the spread of misinformation, might inadvertently not be the most effective strategy for minimizing the overall infection rate within the network. As the reward is solely based on the speed of response, it does not directly account for the magnitude of the misinformation spread, that is, the number of nodes affected. Therefore, the agent may prioritize actions that conclude the propagation swiftly but do not necessarily result in the most substantial reduction in the spread of misinformation. However, our results indicate that under specific training configurations with an action budget or initial infected nodes greater than 1, the reward function R_3 outperforms others in maintaining lower infection rates, as shown in Figure 8 in Appendix A.6. This finding is significant since R_3, a sparser reward type, requires less computational effort and is independent of network observability. As the action budget increases the propagation tends to conclude in fewer timesteps thereby resulting in the RL agent receiving a higher reward in the case of R_3. Figure 2 shows that the RL agent trained with the R_3 reward function chooses actions that conclude propagation in the least time. Conversely, Figure 1 illustrates the sequence of actions chosen by an RL agent trained with the R_1 reward function on the same network. Although R_1 requires more time steps than R_3, it results in a lower infection rate. This can also be observed from Table 1, where the infection rate is higher for the R_3 reward function than any other reward function. In order to effectively implement this we have considered combining this episodic reward along with R_1, resulting in reward type R_5. This has shown a significant performance improvement when compared to the original version.

RQ3: Do reward functions that look at global graph information perform better than those considering local, neighboring information ? Answer: Yes

Analysis of the inference outcomes using Dataset v2, as presented in Table 1, provides substantial evidence supporting Hypothesis *RQ4*. Notably, single factor reward functions, specifically $R_1 = -(\#\,\text{candidate nodes})$ and $R_4 = -(\text{infection rate})$, consistently resulted in lower infection rates

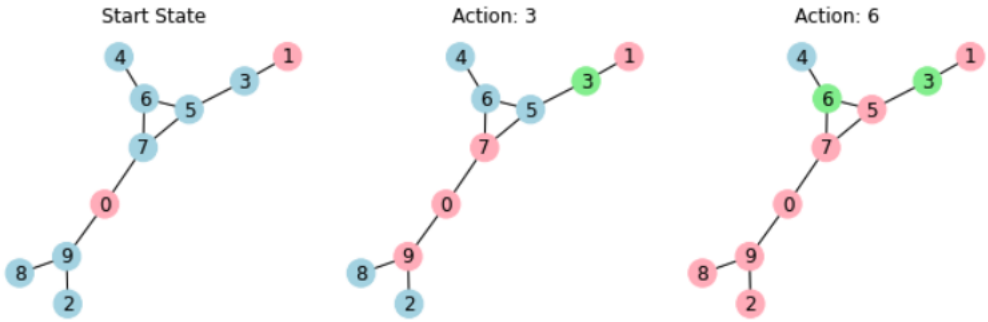

Figure 2: Sequence of actions chosen by the RL agent trained using reward function R_3. Infection Rate=0.7

across various settings compared to their more complex counterparts. This trend was observed in both ResNet and GCN models. From a practical standpoint, R_1 can be particularly advantageous because it does not require complete observability of the network, but just the immediate neighbors of infected nodes. Conversely, R_4, which involves the infection rate, demands a complete understanding of the state of each node within the network, for every time step, to be accurately computed. This requirement for total network observability could limit the practicality of R_4 in situations where such detailed information is unavailable or difficult to gather.

RQ4: Does GCN offer better scalability and performance when compared with ResNet? Answer: Yes

GCNs are hypothesized to outperform traditional convolution-based architectures like ResNet in tasks involving graph data due to their ability to naturally process the structural information of networks and their enhanced ability to represent complex feature sets. This study compares the scalability and performance of a GCN, which excels in node classification within graphs, to a ResNet model that, despite its success in image recognition, may not scale as effectively to larger graph structures beyond the size it was initially trained on. Referring to Table 1, the GCN model, trained on only 10 node networks, consistently exhibits lower average infection rates across all the cases and under varying action budgets, when compared with the ResNet model trained on 50 node networks. The ability of GCN to maintain lower infection rates even as network complexity increases underscores its robustness and scalability in more complex network scenarios. This performance contrast highlights the suitability of GCN architectures for graph-based tasks, supporting the hypothesis that GCNs offer greater scalability and better performance in managing graph data than architectures primarily designed for grid-like data structures.

6 Conclusions

This paper introduces scalable and innovative intervention strategies for containing the spread of misinformation within dynamic opinion networks. Our significant contributions include analysis using continuous opinion models, a novel ranking algorithm for identifying key nodes, and the utilization of GCNs to optimize intervention strategies. Additionally, we design and study various reward functions for reinforcement learning, enhancing our approach to misinformation mitigation.

Despite significant progress, our work has limitations. In the field of computational social science, often more complex agent models are being investigated. While we have made significant efforts to extend the understanding of planning strategies, especially in continuous opinion networks, exploring complex agent traits such as stubbornness and the representation of directed trust, and implementing topic-dependency in a multi-topic network along with distributed planners instead of centralized planners as in our work is a compelling future direction.

Broader Societal Impact: This work provides methods that can be used to exert control on information spread. When used responsibly by authorized information providers, which the authors support, it will help reduce prevalent *infodemics* in social media. But it may also be misused by an adversary to wean control from an authorized party (e.g., information owner) and counter efforts to tackle misinformation. Overall, the authors believe more research efforts are needed to control opinion networks in pursuit of long-term societal benefits.

References

[1] Daron Acemoglu and Asuman Ozdaglar. Opinion dynamics and learning in social networks. *Dynamic Games and Applications*, 1:3–49, 2011.

[2] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–236, 2017.

[3] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th international conference on World wide web*, pages 665–674, 2011.

[4] Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical association*, 69(345):118–121, 1974.

[5] Xuejun Ding, Mengyu Li, Yong Tian, and Man Jiang. Rbotue: Rumor blocking considering outbreak threshold and user experience. *IEEE Transactions on Engineering Management*, 2021.

[6] Israel Junior Borges Do Nascimento, Ana Beatriz Pizarro, Jussara M Almeida, Natasha Azzopardi-Muscat, Marcos André Gonçalves, Maria Björklund, and David Novillo-Ortiz. Infodemics and health misinformation: a systematic review of reviews. *Bulletin of the World Health Organization*, 100(9):544, 2022.

[7] Lidan Fan, Zaixin Lu, Weili Wu, Bhavani Thuraisingham, Huan Ma, and Yuanjun Bi. Least cost rumor blocking in social networks. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 540–549. IEEE, 2013.

[8] Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. Fake news mitigation via point process based intervention. In *International conference on machine learning*, pages 1097–1106. PMLR, 2017.

[9] Adam Fourney, Miklos Z Racz, Gireeja Ranade, Markus Mobius, and Eric Horvitz. Geographic and temporal trends in fake news consumption during the 2016 us presidential election. In *CIKM*, volume 17, pages 6–10, 2017.

[10] Chao Gao and Jiming Liu. Modeling and restraining mobile virus propagation. *IEEE transactions on mobile computing*, 12(3):529–541, 2012.

[11] Mahak Goindani and Jennifer Neville. Social reinforcement learning to combat fake news spread. In *Uncertainty in Artificial Intelligence*, pages 1006–1016. PMLR, 2020.

[12] Qiang He, Dafeng Zhang, Xingwei Wang, Lianbo Ma, Yong Zhao, Fei Gao, and Min Huang. Graph convolutional network-based rumor blocking on social networks. *IEEE Transactions on Computational Social Systems*, 2022.

[13] Yanqing Hu, Shenggong Ji, Yuliang Jin, Ling Feng, H Eugene Stanley, and Shlomo Havlin. Local structure can identify and quantify influential global spreaders in large scale social networks. *Proceedings of the National Academy of Sciences*, 115(29):7468–7472, 2018.

[14] Jiajian Jiang, Xiaoliang Chen, Zexia Huang, Xianyong Li, and Yajun Du. Deep reinforcement learning-based approach for rumor influence minimization in social networks. *Applied Intelligence*, 53(17):20293–20310, 2023.

[15] Zhongyuan Jiang, Xianyu Chen, Jianfeng Ma, and S Yu Philip. Rumordecay: rumor dissemination interruption for target recipients in social networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(10):6383–6395, 2022.

[16] Elias Boutros Khalil, Bistra Dilkina, and Le Song. Scalable diffusion-aware optimization of network topology. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1226–1235, 2014.

[17] Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. Interpretable rumor detection in microblogs by attending to user interactions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8783–8790, 2020.

[18] Masahiro Kimura, Kazumi Saito, and Hiroshi Motoda. Minimizing the spread of contamination by blocking links in a network. In *Aaai*, volume 8, pages 1175–1180, 2008.

[19] Sumeet Kumar and Kathleen M Carley. Tree lstms with convolution units to predict stance and rumor veracity in social media conversations. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 5047–5058, 2019.

[20] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988.

[21] Yaguang Lin, Zhipeng Cai, Xiaoming Wang, and Fei Hao. Incentive mechanisms for crowd-blocking rumors in mobile social networks. *IEEE Transactions on Vehicular Technology*, 68(9):9220–9232, 2019.

[22] Bo Liu, Xiangguo Sun, Qing Meng, Xinyan Yang, Yang Lee, Jiuxin Cao, Junzhou Luo, and Roy Ka-Wei Lee. Nowhere to hide: Online rumor detection based on retweeting graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[23] Jing Ma, Wei Gao, and Kam-Fai Wong. Rumor detection on twitter with tree-structured recursive neural networks. Association for Computational Linguistics, 2018.

[24] Ling-ling Ma, Chuang Ma, Hai-Feng Zhang, and Bing-Hong Wang. Identifying influential spreaders in complex networks based on gravity formula. *Physica A: Statistical Mechanics and its Applications*, 451:205–212, 2016.

[25] Mohammad Ali Manouchehri, Mohammad Sadegh Helfroush, and Habibollah Danyali. Temporal rumor blocking in online social networks: A sampling-based approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(7):4578–4588, 2021.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[27] B. Muppasani, V. Narayanan, B. Srivastava, and M. N. Huhns. Expressive and flexible simulation of information spread strategies in social networks using planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23820–23822, 2024.

[28] Le Nguyen and Nidhi Rastogi. Graph-based approach for studying spread of radical online sentiment. In *Companion Proceedings of the ACM Web Conference 2023*, pages 1373–1380, 2023.

[29] Abu Quwsar Ohi, MF Mridha, Muhammad Mostafa Monowar, and Md Abdul Hamid. Exploring optimal control of epidemic spread using reinforcement learning. *Scientific reports*, 10(1):22106, 2020.

[30] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36, 2017.

[31] Santhoshkumar Srinivasan and Dhinesh Babu LD. A social immunity based approach to suppress rumors in online social networks. *International Journal of Machine Learning and Cybernetics*, 12:1281–1296, 2021.

[32] Tetsuro Takahashi and Nobuyuki Igata. Rumor detection on twitter. In *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*, pages 452–457. IEEE, 2012.

[33] Guangmo Tong, Weili Wu, and Ding-Zhu Du. Distributed rumor blocking with multiple positive cascades. *IEEE Transactions on Computational Social Systems*, 5(2):468–480, 2018.

[34] Guangmo Tong, Weili Wu, Ling Guo, Deying Li, Cong Liu, Bin Liu, and Ding-Zhu Du. An efficient randomized algorithm for rumor blocking in online social networks. *IEEE Transactions on Network Science and Engineering*, 7(2):845–854, 2017.

[35] Guangmo Amo Tong and Ding-Zhu Du. Beyond uniform reverse sampling: A hybrid sampling technique for misinformation prevention. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 1711–1719. IEEE, 2019.

[36] Hanghang Tong, B Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 245–254, 2012.

[37] Senzhang Wang, Xiaojian Zhao, Yan Chen, Zhoujun Li, Kai Zhang, and Jiali Xia. Negative influence minimizing by blocking nodes in social networks. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[38] Penghui Wei, Nan Xu, and Wenji Mao. Modeling conversation structure and temporal dynamics for jointly predicting rumor stance and veracity. *arXiv preprint arXiv:1909.08211*, 2019.

[39] Qingqing Wu, Xianguan Zhao, Lihua Zhou, Yao Wang, and Yudi Yang. Minimizing the influence of dynamic rumors based on community structure. *International Journal of Crowd Science*, 3(3):303–314, 2019.

[40] Ruidong Yan, Deying Li, Weili Wu, Ding-Zhu Du, and Yongcai Wang. Minimizing influence of rumors by blockers on social networks: algorithms and analysis. *IEEE transactions on network science and engineering*, 7(3):1067–1078, 2019.

[41] Ruidong Yan, Yi Li, Weili Wu, Deying Li, and Yongcai Wang. Rumor blocking through online link deletion on social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(2):1–26, 2019.

[42] Enyu Yu, Yan Fu, Qing Tang, Jun-Yan Zhao, and Duan-Bing Chen. A re-ranking algorithm for identifying influential nodes in complex networks. *IEEE Access*, 8:211281–211290, 2020.

[43] Ahmad Zareie and Rizos Sakellariou. Rumour spread minimization in social networks: A source-ignorant approach. *Online Social Networks and Media*, 29:100206, 2022.

[44] Jianguo Zheng and Li Pan. Least cost rumor community blocking optimization in social networks. In *2018 third international conference on security of smart cities, industrial control system and communications (SSIC)*, pages 1–5. IEEE, 2018.

A Appendix

A.1 Related Works

This section reviews existing studies on controlling the flow of misinformation in networks. While most previous research has focused on detecting fake news through various features such as linguistic, demographic, or community-based indicators, there has been comparatively less work on mitigating misinformation. Mitigation strategies are typically categorized into three main approaches: removing critical nodes, severing essential connections, and countering rumors with factual information. In the following sections, we will first discuss the literature on misinformation detection, followed by a review of studies aimed at mitigating the spread of misinformation. Finally, we will provide an overview of approaches utilizing Graph Neural Networks (GNN) and Reinforcement Learning (RL) to mitigate misinformation spread.

Misinformation Detection: Initial efforts in misinformation detection aimed at curbing rumor spread through strategic node blocking. Wu et al. [39] developed a community detection algorithm to segment network nodes, evaluate their influence, and block key nodes. Zheng and Pan [44] addressed the least cost rumor community blocking (LCRCBO) using a community-centric influence model and a greedy algorithm to select optimal nodes for containment. However, both methods have raised concerns regarding cost-effectiveness and operational efficiency. Expanding on node-centric approaches, Ding et al. [5] developed a dynamic rumor propagation model with algorithms to identify and remove critical nodes and connections, introducing an 'outbreak threshold' to evaluate interventions. In contrast, Khalil et al. [16] and Yan et al. [41] advanced edge removal strategies under a linear threshold (LT) model, creating heuristic algorithms to manage misinformation spread effectively [15]. Tong and Du [35] used a hybrid sampling method, which could assign high weights to users susceptible to misinformation, to pinpoint users most vulnerable to fake news, while Zareie and Sakellariou [43] introduced a passive edge-blocking technique that leverages entropy to balance network diffusion efficiency. Nguyen et al. [28] applied network analysis to explore sentiment propagation in social networks, finding that sentiments often cluster within comment threads, suggesting that online forums may serve as echo chambers that reinforce uniform opinions among participants.

Misinformation Propagation Minimization: This research category promotes disseminating truthful information as a counter-rumor measure. Lin et al. [21] suggested a crowdsourcing framework to enable users to select from various collaborative or independent rumor control methods. Tong et al. [33] analyzed the effectiveness of the peer-to-peer independent cascade (PIC) model in private social networks, where independent rumor agents create 'positive cascades', and demonstrated that such strategies are robust under Nash equilibria. Yan et al. [40] identified the rumor minimization challenge as monotonically decreasing and devised a two-stage process for selecting effective blocking candidates. Manouchehri et al. [25] addressed maximizing influence blocking (IBM) with considerations for timing and urgency, proposing an efficient, theoretically sound sampling method. Lastly, Srinivasan and LD [31] proposed a competitive cascade model that focuses on leveraging user opinions and the critical nature of rumors to identify and activate influential nodes promoting positive information.

These studies highlight the complexity of misinformation propagation minimization and underscore the need for deeper analysis. Our study builds on these works by exploring how misinformation spreads under different environmental conditions and agent dynamics. We aim to propose more effective mitigation strategies, contributing to a nuanced understanding of misinformation dynamics and enhancing the resilience of information networks.

