A Appendix

A.1 Dataset Details

We use three graph models: ER random graph, Barabási-Albert (BA) graph Albert & Barabási (2002) and Random Geometric Graph (Geometric or RGG) Dall & Christensen (2002). The density of all three ($|E|/\binom{N}{2}$) is adjustable, but BA can produce exact trees. Fixing the number of nodes to $N = 1,000$, we first obtain one random instance of tree BA, and dense BA, ER and Geometric graphs with $|E| \approx 10,000$ using the `NetworkX` library (NetworkX developer team, 2014) and then use `NDLib` (Rossetti et al., 2017) to simulate SIR and SEIR epidemic dynamics on the graph (supp. A.1). For each sample graph, we pick a P0 seed node i at random to be the patient zero at time $t = 0$ and then we run S(E)IR a fixed number of steps T. The epidemic parameters (α, β, γ) are chosen such that we can vary R_0 to study model performance. We set $\gamma = 0.4$ and $\beta = R_0\gamma/\lambda_1$ where λ_1 is the largest eigenvalue of the graph. For SEIR, we set $\alpha = 0.5$. We generate $20,000$ simulations and use $80 - 10 - 10$ train-validation-test split. For each sample we select $t \in \{1, \cdots T\}$ uniformly at random and try to predict P0 at time $t = 0$ given the graph adjacency matrix A and node features x_i^t.

Table 1 describes the details of the synthetic datasets.

Table 1: Description of the sampled graph statistics

Dataset	**# of Nodes**	**# of Edges**	**Density**	**Diameter**
BA-Tree	1,000	999	0.99	19
BA-Dense	1,000	9,900	9.90	4
Geometric	1,000	9,282	9.28	21
ER-Dense	1,000	9,930	9.93	4

A.2 Training and Hardware

We train the model with an ADAM optimizer for 150 epochs with an initial learning rate of 0.003 and decay the learning rate by 0.5 when the validation loss plateaus with a patience of 10 epochs. We perform hyperparameter tuning over a validation set with a random search strategy. We sweep over the hyperparameter space and track our experiments using Weights and Biases Biewald (2020) choosing the model with the lowest validation error. We run our experiments on Nvidia 2080Ti GPUs and report performance averaged over 4 random seeds.

A.3 Hyper-parameter Details

Table 2: Description of hyper-parameters used. All of our models have been trained with 4 random seeds. The initial learning rate is mentioned in the table below and additionally we decay the learning rate by 0.5 with a patience of 10 epochs when the validation error plateaus. Note that `GAT` had 4 attention heads and has been trained with 5 layers due to a limitation on GPU memory.

Hyperparameters	`GCN-S`	`GCN-R`	`GCN-M`	`GAT`
Number of Epochs	150	150	150	150
Batch Size	128	128	128	32
GNN Hidden Dim	128	128	128	128
Dropout	0.265	0.265	0.265	0.265
Number of GNN Layers	10	10	10	5
Initial Learning Rate	0.0033	0.0033	0.0033	0.004

A.4 Notes on DMP implementation

We include DMP Lokhov et al. (2014) as a baseline against our proposed GNN based method. As DMP does not have code that is publicly available, we implemented DMP using Python for a fair comparison with GNNs. Accordingly, our implementation of DMP uses DGL Wang et al. (2019) which enables us to vectorize belief propagation (BP) and marginalization and now it runs in parallel for all nodes.

Given a graph $G(V, E)$, we observe O^t as the state of the graph with nodes $i \in V$. DMP employs MLE estimation to determine the node i_{P0} that may have led to the observed snapshot O. For a single sample in our dataset D, we use algorithm 1. In order to implement DMP efficiently, we implemented it as a message-passing on a graph using DGL. We sequentially initialize node and edge features for all node i and then as we obtain $N = |V|$ set of graphs with node i acting as P0 in G_i. DMP then allows us to obtain $i = \text{argmax}_i P(O|i)$. The advantage of our implementation then is that we can process all N graphs in parallel as if it were one large graph with N^2 nodes and E^2 edges thanks to DGL's support for batching graphs. A salient feature of using DGL is that the message passing framework allows us to additionally process all the nodes and edges for a single time step t in parallel. The nature of BP algorithms do not allow us to do away with the for-loop over time $\mathbf{t}$ and that remains the only sequential aspect of our implementation. Finally, we use algorithm 1 to process each sample in our test set sequentially. It should be noted that we can further vectorize over a batch of samples in our test set. However, the memory required for DMP is $O(bN^2E^2)$ with b being the size of the batch and so memory requirements quickly blow up. Accordingly, we leave this aspect of implementation for future work.