GNN Based Approaches: Graph Convolutional Networks (GCNs) are increasingly being utilized to detect the rumors and propagation patterns within social networks. For instance, Wei et al. [38] developed a GCN-based model to analyze user stances and conversation content for better rumor detection. Ma et al. [23] created a tree kernel to compare similarities between subtrees in retweeting trees. Kumar and Carley [19] employed a multitask learning framework to extract representations from retweeting trees for rumor and stance detection. Similarly, Khoo et al. [17] explored various influences within a retweeting tree and utilized a transformer model to enhance rumor detection by learning the interactions among these influences. Moreover, Liu et al. [22] proposed a structure-aware

retweeting GNN that identifies rumor patterns based on retweeting behaviors. This method leverages both node and structural-level data, suggesting that propagation paths offer distinct insights into the credibility of the disseminated information. Contrastingly, while the detection of rumors has been extensively studied, the use of GCNs for rumor minimization remains under-explored. Authors in [12] introduce an innovative approach for blocking rumors on social networks by integrating user opinions with confidence levels into a new model (CBOA) and employing a directed GCN (DGCN) to identify and block critical nodes capable of mitigating rumor spread.

However, there are opportunities to enhance their study. Our research investigates the scalability of GCNs and their performance across three different environments with varying agent dynamics. Additionally, we propose a novel ranking algorithm for training GNNs. Furthermore, we identify scenarios where supervised learning faces challenges and address these limitations using reinforcement learning techniques.

RL Goindani and Neville [11] develop a social reinforcement learning approach to mitigate the spread of fake news through social networks. Their method involves learning an intervention model to enhance the spread of true news, thereby reducing the influence of fake news. The authors model the news diffusion process using a Multivariate Hawkes Process (MHP) and employ policy optimization to learn intervention strategies. Ohi et al. [29] investigate strategies to mitigate the spread of pandemics using a reinforcement learning approach. The model is based on the SEIR (Susceptible-Exposed-Infectious-Recovered) compartmental model. It allows for dynamic interaction where individuals move randomly, influencing the spread of the disease. The agent is trained to determine optimal movement restrictions (from no restrictions to full lockdowns) to minimize disease spread while considering economic factors.

Current research in this field largely focuses on identifying misinformation or removing nodes and edges to limit rumor propagation. While some studies, such as those by He et al. [12], investigate misinformation suppression methods, they often do not address complex environmental scenarios, which our study aims to explore. Our research specifically targets the minimization of misinformation spread after the detection of misinformed agents is done. We aim to hinder their attempts to disseminate false information by strategically blocking selective nodes with positive information. Our work begins with the implementation of a GNN-based supervised learning model to detect misinformation. To overcome the limitations of supervised learning, we further incorporate a reinforcement learning paradigm. This progression enables us to develop and optimize more effective strategies within complex network environments, ultimately enhancing the robustness of misinformation control measures.

A.2 Model Details

A.2.1 Graph Neural Networks (GNNs)

Graph Neural Networks (GCNs) are advanced deep learning models tailored for handling data with a graph structure. Such models are particularly adept at processing information within complex networks by learning to synthesize node representations. These representations are derived through the aggregation and transformation of information from neighboring nodes. Specifically, the representation of a node v_i in a graph G is iteratively updated by integrating information from its immediate neighbors, denoted as $N(v_i)$. This process employs a propagation function f, parameterized by neural network weights W and an activation function σ. The updated representation of a node in a multi-layered Graph Convolutional Network (GCN), as proposed in the literature, can be mathematically expressed as:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{4}$$

Here, $\tilde{A} = A + I_N$ represents the augmented adjacency matrix of the graph G, incorporating self-connections by adding the identity matrix I_N. The diagonal matrix $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ facilitates the normalization of $\tilde{A}$. The term $W^{(l)}$ denotes the weight matrix at layer l, and $\sigma(\cdot)$ is the activation function. The matrix $H^{(l)} \in \mathbb{R}^{N\times D}$ encapsulates the node features at layer l.

We engage with a graph comprising N nodes in the application discussed. Our GCN outputs a vector $O \in \mathbb{R}^{N\times 1}$, representing the likelihood of each node $v_i \in V$ being pivotal for propagation through the network. This mechanism enables the model to figure out significance of each node in mitigating information spread within the graph structure.

GCN Architecture Overview Our model is structured as follows: It comprises three graph convolutional layers followed by a linear layer. The architecture is designed to progressively transform the input node features into a space where the final classification (e.g., determining the likelihood of a node being pivotal in propagation minimization).

- **Initial Graph Convolution:** The model begins with a graph convolutional layer (`GCNConv`), taking `input_size` features and transforming them into a hidden representation of size `hidden_size`. This layer is followed by a ReLU activation function.
- **Hidden Graph Convolutional Layers:** After the initial layer, the architecture includes four `GCNConv` layers, all utilizing `hidden_size` units. These layers are designed to iteratively process and refine the features extracted from the graph's structure. Each of these layers is followed by a ReLU activation to introduce non-linearity.
  - The model employs repeated application of the `GCNConv` layer with `hidden_size` units, demonstrating the capacity to deepen the network's understanding of the graph's topology through successive transformations.
- **Output Graph Convolution:** A final `GCNConv` layer reduces the hidden representation to the desired `num_classes`, preparing the features for the prediction task.
- **Linear Layer and Sigmoid Activation:** The architecture concludes with a linear transformation (`nn.Linear`) directly mapping the output of the last `GCNConv` layer to `num_classes`. A sigmoid activation function is applied to this output, producing probabilities for each class in a binary classification scenario.

In this specific implementation, the `input_size` is set to 3, indicative of the initial feature dimensionality per node. The `hidden_size` is configured at 128, providing a substantial feature space for intermediate representations. Lastly, the `num_classes` is established at 1, signifying only one numerical output for each nodes.

Packages Used The development of our supervised learning models, particularly those utilizing graph convolutional networks, leveraged several Python packages instrumental in defining, training, and evaluating our models. Below is a list of these packages and a brief description of their roles in our implementation:

- `torch`: Serves as the foundational framework for constructing and training various neural network models, including those for graph-based data.
- `torch_geometric`: An extension of PyTorch tailored for graph neural networks. It provides efficient data structures for graphs and a collection of methods for graph convolutional operations, making it essential for implementing graph convolutional neural networks (GCNs).
- `networkx`: Utilized for generating and manipulating complex networks. In our project, `networkx` is primarily used for creating synthetic graph data and for preprocessing tasks that require graph analysis and manipulations before feeding the data into the neural network models.

A.2.2 Residual Network (ResNet)

The Residual Network (ResNet) model implemented in our study is a variant of the conventional ResNet architecture. The core component of our ResNet model is the **ResidualBlock**, which allows for the training of deeper networks by addressing the vanishing gradient problem through skip connections. Each **ResidualBlock** consists of two sequences of convolutional layers (`Conv2d`), batch normalization (`BatchNorm2d`), and sigmoid activations. A distinctive feature is the adaptation of the skip connection to include a convolutional layer and batch normalization if there is a discrepancy in the input and output channels or the stride is not equal to one.

ResNet Architecture Overview

- **Initial Convolution:** Begins with a convolutional layer applying 64 filters of size 3x3, followed by batch normalization and sigmoid activation.
- **Residual Blocks:** Three main layers (`layer1`, `layer2`, `layer3`) constructed with the `_make_layer` method. Each layer contains a sequence of **ResidualBlocks**, with channel sizes of 32, 64, and 128, respectively. The number of blocks per layer is determined by the `num_blocks` parameter.
  - Each **ResidualBlock** implements two sequences of convolutional operations, batch normalization, and sigmoid activation, with an optional convolution in the shortcut connection for channel or stride adjustments.
- **Adaptive Input Reshaping:** Inputs are dynamically reshaped to a square form based on the square root of the second dimension, ensuring compatibility with different input sizes.
- **Pooling and Output Layer:** Concludes with an average pooling layer to reduce spatial dimensions, followed by a fully connected layer mapping 128 features to a single output, thus producing the final prediction.

Training Details Our model is trained using the PyTorch library, leveraging its comprehensive suite of neural network tools and functions. The optimizer of choice is the Adam optimizer (`torch.optim.Adam`), selected for its adaptive learning rate properties, which helps in converging faster. The learning rate was set to 0.0005, balancing the trade-off between training speed and the risk of overshooting minimal loss values. The training process involved the iterative adjustment of weights through backpropagation, minimizing the loss calculated at the output layer. This procedure was executed repeatedly over batches of training data, with the model parameters updated in each iteration to reduce the prediction error.

Packages Used The implementation of our ResNet model and the training process was facilitated by the following Python packages:

- `torch`: Provides the core framework for defining and training neural networks.
- `torch.nn`: A submodule of PyTorch that contains classes and methods specifically designed for building neural networks.
- `torch.nn.functional`: Offers functional interfaces for operations used in building neural networks, like activations and pooling functions.
- `torch.optim`: Contains optimizers such as Adam, which are used for updating model parameters during training.

A.3 Metrics

Training Loss

- **SL**: In the Supervised Learning (SL) framework, for our study, we employed the Binary Cross Entropy (BCE) loss to train the Graph Convolutional Network (GCN) model. Here, we delineate the iterative process used during the training phase under this setting. The model is fed with a new graph state at the beginning of each iteration. The GCN produces an output vector $O \in [0,1]^{N\times 1}$, where N is the number of nodes in the graph. Each component of O, denoted as O_i, represents the probability that node i should be blocked to minimize the propagation rate in the network. A greedy algorithm is employed to ascertain the optimal node to block. For each node i, temporarily set the node as blocked. We compute the propagation rate of the network with node i blocked. Then, we revert the blockage of node i and proceed to evaluate the next node. We select the node that, when blocked, results in the lowest propagation rate across the network. Upon determining the node j, which yields the minimum propagation rate when blocked, a target vector $T \in 0, 1^{N\times 1}$ is constructed such that $T_j = 1$ (indicating the target node to block), and $T_i = 0$ for all $i \neq j$.

  The BCE loss between the output vector O and the target vector T is computed as follows:

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[T_i \log(O_i) + (1 - T_i)\log(1 - O_i)\right]$$

The Binary Cross Entropy (BCE) loss is particularly useful in this context because it measures the performance of the model in terms of how effectively it can predict the binary outcome (block/no block) for each node in the graph. Unlike our reinforcement learning setup, which utilizes batch updates across multiple states or episodes, the supervised learning approach updates the model weights based on the loss calculated from a single graph state per iteration.

- **RL**: The training loss for our model is computed using the Mean Squared Error (MSE) metric. In each episode, the loss is calculated across a batch of samples drawn from the replay buffer. Specifically, it measures the squared difference between the predicted value function of the current state from the policy network and the Bellman target — the observed reward plus the value of the subsequent state as estimated by the target network. Executing this calculation over batches of experiences allows the policy network to learn from a diverse set of state transitions, thereby refining its predictions to better approximate the true expected rewards through temporal difference learning.

Evaluation Metric - Infection Rate The infection rate measures the proportion of the network that is infected over time and is a crucial metric for evaluating the spread of misinformation within a simulated environment. It is calculated as the ratio of infected nodes to the total number of nodes within the network at a given timestep:

$$\text{Infection Rate} = \frac{\text{Number of Infected Nodes}}{\text{Total Number of Nodes}}$$

This metric serves not only as a means to understand the dynamics of misinformation spread during simulations but also as a vital testing metric for evaluating model performance on test datasets. Models that effectively contain or reduce the infection rate are considered to have performed well, as they demonstrate the ability to mitigate the spread of misinformation across the network.

A.4 Hardware details

We have used two servers to run our experiments. One with 48-core nodes each hosting 2 V100 32G GPUs and 128GB of RAM. Another with 256-cores, eight A100 40GB GPUs, and 1TB of RAM. The processor speed is 2.8 GHz.

A.5 Training Details

A.5.1 SL

Training Methodology: Our SL setup is coupled with a ranking algorithm which is shown in Algorithm 1. We GCN with an input size of 3 (opinion value, degree of node, proximity to source node), a hidden size of 128, and an output size of 1. The model was trained using the `Adam` optimizer with a learning rate of 0.001 and a binary cross-entropy loss function. The training process involved 1000 epochs, where in each epoch, a graph with 25 nodes was generated. During each epoch, the model iteratively minimized the infection rate by selecting nodes to block based on GCN output until no uninfected nodes remained. The loss was calculated and backpropagated, and the weights were updated accordingly. The total loss for each epoch was averaged over iterations.

A.5.2 RL

Training Algorithm: Shown in Algorithm 2.
In the DQN framework with experience replay, two neural networks are utilized: the target network $\hat{Q}_{\theta^-}$ and the policy network Q_θ. The target network is periodically updated by copying weights from the policy network, while the policy network is optimized continuously through the replay memory D, which stores agent's past experiences. During training, the agent samples random mini-batches of transitions from D to minimize the loss function via gradient descent, enabling the policy network

Algorithm 1 Ranking Algorithm

Input: Graph G, infected nodes I, blocked nodes B, action budget K
Output: Target matrix $T \in \mathbb{R}^{N \times 1}$
$S \leftarrow$ initialize_simulation_network(G, I, B)
$M \leftarrow$ get_uninfected_and_unblocked_nodes(S)
$C \leftarrow$ combinations(M, K)
$min\_rate \leftarrow 1$
$target\_set \leftarrow$ None

for $c \in C$ **do**
    $temp\_S \leftarrow S$
    block_nodes$(temp_S, c)$ ▷ Temporarily block nodes in c by setting their opinion values to 1
    $rate \leftarrow$ simulate_propagation$(temp_S)$ ▷ Infection Rate $= \frac{\text{Number of Infected Nodes}}{\text{Total Number of Nodes}}$
    if $rate < min\_rate$ **then**
        $min\_rate \leftarrow rate$
        $target\_set \leftarrow c$
    end if
end for

$T \leftarrow [0]^{N \times 1}$
for $i \in target\_set$ **do**
    $T[i] \leftarrow 1$
end for

return T

to estimate optimal actions. This strategy helps the agent break the temporal correlation inherent in sequential data, enhancing the stability and convergence of the learning process. We use the Mean Squared Error (MSE) metric to calculate the loss value between the policy network and the target network.

Training Methodology: The Neural Network model is trained using a variant of Q-learning, with a replay buffer approach to stabilize the learning process, aimed at learning the value function. Training commences with a fully exploratory policy (epsilon = 1) and transitions to an exploitation-focused strategy as epsilon decays over time. The learning rate is set to 5×10^{-4}, and mean squared error (MSE) loss is utilized to measure the prediction quality.

At each episode, 200 random initial states are generated, with a selected parameter of the number of initially infected nodes, and actions are determined through an epsilon-greedy method, balancing between exploration and exploitation. The agent performs actions by selecting nodes in the network to effectively contain misinformation spread. For each action, the network's new state and corresponding reward are observed, which are then stored in the replay buffer.

Batch updates are carried out by sampling from this buffer, ensuring that learning occurs across a diverse set of state-action-reward-next-state tuples. We have used a batch-size of 100 across the experiments. The policy network parameters are optimized using the Adam optimizer, and the target network's parameters are periodically updated to reflect the policy network, reducing the likelihood of divergence.

The training process continues for 300 number of episodes, with the epsilon parameter decaying after each timestep within an episode, encouraging the model to rely more on learned values rather than random actions as training progresses. The duration of each episode and overall training, along with average rewards and loss, are logged for post-training analysis. The model parameters yielding the best performance on the validation set are preserved for subsequent evaluation phases.

A.6 Inference Results

A.6.1 SL

We have conducted comprehensive testing of our SL model across various topographies and environments, examining the performance under different conditions. The overall performance comparison

Algorithm 2 DVN with experience replay

Input: states per episode n, batch size m, action budget k, parameter update interval T', max number of episodes $e_{\max}$
Output: V network $\hat{V}_\theta$
Initialize experience replay memory D
Initialize policy network V with random weights θ
Initialize target network $\hat{V}$ with weights θ^-
Initialize ϵ-decay to 1 and anneal to 0.1 with training
for $e = 1$ to $e_{\max}$ **do**
    $t \leftarrow 1$
    Generate n random states s_t with initial infected nodes
    Calculate candidate node set C for s_t
    while any $|C| > 0$ **do**
        Initialize blocker set $B_t \leftarrow \emptyset$
        Randomly sample a number x from uniform distribution $\mathcal{U}(0, 1)$
        if $x < \epsilon$ **then**
            Randomly sample k candidates from C as blocker set B_t
        else
            Initialize the infection prediction set $K \leftarrow \emptyset$
            for all node u of candidate set C **do**
                Calculate the infection number K_u using $V_\pi(s_{t+1})$, where s_{t+1} is the state resulting after taking action u in state s_t
                Append K_u to K
            end for
            Select the k nodes with the least infection prediction from K as the blocker set B_t
        end if
        Block the nodes in the blocker set B_t
        Update the state s_{t+1}
        Update the candidate set C
        Update the reward r_t
        $t \leftarrow t + 1$
        for $i = 0$ to $n - 1$ **do**
            Store the transition $(s^i_{t-1}, B^i_t, r^i_t, s^i_t)$ in D
        end for
        Sample a random minibatch of m transitions (s_j, a_j, r_j, s_{j+1}) from D
        $y_j \leftarrow \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \max_a \hat{V}_{\theta^-(s_{j+1})} & \text{otherwise} \end{cases}$
        Calculate loss using mean squared error between y_j and $V_\theta(s_j)$, set gradients to zero, perform backpropagation, and update weights using the Adam optimizer
        if $t \bmod T' = 0$ **then**
            Update target network $\theta^- \leftarrow \theta$
        end if
    end while
end for

for each model under these varied conditions is illustrated in Figure 3. This figure provides a comprehensive view of how the model performs across the different environments and topographical scenarios.

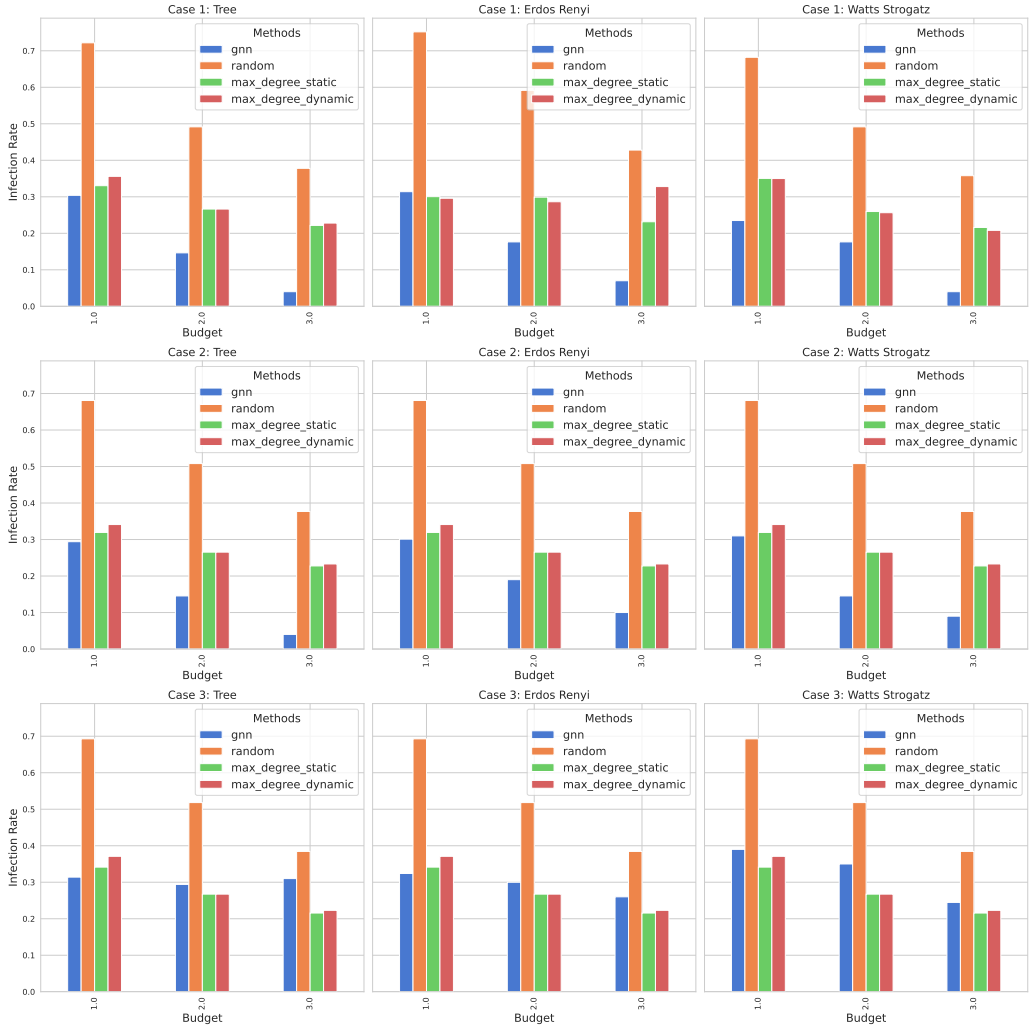

Figure 3: Comparative analysis of the GCN-Based SL Model Against Baseline Models Across Different Network Types and Budgets. Each subfigure represents one of the three cases $(1, 2, \text{and } 3)$, organized by rows, for three different types of networks: Tree, Erdős-Rényi, and Watts-Strogatz, organized by columns. Within each panel, the infection rate is plotted for four methodologies. SL based on GCN (blue), random node selection (orange), static selection of maximum degrees (green), and dynamic selection of maximum degrees (red) across three levels of budget (1, 2, and 3). These results underscore the variability in performance with changes in network structure and budget allocation, highlighting the superior effectiveness of the GCN model in simpler cases and under increased budget conditions, with diminishing returns in more complex environments.

Dataset v2 Testing In addition to the initial dataset, we have also tested our model using dataset v2. For a more granular analysis, we have compiled the test results into three distinct cases:

- **Case 1:** Detailed in Figure 4

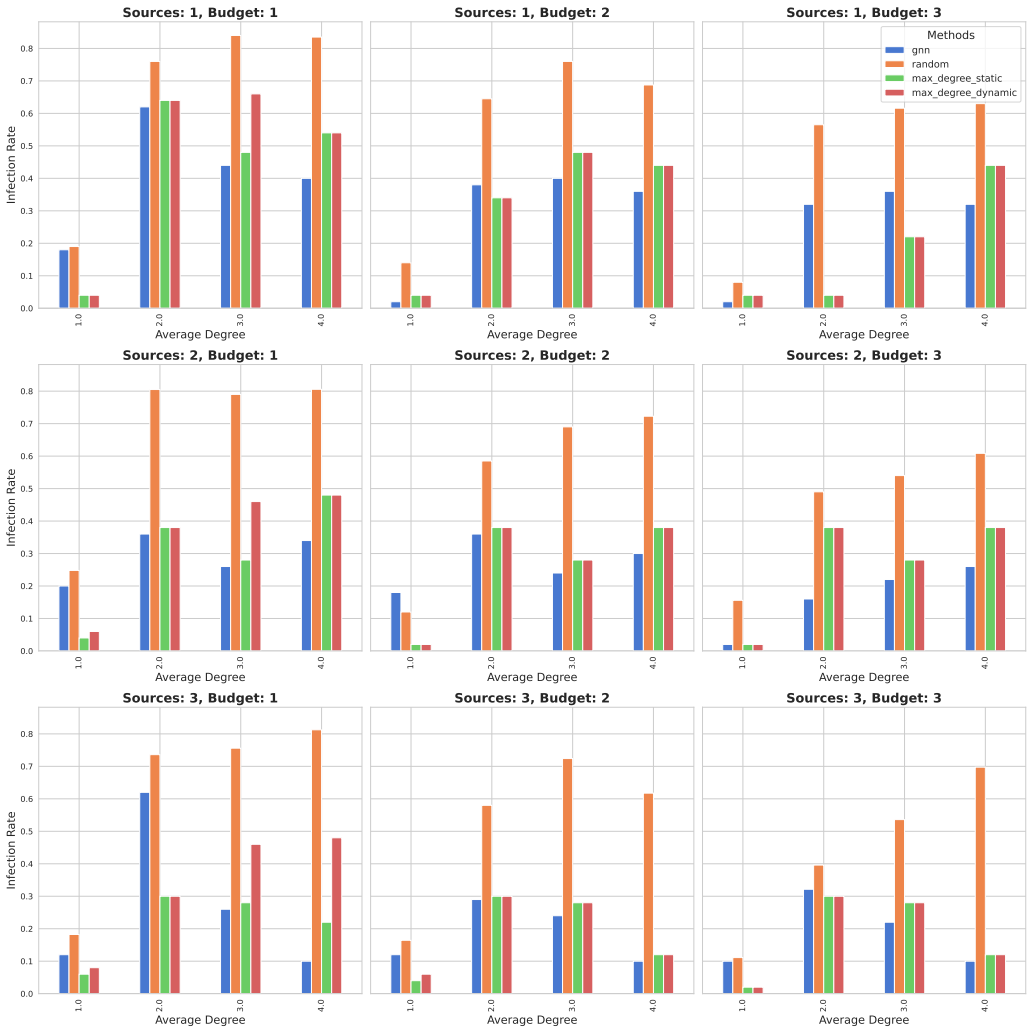

Figure 4: Case-1: Comparative Mean Infection Rate across different parameter settings for a GCN-based SL model trained on a 25-node dataset and tested on dataset v2 consisting of 50 nodes

- **Case 2:** Detailed in Figure 5

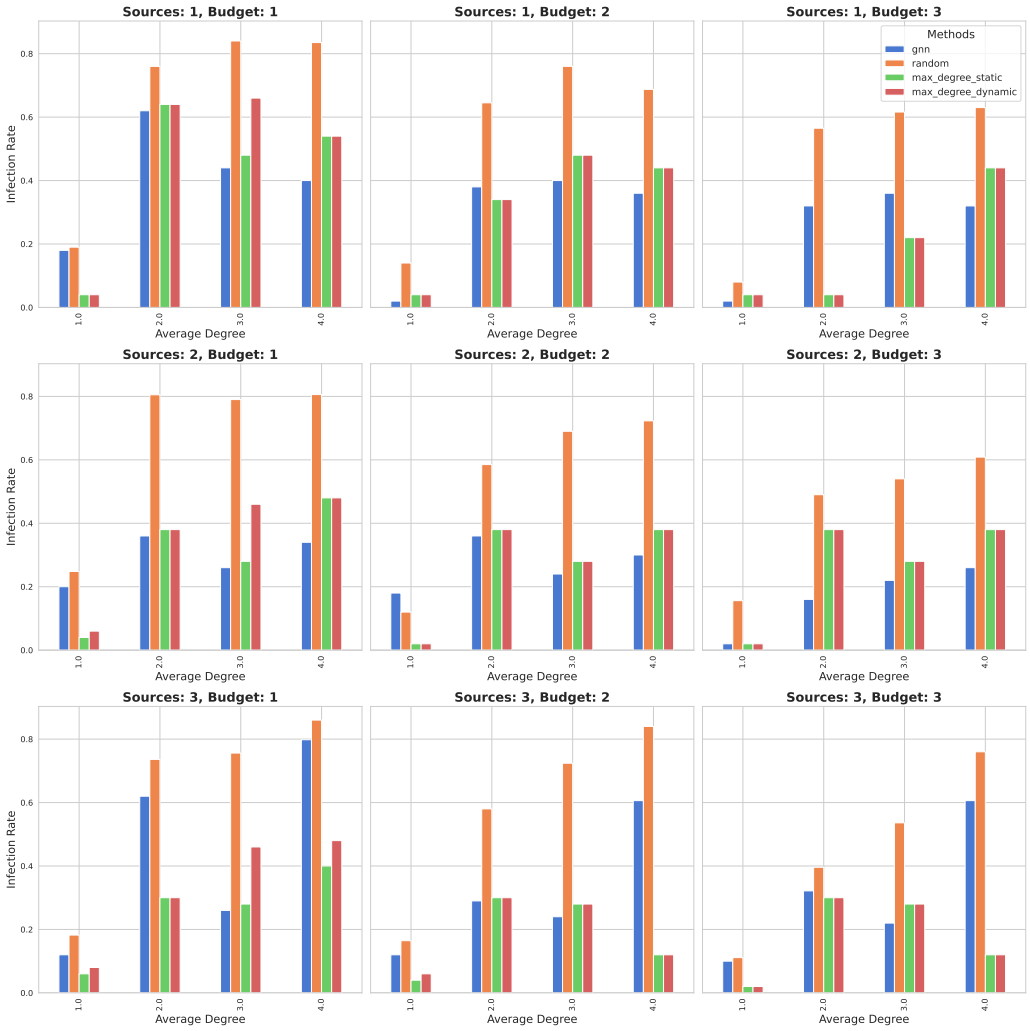

Figure 5: Case-2: Comparative Mean Infection Rate across different parameter settings for a GCN-based SL model trained on a 25-node dataset and tested on dataset v2 consisting of 50 nodes

- **Case 3:** Detailed in Figure 6

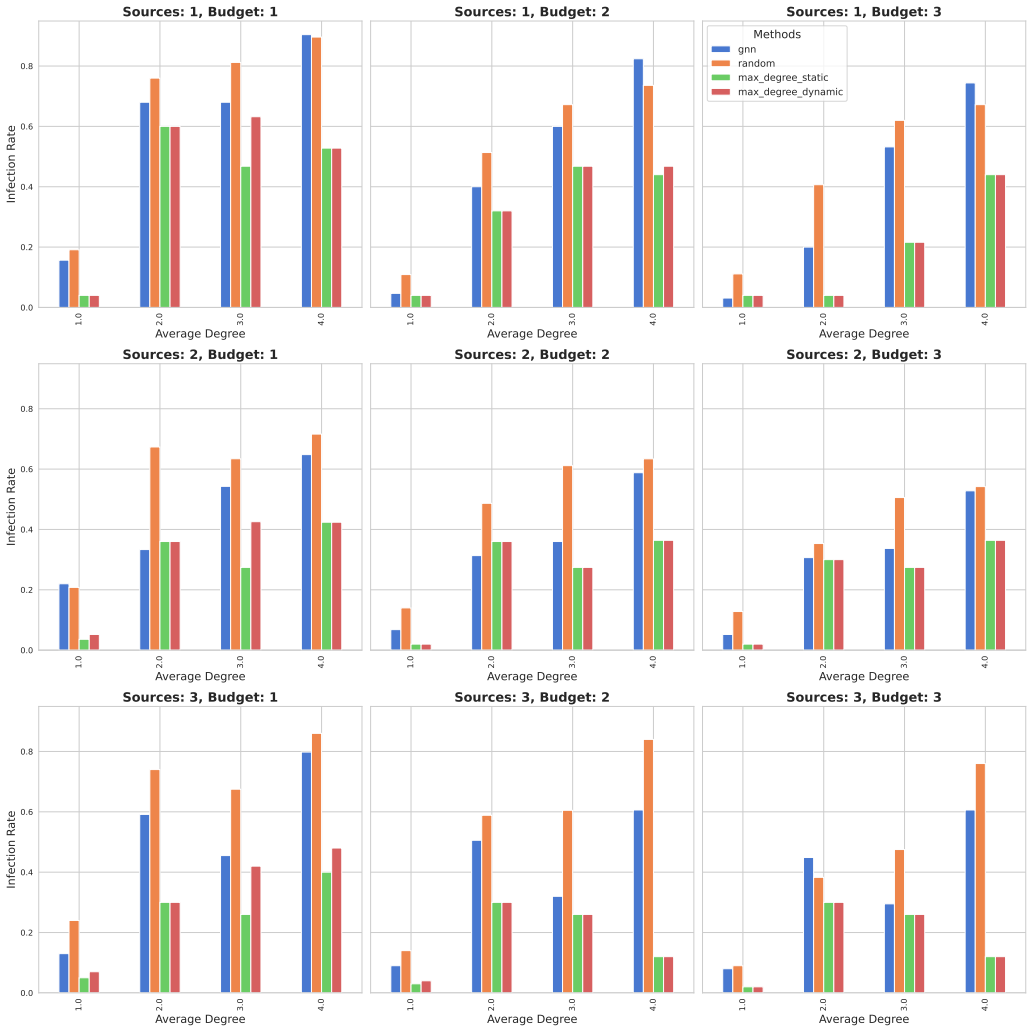

Figure 6: Case-3: Comparative Mean Infection Rate across different parameter settings for a GCN-based SL model trained on a 25-node dataset and tested on dataset v2 consisting of 50 nodes

A.6.2 RL

Comparison of MSE loss across different reward functions for Case-1: Figure 7.