Algorithm 1: Dynamic Message Passing given graph G, snapshot O and time $\mathbf{t}$

```
for $i \in V$ do
    set node $i$ to be P0
    initialize node features and edge features with eq (12, 13) in DMP;
    for ($t = 0;\ t < \mathbf{t};\ t = t + 1$) do
        for $e \in E$ do
            perform message passing with eq (15, 16, 17) in DMP
    for $j \in V$ do
        marginalize and update node states with eq (18, 19, 20) in DMP.
    Calculate $P(O|i)$ with eq 21 in DMP.
return $i = argmax_i P(O|i)$
```

A.5 Effect of varying number of GCN−S layers on top-1 accuracy

Fig. 6 shows the top-1 accuracy of P0 of the GCN−S model for varying number of layers. We do not observe a significant effect coming from the number of layers. This may be due to the accuracy limitations with $t_{\max}$ and cycles affecting all the models equally, and superseding other effects such as the diameter of the graph. Another possible reason may be that the 20,000 samples on a graph of 1,000 nodes has many repetitions of the same P0, resulting in both shallow and deep models memorizing patterns.

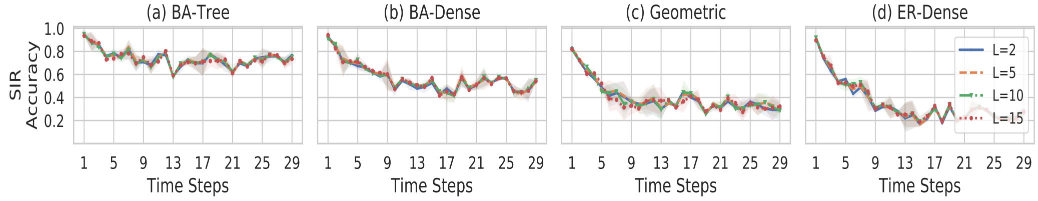


Figure 6: Performance of GCN-S for SIR epidemic dynamics as top-1 accuracy over the test set with varying number of layers.

B Theoretical Analysis

B.1 Early Stage Evolution of SIR and SEIR

The SIR equation on a graph are

$$\frac{dS_i}{dt} = -\beta \sum_j A_{ij} I_j S_i, \qquad \frac{dR_i}{dt} = \gamma I_i, \qquad \frac{dS_i}{dt} + \frac{dI_i}{dt} + \frac{dR_i}{dt} = 0. \tag{11}$$

In very early stages, when $t \ll 1/\gamma$ and $\sum_i I_i + R_i \ll N$, we have $S_i \approx 1$ and we have exponential for I_i because

$$\begin{aligned} \frac{dS_i}{dt} &= -\frac{dI_i}{dt} - \frac{dR_i}{dt} = -\frac{dI_i}{dt} - \gamma I_i \\ \frac{dI_i}{dt} &= \beta \sum_j A_{ij} I_j S_i - \gamma I_i \approx \sum_j \left(\beta A_{ij} - \gamma \delta_{ij}\right) I_j \\ I_i(t) &\approx \sum_j \left(\exp[t\left(\beta A - \gamma \mathbf{I}\right)]\right)_{ij} I_j(0) \end{aligned} \tag{12}$$

Expanding this using the eigen-decomposition $A = \sum_i \lambda_i \psi_i \psi_i^T$ yields eq. (29).