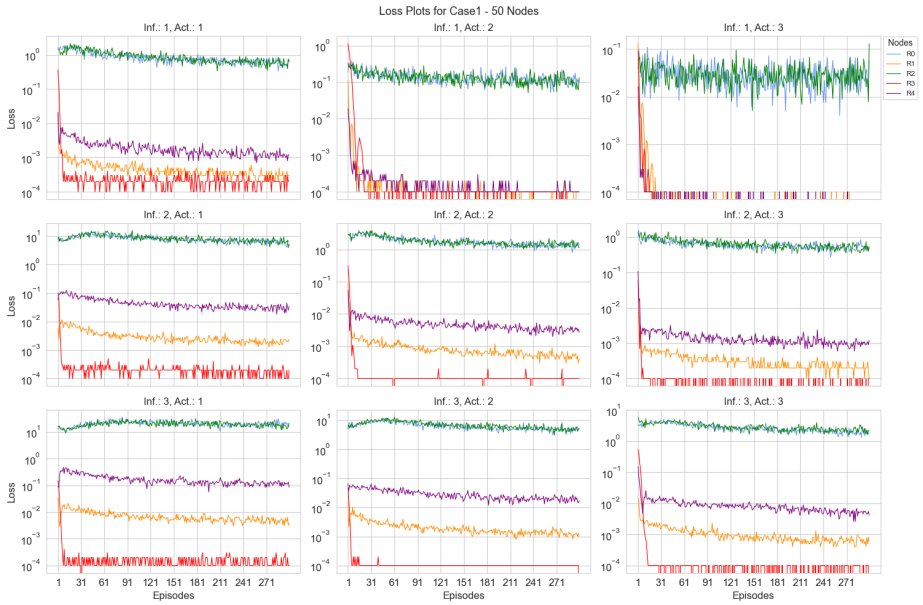

Figure 7: Case-1: Comparative MSE loss across different reward functions for a ResNet model trained on a 50-node dataset. Columns represent an increase in action budget during training, while rows indicate a rise in the number of initial infected nodes.

Comparison of Mean Infection Rate across different reward functions, showcasing that the reward R_3 performs better in model configurations with higher action budget and higher initial infected nodes: Figure 8

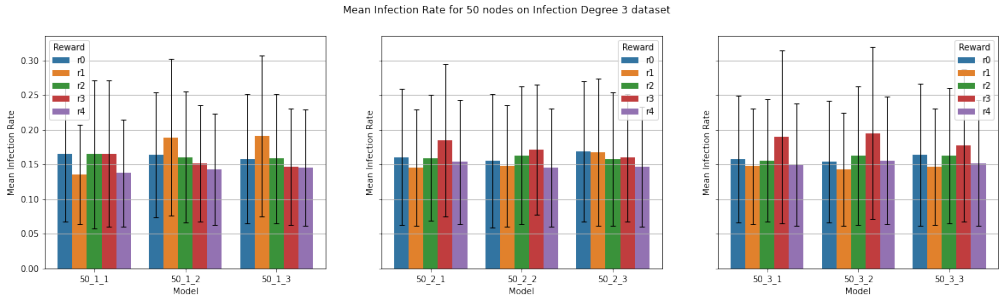

Figure 8: Case-1: Comparative Mean Infection Rate across different reward functions for a ResNet model trained on a 50-node dataset tested on Dataset v2 of 50 nodes with degree of connectivity 3.

Case-1

- **Type:** Binary Opinion and Binary Trust.
- **Opinion Dynamic Model:** Discrete Switching.

R0 The loss plot is presented in Figure 9. Dataset v1 Inference Result: 50 Nodes - Figure 10.

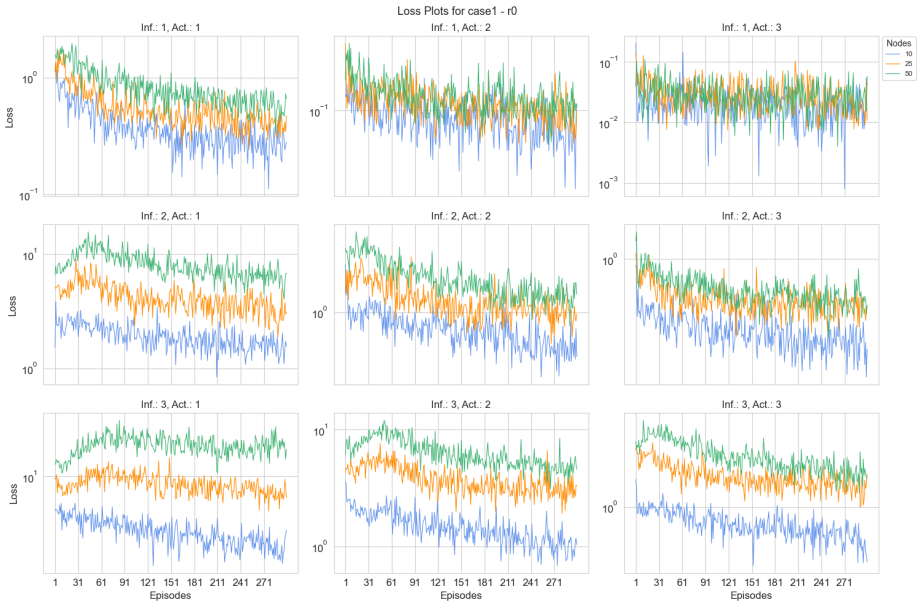

Figure 9: Case-1 using R0: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

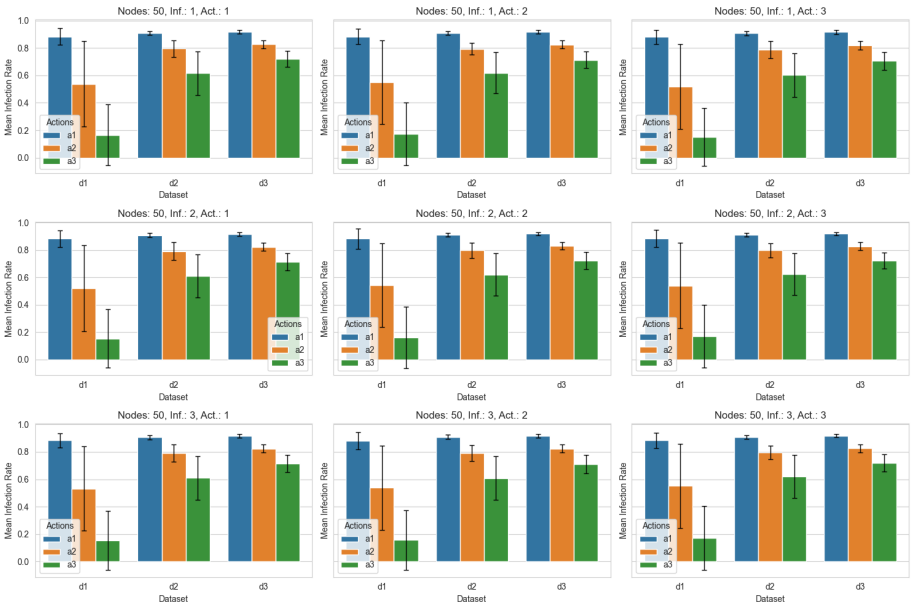

Figure 10: Case-1 using R0: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R1 The loss plot is presented in Figure 11. Dataset v1 Inference Result: 50 Nodes - Figure 12.

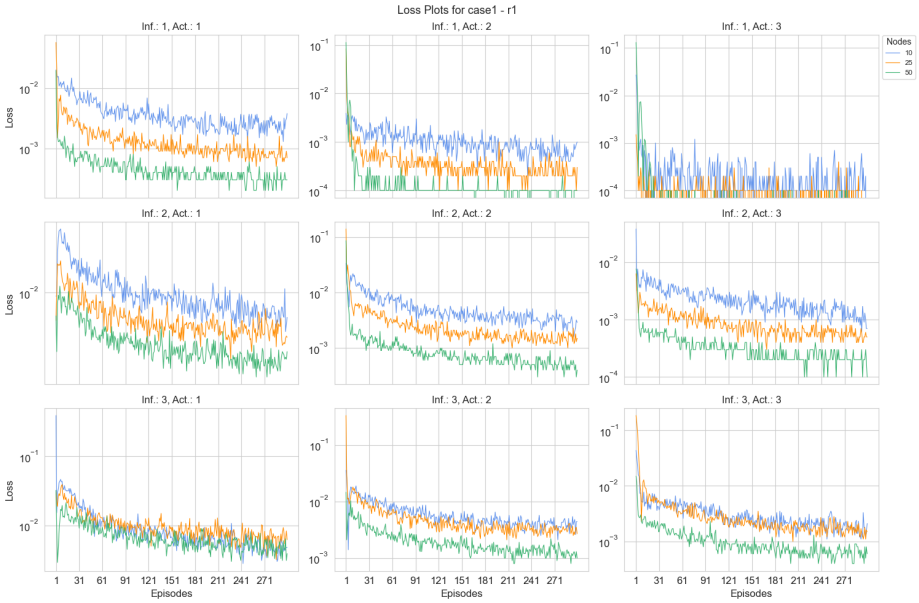

Figure 11: Case-1 using R1: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

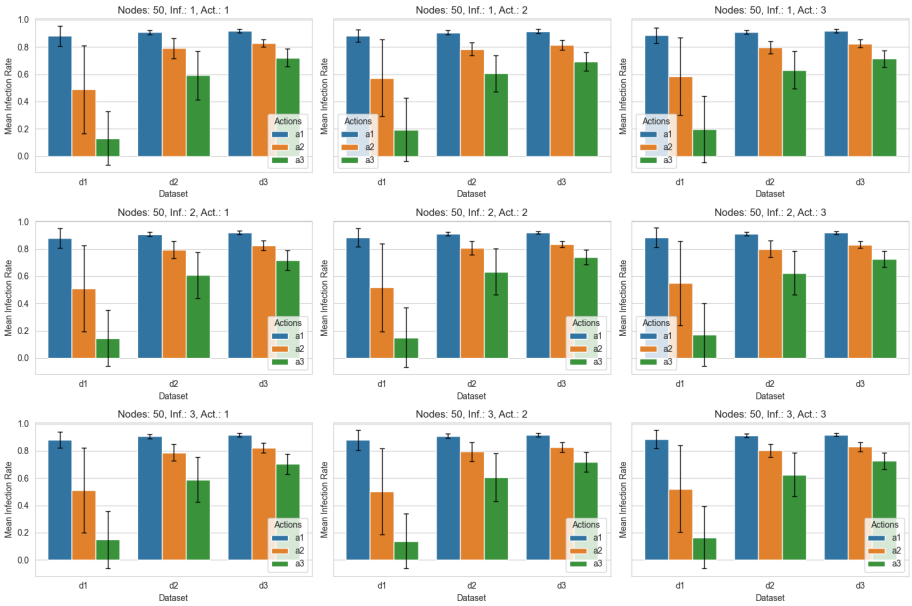

Figure 12: Case-1 using R1: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R2 The loss plot is presented in Figure 13. Dataset v1 Inference Result: 50 Nodes - Figure 14.

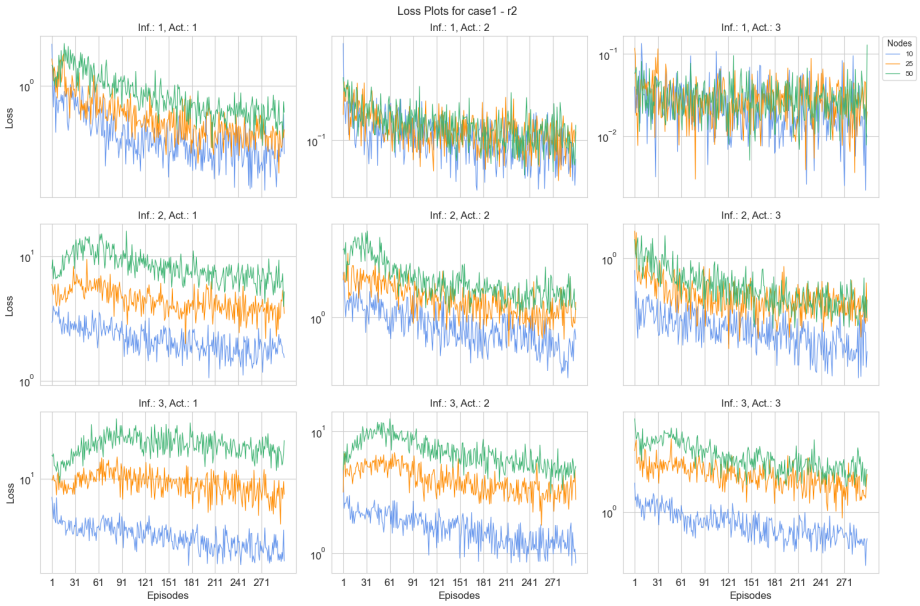

Figure 13: Case-1 using R2: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

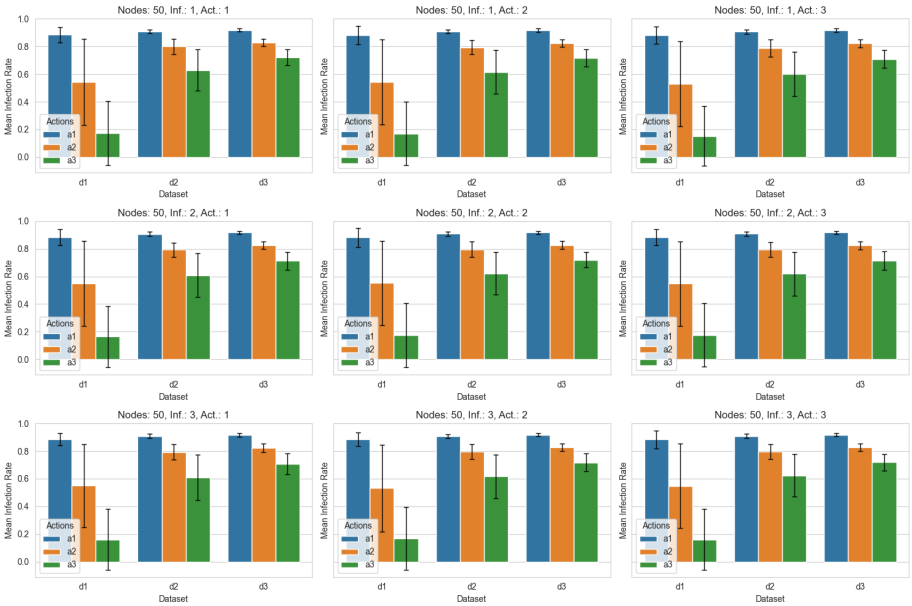

Figure 14: Case-1 using R2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R3 The loss plot is presented in Figure 15. Dataset v1 Inference Result: 50 Nodes - Figure 16.

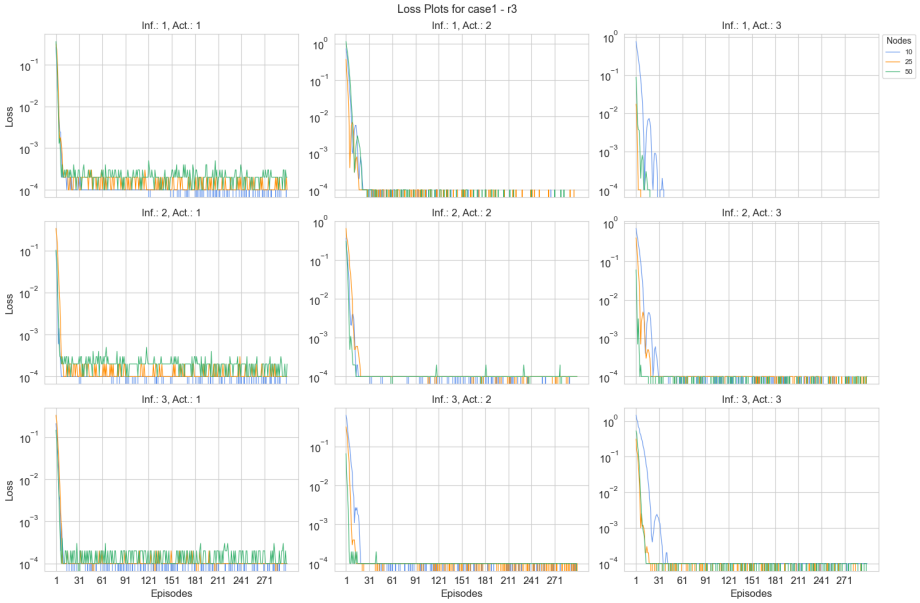

Figure 15: Case-1 using R3: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

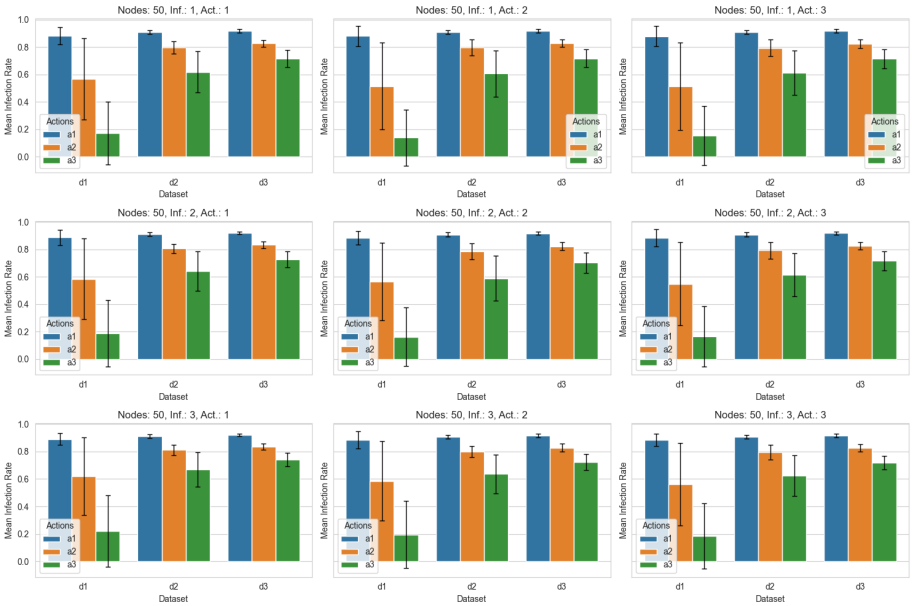

Figure 16: Case-1 using R3: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R4 The loss plot is presented in Figure 17. Dataset v1 Inference Result: 50 Nodes - Figure 18.