B.2 Transition Probabilities

More generally, when the graph is weighted, the probability of susceptible node i getting infected depends on A_{ij} and the probability of node j being in the infected state. For brevity, define $p_i^\mu(t) \equiv P(x_i^t = \mu)$, with $\mu \in \{S, I, ..., R\}$. The infection probability in SIR (2) can be written as

$$P(x_i^{t+1} = I | x_i^t = S) = 1 - \prod_j \left(1 - \beta A_{ij} p_j^I\right) = \beta \sum_j A_{ij} p_j^I - \beta^2 [A p^I]^2 + O(\beta^3). \tag{13}$$

B.3 Reaction Diffusion Formulation

For brevity, define $p_i^\mu(t) \equiv P(x_i^t = \mu)$. In a network diffusion process the assumption is that node i can only be directly affected by state of node j if there is a connection between them, i.e. if $A_{ij} \neq 0$. This restriction means that the general reaction-diffusion process on a graph has the form

$$F_a(A; p)_i^\mu \equiv \sum_j f_a \Big(g_a(A)_{ij} h_a(p_j)^\mu \Big) \tag{14}$$

$$p_i^\mu(t+1) = F(A; p(t))_i^\mu = \sigma\left(\{F_a(A; p(t))_i^\mu\}\right) \tag{15}$$

With

$$g_a(A)_{ij} = \theta(A_{ij}) \tilde{g}_a(A)_{ij} \qquad h_a(p_i)^\mu = \sigma_a \left(\sum_\nu W_{a,\nu}^\mu p_i^\nu + b_a^\mu \right) \tag{16}$$

where $\theta(\cdot)$ is the step function and $\sigma_a(\cdot)$ a nonlinear function. In regular diffusion on a graph, we have two states S, I and diffusion is changing the $S \to I$ state. The probability $P_{ij} \equiv P(x_i^{t+1} = I | x_j^t = S)$ of node i getting infected at $t+1$, given node j was in the infected state at time t, can be expressed in the form of is determined by the adjacency matrix A_{ij} because node j can only infect its neighbors. The infection probability is given by $p_i^I(t+1) = \beta A_{ij} p_j^I(t)$ and $p_i^S = 1 - p_i^I$. Hence, for diffusion

$$f_1(x) = x \qquad g_1(A) = \beta A, \qquad h_1(p_j)^\mu = \sum_\nu \delta_I^\mu \delta_\nu^I p_j^\nu. \tag{17}$$

In regular diffusion there is no condition on the target node i and even if it is in the I state the dynamics is the same. In the SI model, however, the infection only spreads to i if it is in the S state. Thus, we have to multiply the dynamics by $p_i^S \equiv P(x_i^t = S)$ which yields

$$p_i^I(t+1) = \beta A_{ij} p_j^I(t) p_i^S. \tag{18}$$

This can still be written as (9) by adding the extra functions

$$f_2(x) = x, \qquad g_2(A) = I, \qquad h_2(p_j)^\mu = \sum_\nu \delta^\mu_S \delta^S_\nu p^\nu_j \tag{19}$$

and having

$$p_i(t+1)^I = F_1(A;p(t))^I_i F_2(A;p(t))^S_i \tag{20}$$

where $F_a = f_a(g_a \cdot h_a)$ are as in (14). More complex epidemic spreading models such as SIR and SEIR can also be written in a similar fashion. In SIR and SEIR the rest of the dynamic equations are linear and do not involve the the graph adjacency A at all, meaning $g_a(A) = I$ in the rest of the equations.

B.4 Discrete Time Agent-based SIR as a Reaction Diffusion System

The agent-based models (2) and (3), which correct for double-counting of infection from multiple neighbours, are sometimes written as

$$P(x^{t+1}_i = I | x^t_i = S) = 1 - (1-\beta)^{\xi_i}, \tag{21}$$

where ξ_i is the total number of neighbors j of i which are infected, meaning $x^t_j = I$. We will first show that this is a special case of the form given in our paper. First, note that in (2) the terms can also be written as

$$(1-\beta)^{\xi_i} = \prod_j \left(1 - \beta \delta_{x^t_j, I}\right) \tag{22}$$

In the probabilistic model, we have to replace the strict condition of j being in the I state with its probability, so $\delta_{x^t_j, I} \to P(x^t_j = I) = p^I_j(t)$.