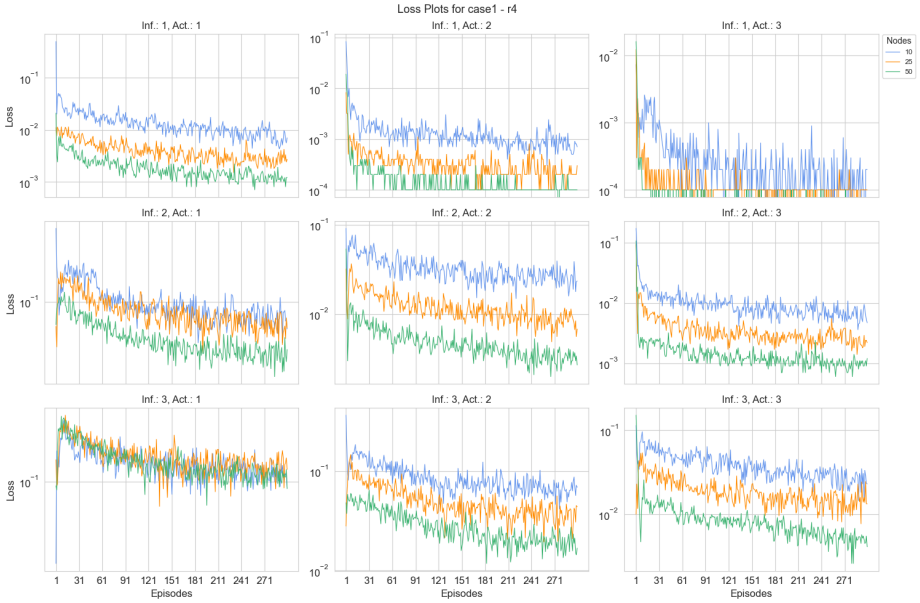

Figure 17: Case-1 using R4: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

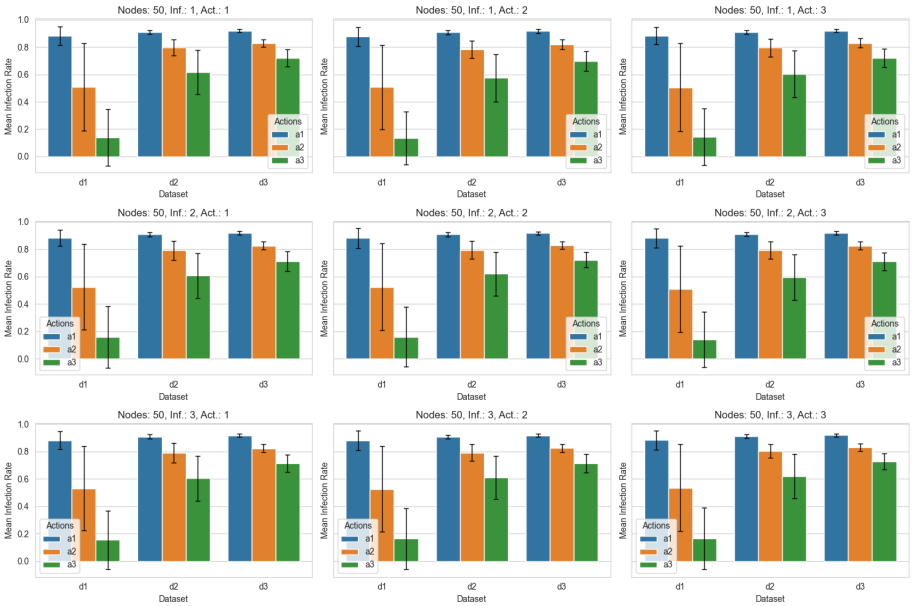

Figure 18: Case-1 using R4: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Dataset v2 Results

Degree of connectivity 1 50 Nodes - Figure 19

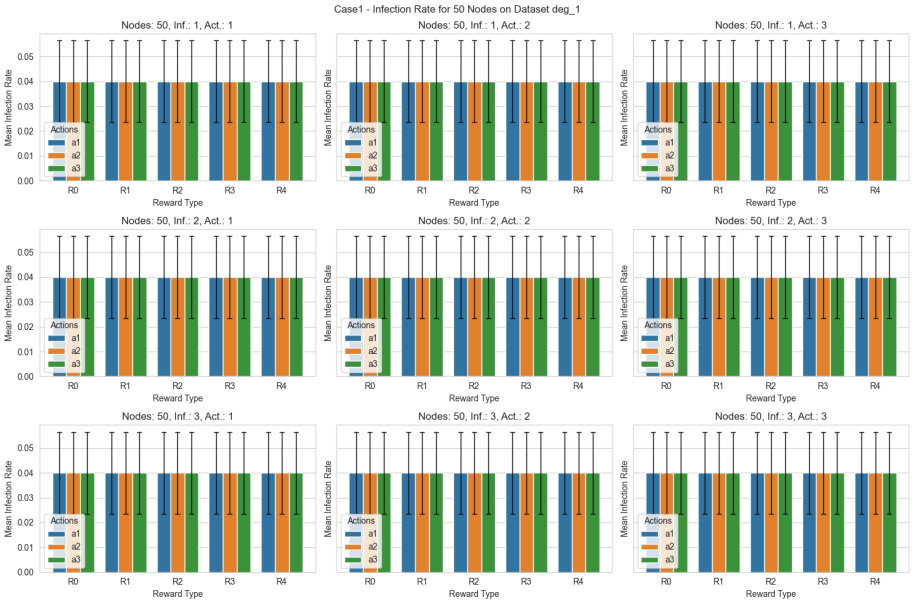

Figure 19: Case-1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of connectivity 1, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 2 50 Nodes - Figure 20

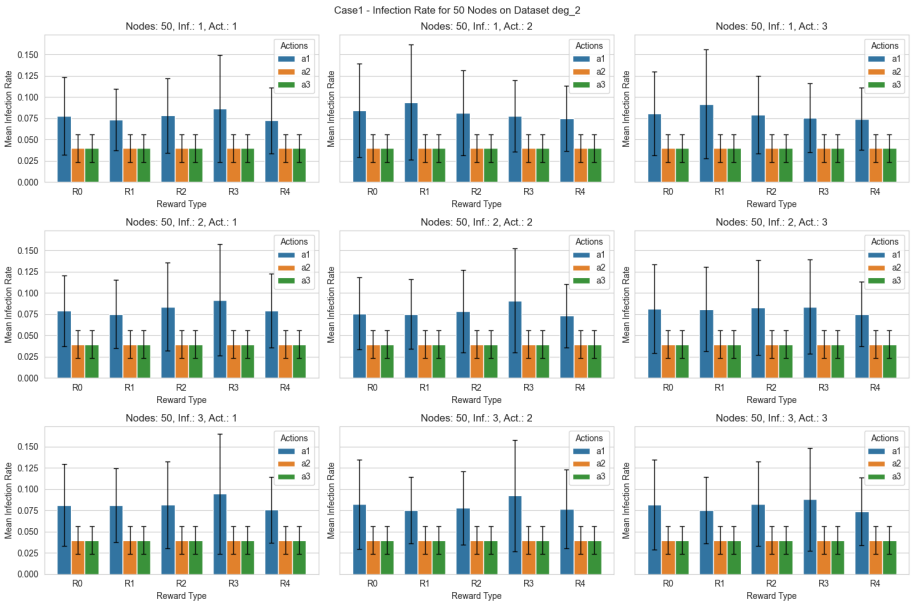

Figure 20: Case-1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of connectivity 2, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 3 50 Nodes - Figure 21

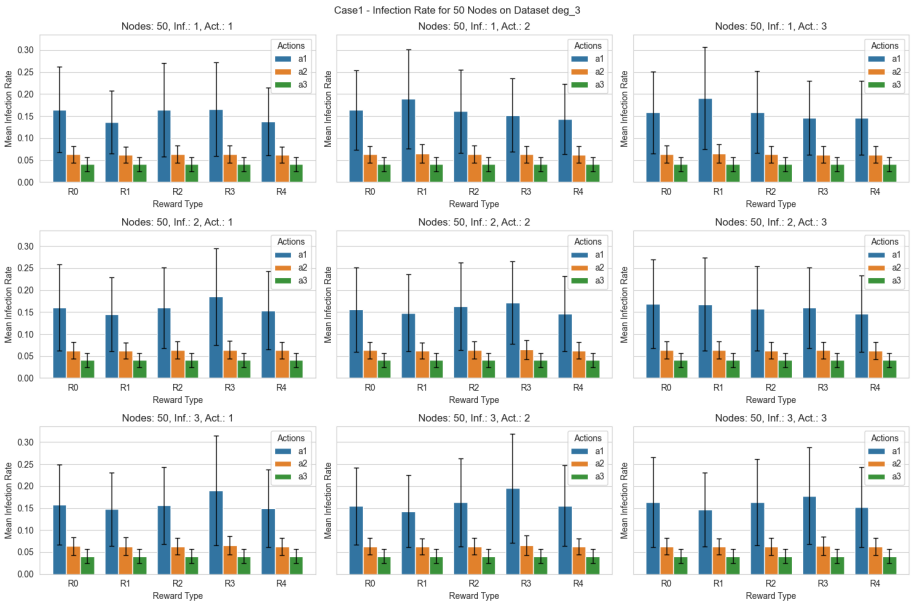

Figure 21: Case-1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of connectivity 3, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 4 50 Nodes - Figure 22

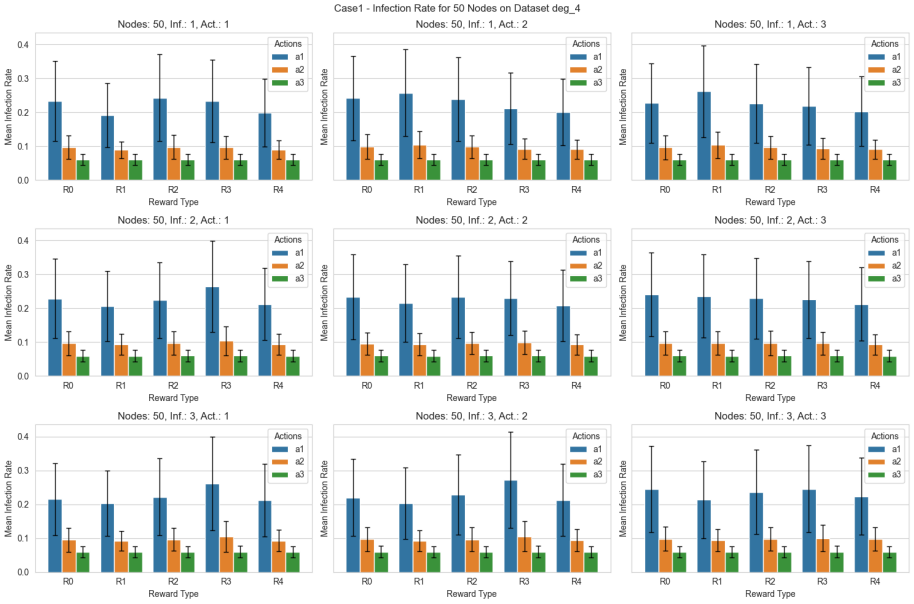

Figure 22: Case-1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of connectivity 4, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Case-2

- **Type:** Floating Point Opinion and Binary Trust.
- **Opinion Dynamic Model:** Linear Adjustment.

R0 The loss plot is presented in Figure 23. Dataset v1 Inference Result: 50 Nodes - Figure 24.

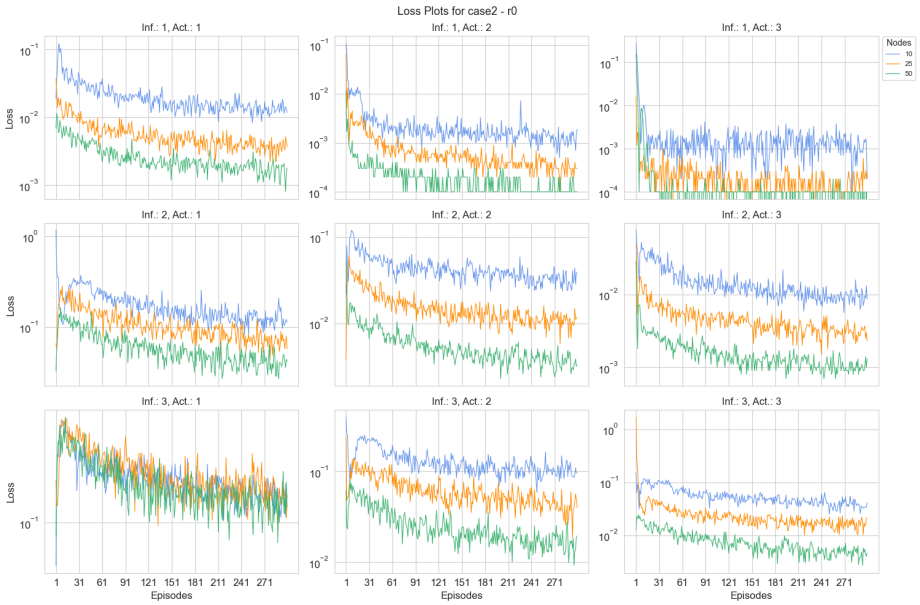

Figure 23: Case-2 using R0: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

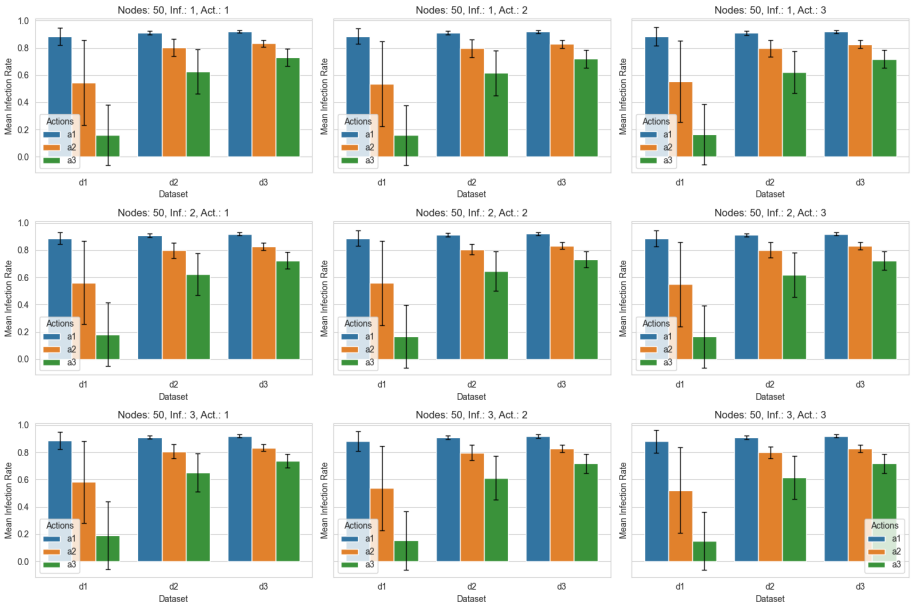

Figure 24: Case-2 using R0: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R1 The loss plot is presented in Figure 25. Dataset v1 Inference Result: 50 Nodes - Figure 26.

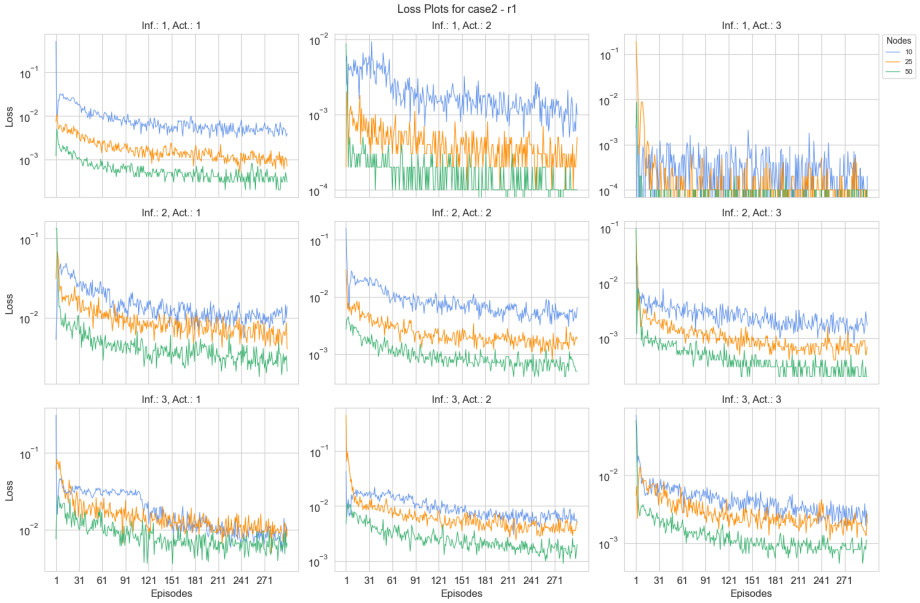

Figure 25: Case-2 using R1: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

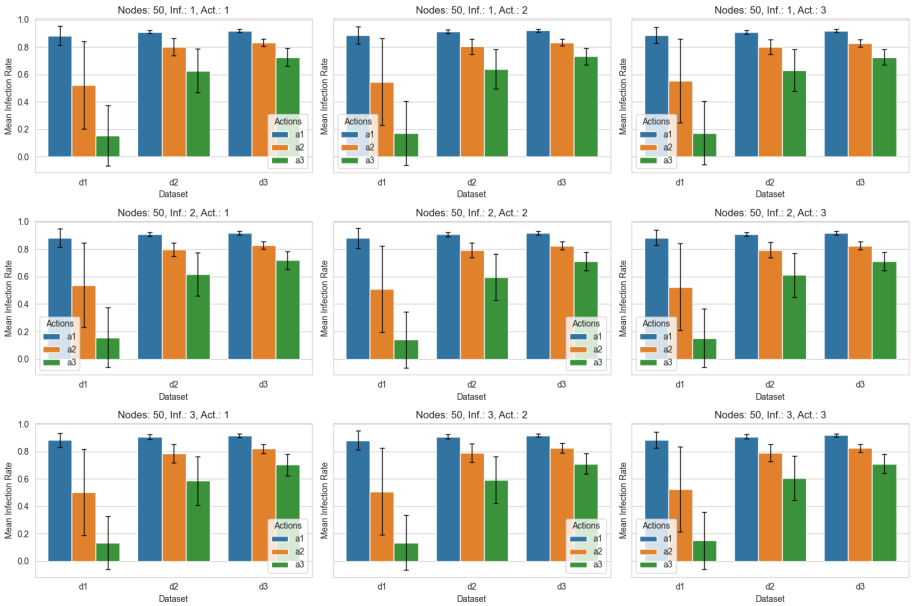

Figure 26: Case-2 using R1: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R2 The loss plot is presented in Figure 27. Dataset v1 Inference Result: 50 Nodes - Figure 28.

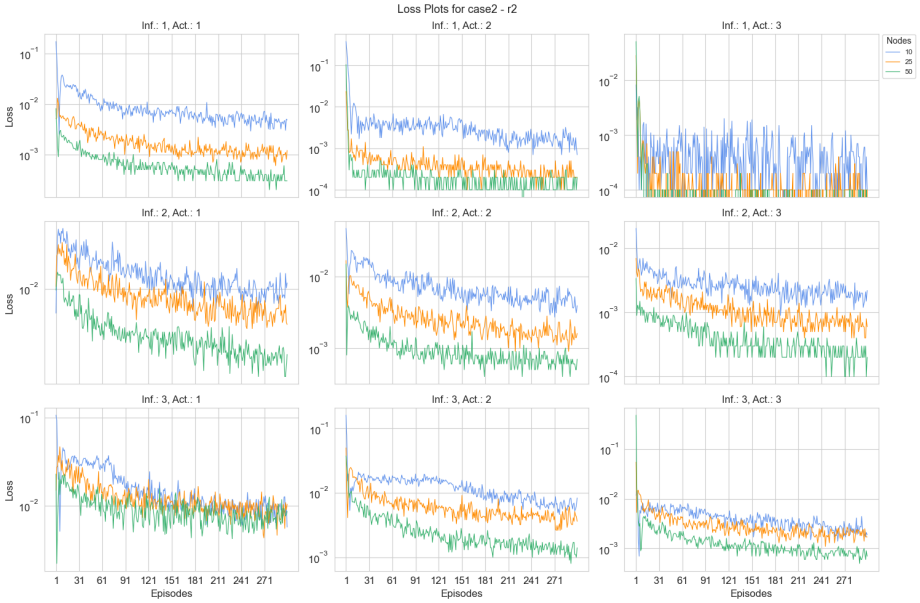

Figure 27: Case-2 using R2: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

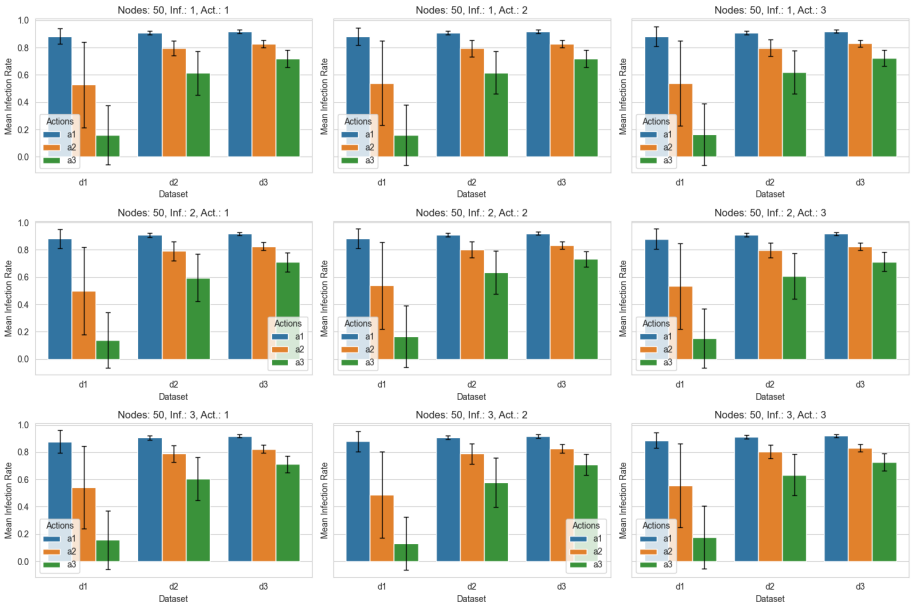

Figure 28: Case-2 using R2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R3 The loss plot is presented in Figure 29. Dataset v1 Inference Result: 50 Nodes - Figure 30.

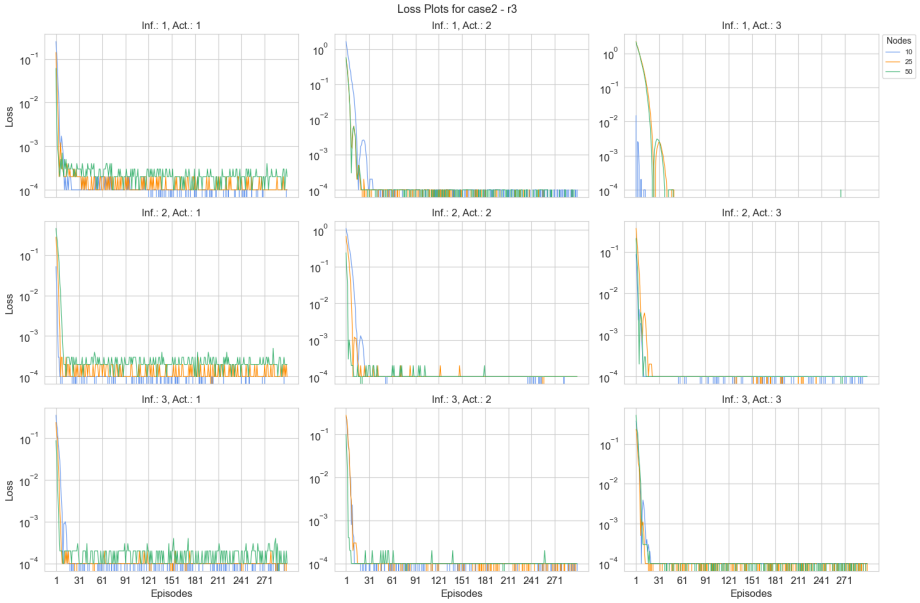

Figure 29: Case-2 using R3: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

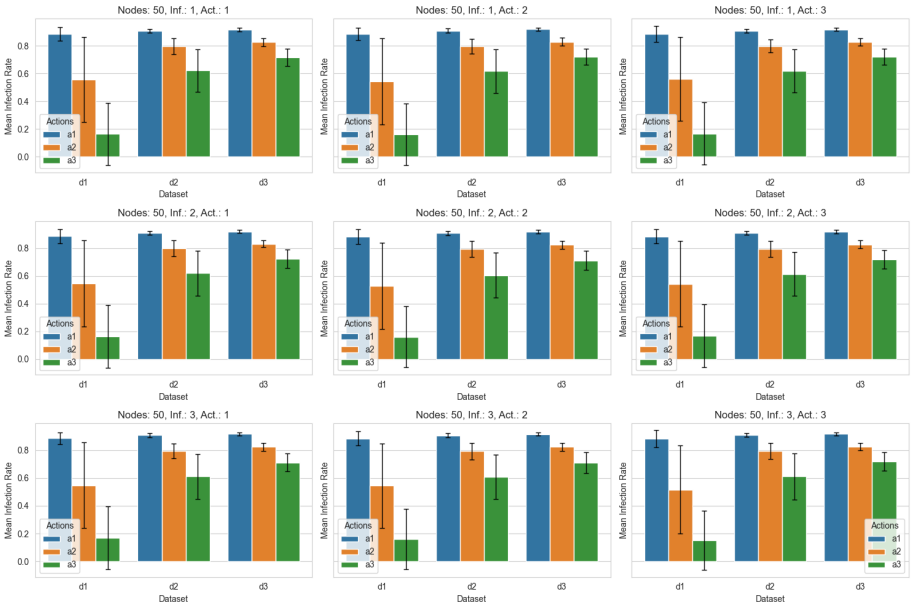

Figure 30: Case-2 using R3: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R4 The loss plot is presented in Figure 31. Dataset v1 Inference Result: 50 Nodes - Figure 32.

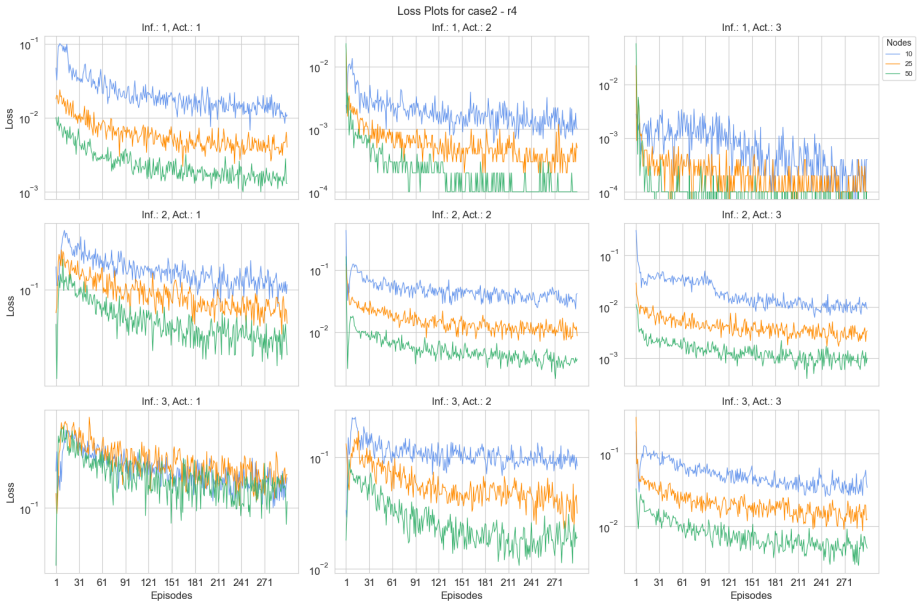

Figure 31: Case-2 using R4: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

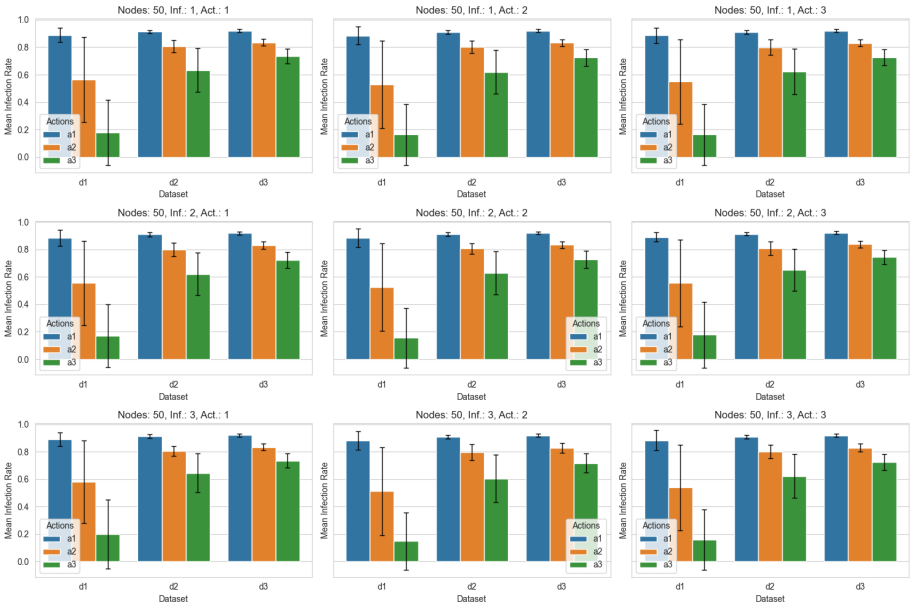

Figure 32: Case-2 using R4: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Dataset v2 Results

Degree of connectivity 1 50 Nodes - Figure 33

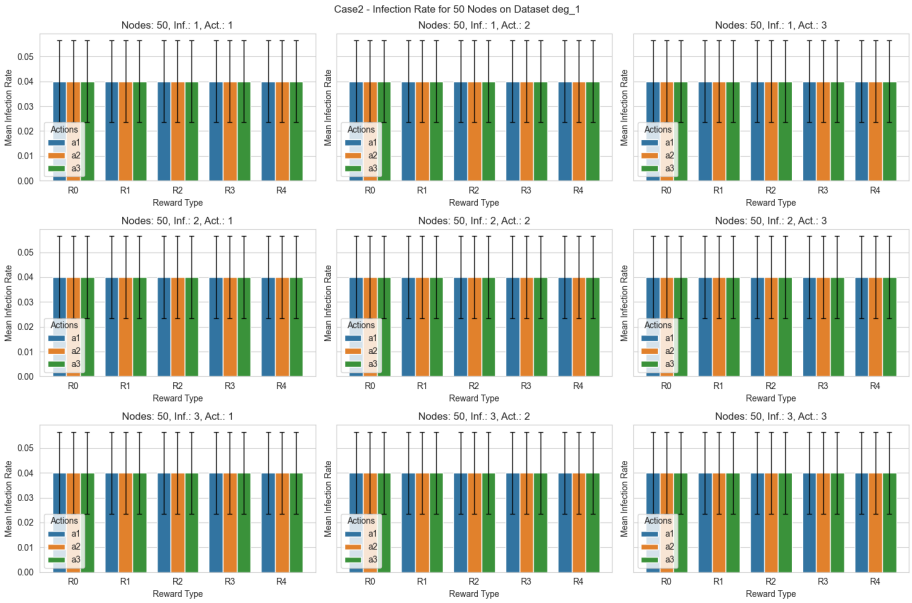

Figure 33: Case-2 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 1, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 2 50 Nodes - Figure 34

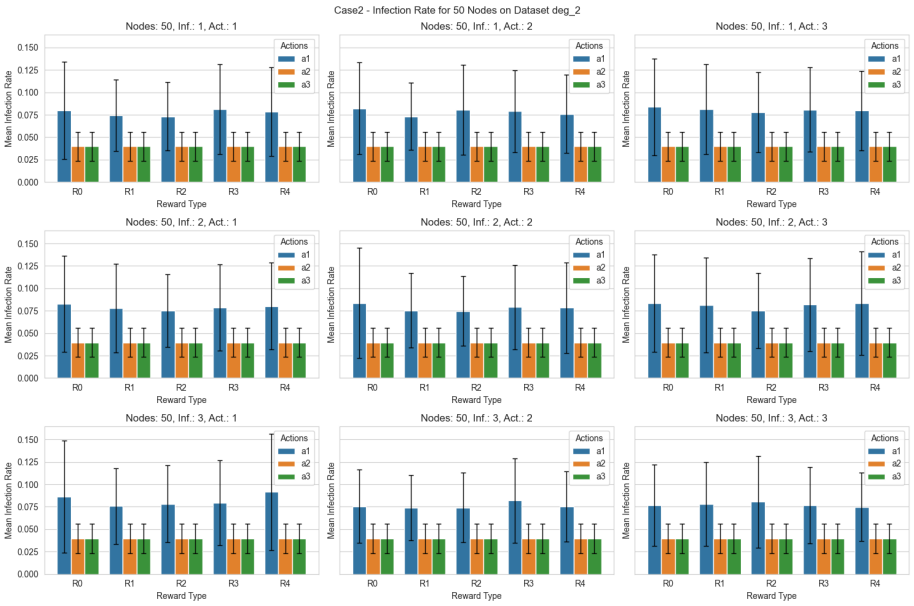

Figure 34: Case-2 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 2, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 3 50 Nodes - Figure 35

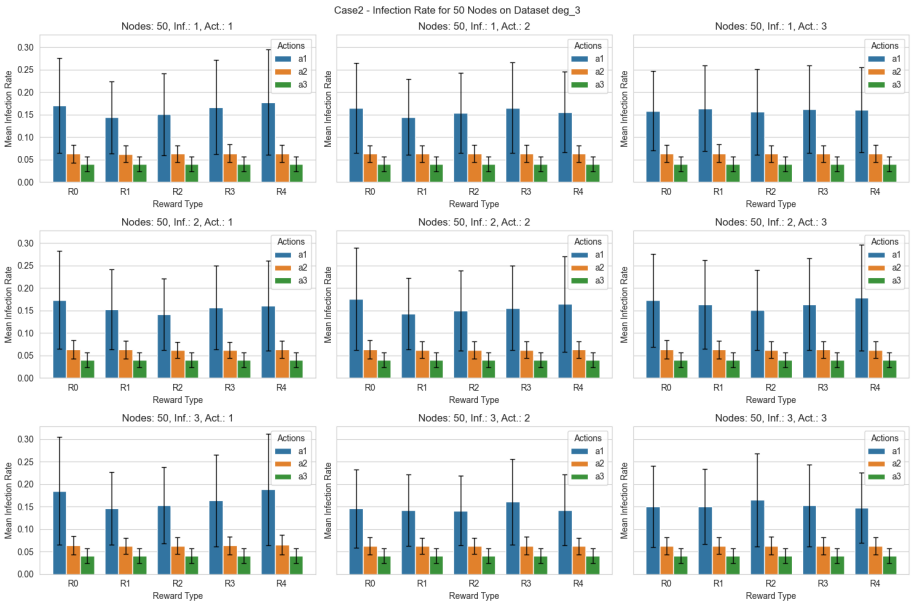

Figure 35: Case-2 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 3, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 4 50 Nodes - Figure 36

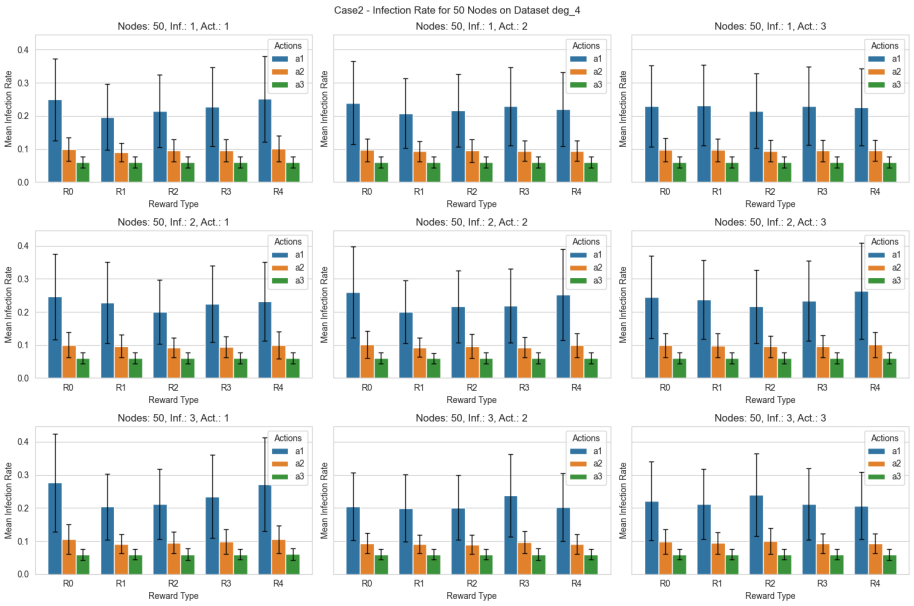

Figure 36: Case-2 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 4, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Case-3

v1

- **Type:** Floating Point Opinion and Floating Point Trust.
- **Opinion Dynamic Model:** Linear Adjustment.

R0 The loss plot is presented in Figure 37. Dataset v1 Inference Result: 50 Nodes - Figure 38.

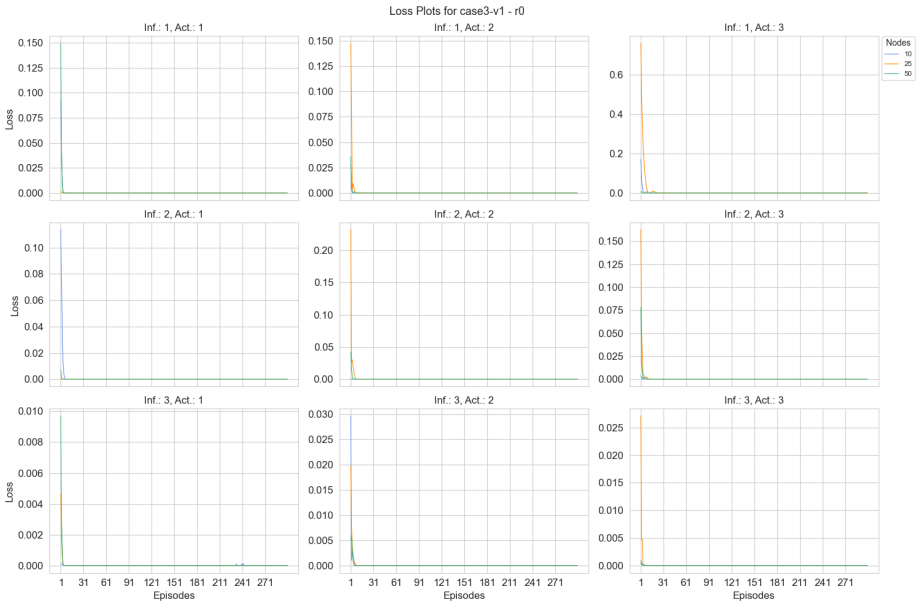

Figure 37: Case-3 v1 using R0: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). The loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

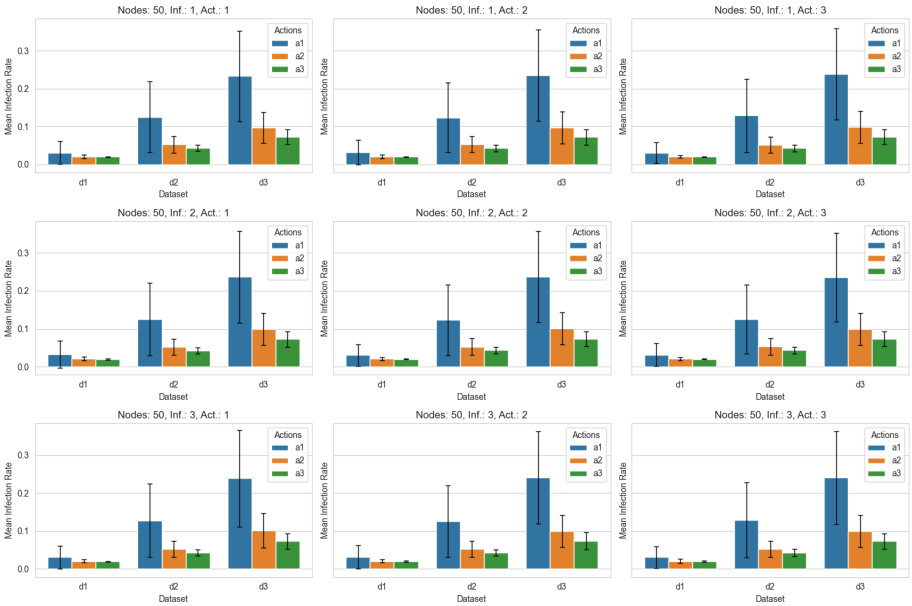

Figure 38: Case-3 v1 using R0: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R1 The loss plot is presented in Figure 39. Dataset v1 Inference Result: 50 Nodes - Figure 40.

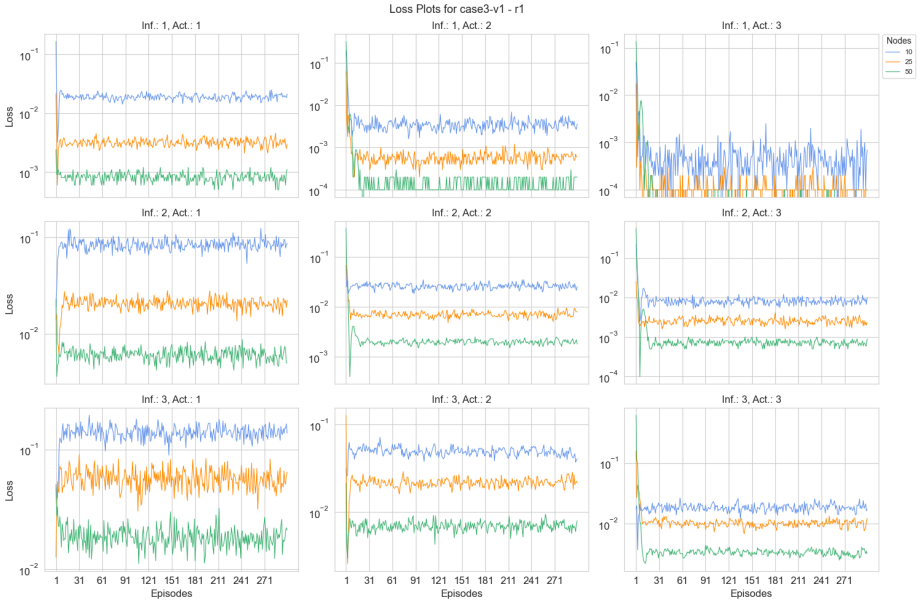

Figure 39: Case-3 v1 using R1: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

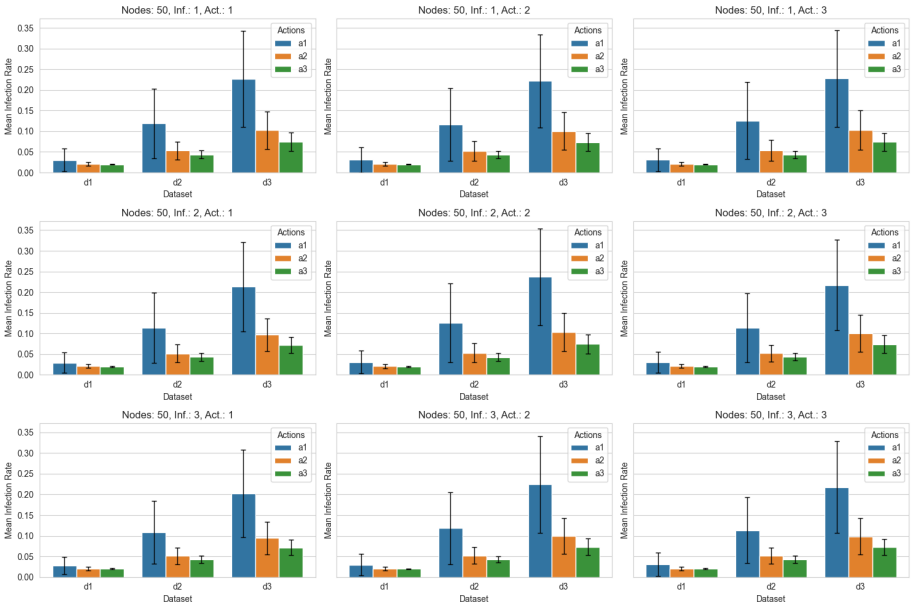

Figure 40: Case-3 v1 using R1: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R2 The loss plot is presented in Figure 41. Dataset v1 Inference Result: 50 Nodes - Figure 42.

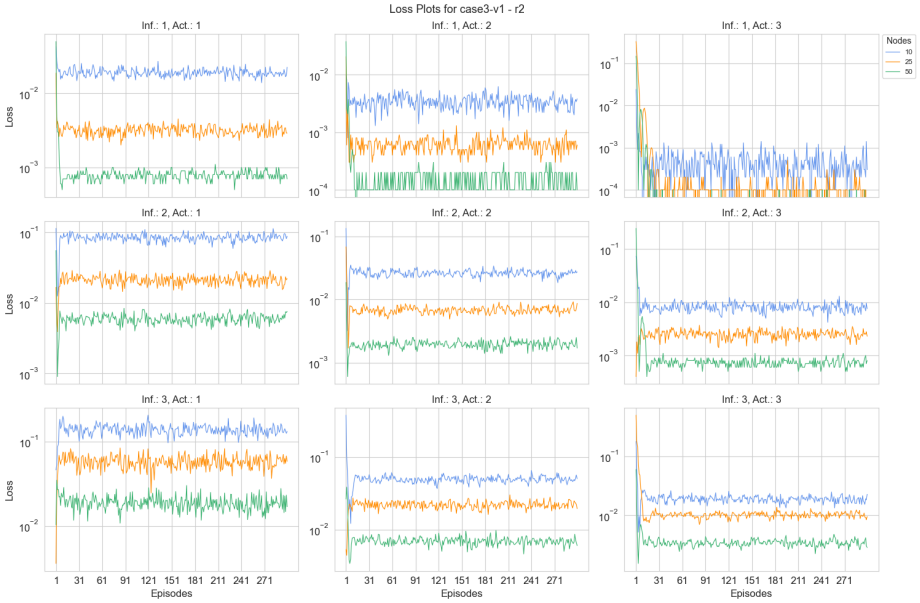

Figure 41: Case-3 v1 using R2: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

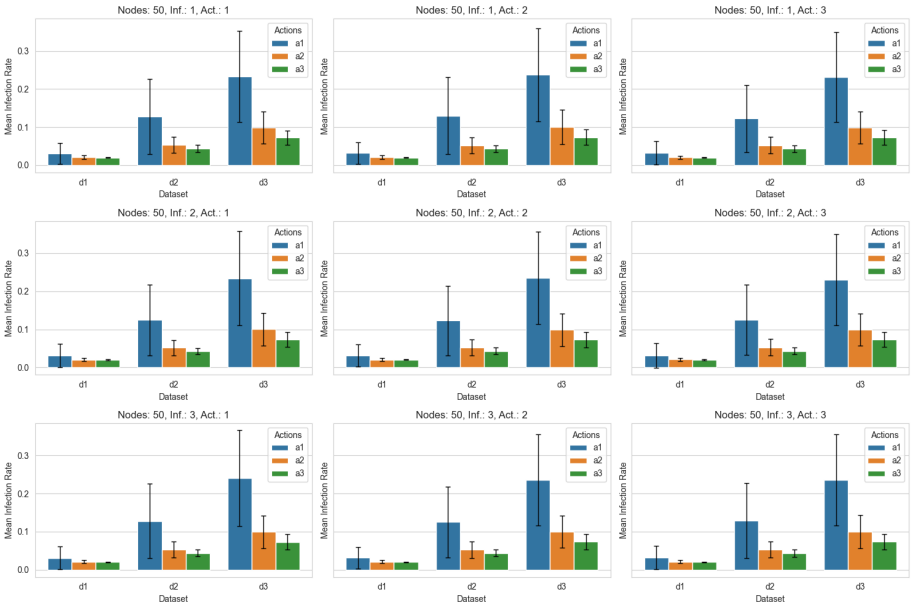

Figure 42: Case-3 v1 using R2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R3 The loss plot is presented in Figure 43. Dataset v1 Inference Result: 50 Nodes - Figure 44.

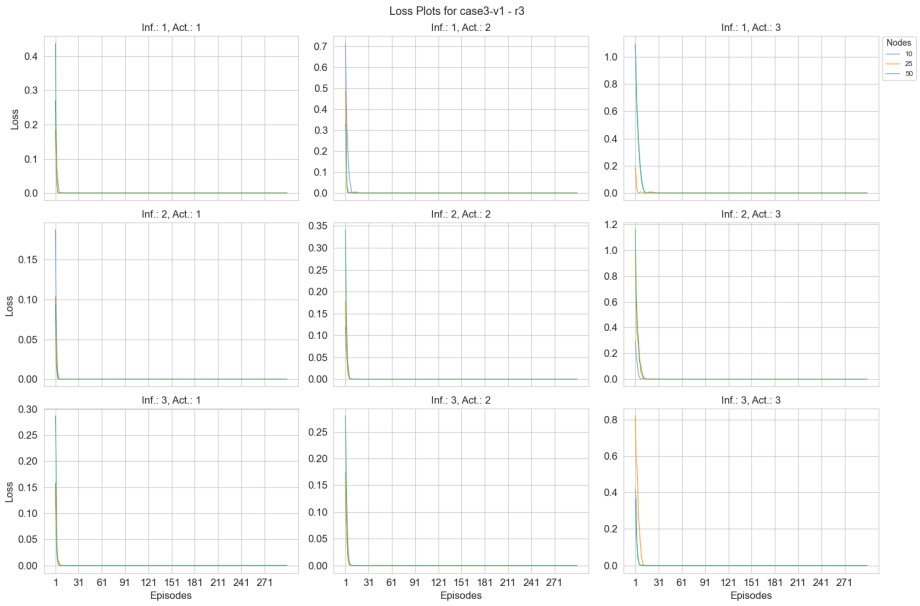

Figure 43: Case-3 v1 using R3: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). The loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

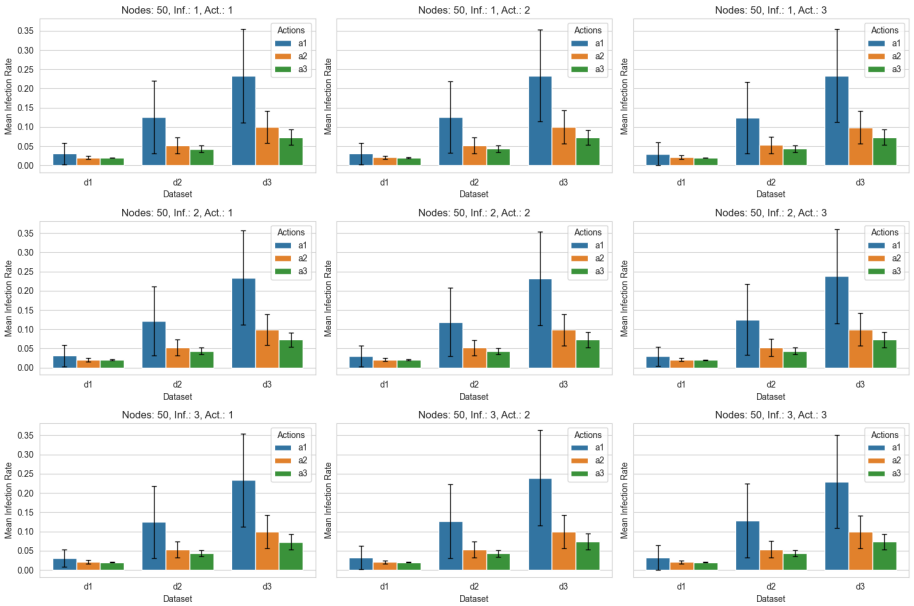

Figure 44: Case-3 v1 using R3: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

R4 The loss plot is presented in Figure 45. Dataset v1 Inference Result: 50 Nodes - Figure 46.

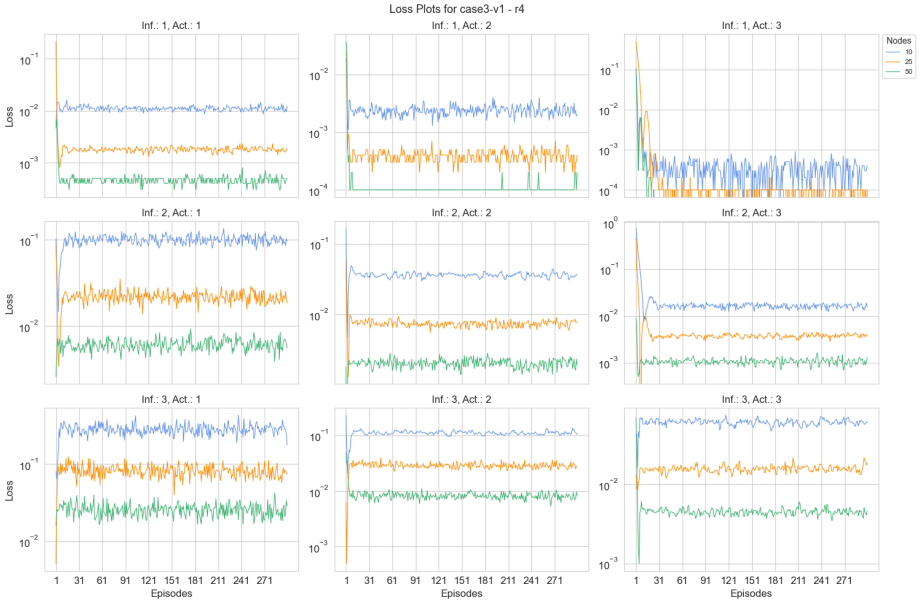

Figure 45: Case-3 v1 using R4: Training MSE loss evolution for RL policy using ResNet model across networks of 10 (blue), 25 (orange), and 50 (green) nodes, for varying initial misinformation sources (Inf.) and action budgets (Act.). Plotted on a logarithmic scale, the loss decreases over episodes, indicating improved policy performance and adaptation across network sizes.

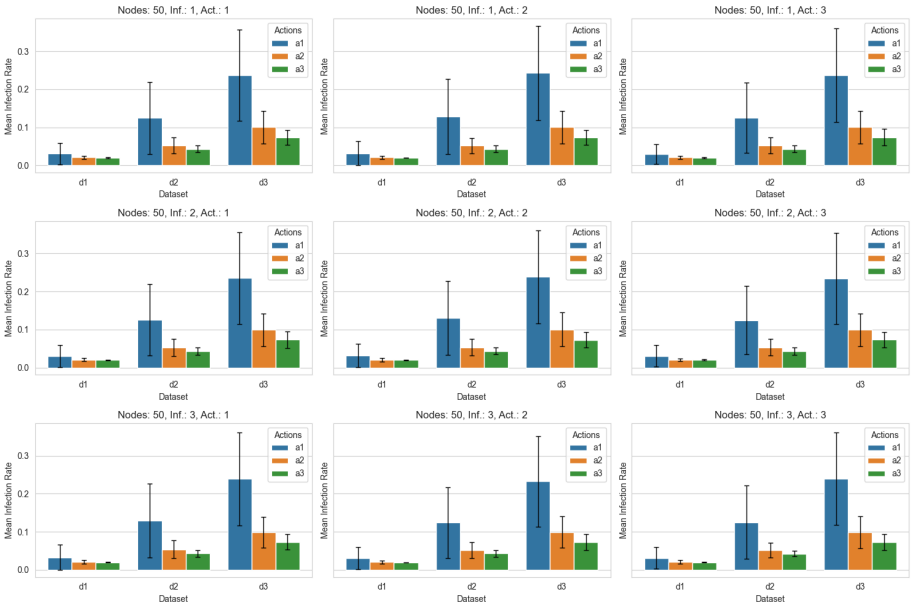

Figure 46: Case-3 v1 using R4: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for datasets d1, d2, and d3 of Dataset v1 type, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Dataset v2 Results

Degree of connectivity 1 50 Nodes - Figure 47

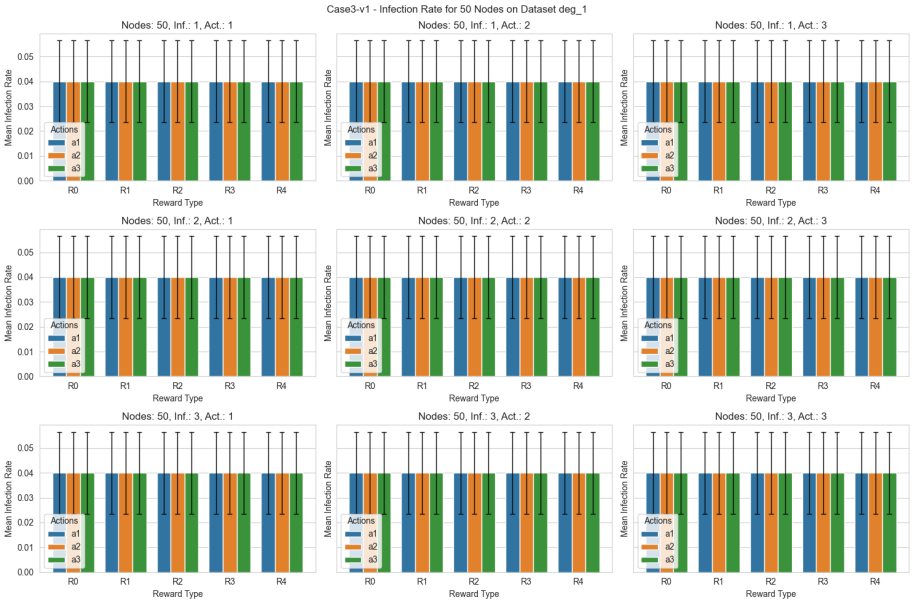

Figure 47: Case-3-v1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 1, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 2 50 Nodes - Figure 48

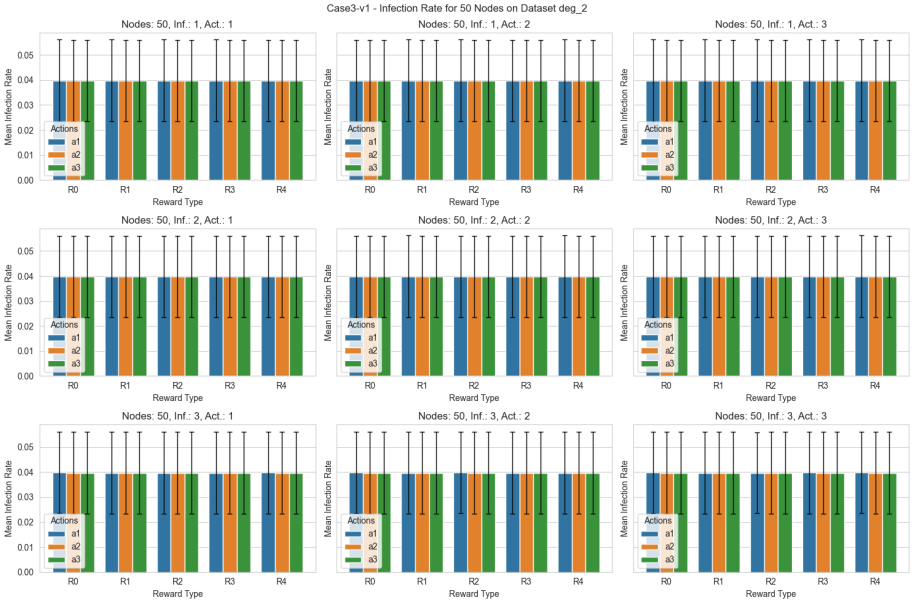

Figure 48: Case-3-v1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 2, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 3 50 Nodes - Figure 49

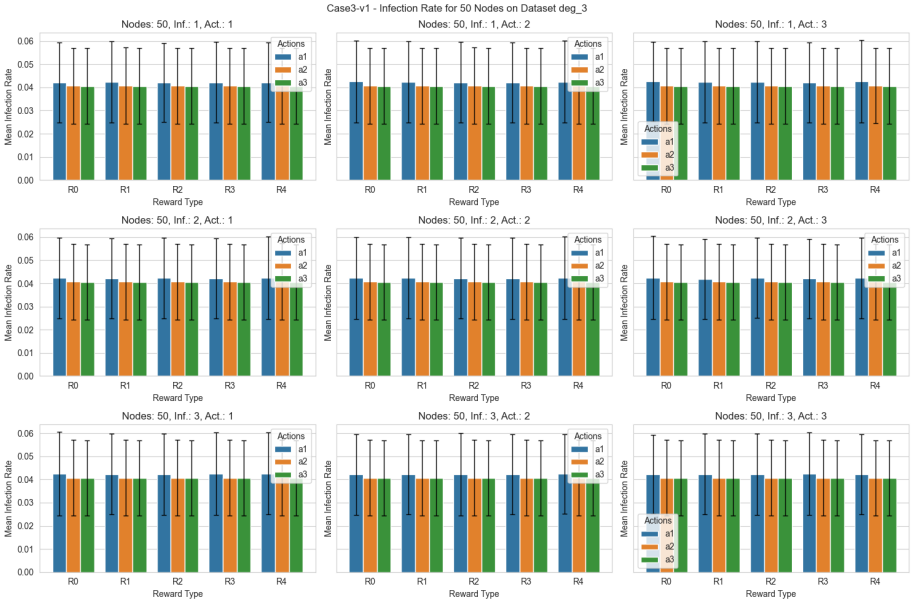

Figure 49: Case-3-v1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 3, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

Degree of connectivity 4 50 Nodes - Figure 50

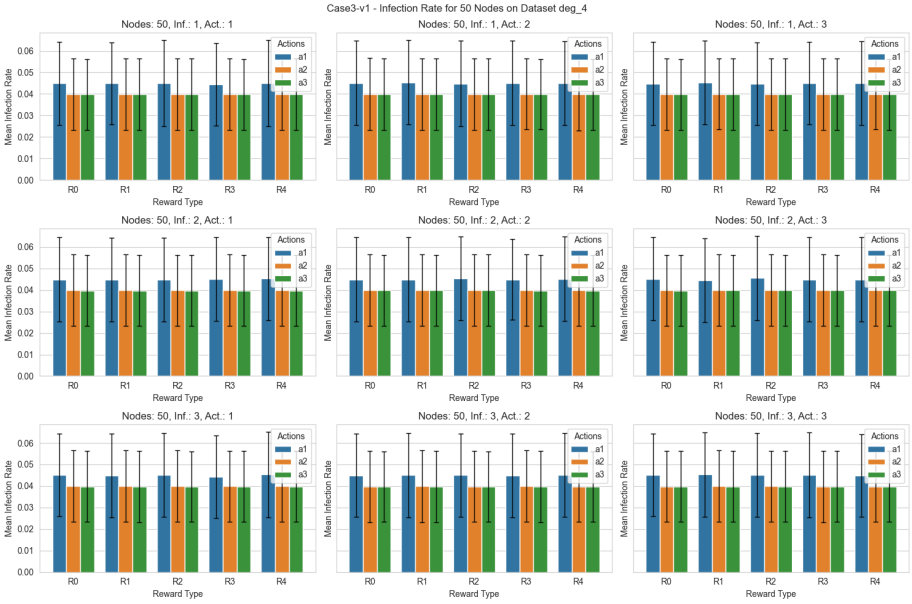

Figure 50: Case-3-v1 inference on Dataset v2: Performance comparison of the RL policies trained using ResNet model for a 50-node network. The barplot displays the mean infection rate for different reward functions on Dataset v2 of Degree of Connectivity 4, differentiated by the number of initial misinformation sources (Inf.) and action budgets (Act.: a1, a2, a3). Each subplot illustrates the performance of a policy trained with the parameters indicated in its title. Lower infection rates indicate more effective policy learning and misinformation containment.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract are properly explained and proved in the main paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A discussion on limitations is provided in the Conclusion (Section 6)

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The different experiment setups along with the details of generating training and testing data are clearly provided in the main paper with additional details in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We uploaded a `zip` file for our code and datasets used in our study as supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide these expressive details in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the mean-variance error bars for our results and these are presented in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware details are provided in the Appendix

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We followed the ethical guidelines properly.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In conclusion (Section 6)

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes MIT license.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide detailed documentation of our code and datasets as a `zip` file.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.