$$P(x^{t+1}_i = I | x^t_i = S) = 1 - \prod_{j \in \partial_i} \left(1 - \beta \hat{A}_{ij} p^I_j\right) \tag{23}$$

and for small β yield

$$P(x^{t+1}_i = I | x^t_i = S) = \beta \sum_j \hat{A}_{ij} p^I_j - \beta^2 \sum_{j,k} \hat{A}_{ij} p^I_j \hat{A}_{ik} p^I_k + O(\beta^3) \tag{24}$$

which yields the simplified equation $p_i(t+1)^I = p^S_i(t) \sum_j \beta \hat{A}_{ij} p^I_j(t)$. Note that if the infection rate per time step β is large $\beta \sum_j \hat{A}_{ij} p_j$ can exceed 1, rendering (24) inconsistent with p^I_i being probabilities. Both (23) and (24) both can be written in the form of RD (15) and (9). We utilize the h_1, g_1 and h_2, g_2 found for diffusion (17) and SI (19)

$$F_1(A;p)^\mu_i = \sum_j \log\left(1 - \beta \hat{A}_{ij} h_1(p_j)^\mu\right) \qquad F_1(A;p)^\mu_i = h_2(p_i)^\mu \tag{25}$$

and defining the probability as

$$\begin{aligned} p^I_i(t+1) = {F_1}^S_i \left(1 - \exp\left[{F_2}^I_i\right]\right) &= p^S_i(t) \left(1 - \prod_j \left(1 - \beta A_{ij} p^I_j(t)\right)\right) \\ &\approx \beta p^S_i(t) \sum_j A_{ij} p^I_j(t) \end{aligned} \tag{26}$$

B.5 Proofs

Proposition 2. *Reaction-diffusion dynamics on graphs is structurally equivalent of the message-passing neural network ansatz.*

Proof: Analyzing the full stochastic model requires closely tracking the individual events and varies in each run. Hence, we will work with mean-field diffusion dynamics using transition probabilities, instead. Denoting $p_i^\mu(t) \equiv P(x_i^t = \mu)$ of node i being in states such as $\mu \in \{S, I, ..., R\}$ at time t, a Markovian reaction-diffusion dynamics can be written as

$$p_i^\mu(t+1) = \sigma\left(\sum_j F\Big(\mathcal{A}_{ij} \cdot h(p_j)^\mu\Big)\right), \qquad h_a(p_i)^\mu = \sigma\left(\sum_\nu W^\mu_{a,\nu} p_i^\nu + b^\mu\right) \tag{27}$$

where $\mathcal{A}^a_{ij} = \theta(A_{ij}) f(A)_{ij}$ with $\theta(\cdot)$ being the step function and $\sigma(\cdot)$ a nonlinear function. To see this, note that RD processes on graphs involve a message-passing (MP) step (e.g. an infection signal coming from neighbors of a node), and a reaction step where messages of different states μ passed to node i interact with each other on node i. RD dynamics such as the SIR and SEIR models are also Markovian and the probability $p_i^\mu(t)$ only depends on the probabilities at $t-1$. These are also the conditions satisfied by MPNN. In (27), $\mathcal{A}$ are a set of propagation rules for the messages, which are only nonzero where A is nonzero, same as the aggregation rule in MPNN. To have interactions between states μ occurring inside each fixed node i, $h(p_i)$ can mix the states μ but not change the node index i, leading to the form of $h(p_i)$ in (27), which is the general ansatz for a neural network with weight sharing for nodes, same as in MPNN, and graph neural networks in general. □

B.5.1 Proof of Theorem 3

Theorem 3 (Time Horizon)**.** *Assume SIR dynamics* (1) *on a connected graph of N nodes, starting with a single patient zero. Denoting the adjacency matrix by A and its largest eigenvalue by λ_1, the average infection probability, both over nodes and choice of patient zero, $\langle I(t) \rangle \equiv \langle \sum_i I_i(t)/N \rangle_{\mathrm{P0}}$ becomes $O(1)$ after $t_{\max}$ time steps given by*

$$t_{\max} \sim \frac{\log N}{\gamma(R_0 - 1)}, \qquad R_0 \equiv \frac{\beta \lambda_1}{\gamma} \tag{28}$$

Proof: Consider the spectral expansion $A = \sum_{a=1}^N \lambda_a \psi^{(a)} \psi^{(a)T}$, with $\lambda_1 > \cdots > \lambda_N$. In (1) early in the disease spreading, all nodes are susceptible, meaning $S_i \approx 1$, $R_i \approx 0$, and $I_i \approx 1 - S_i$. Thus, combining the three SIR equations, keeping only I_i, the infection spreads as Newman (2018)

$$I_i(t) \approx \sum_j \exp\left[t(\beta A - \gamma \mathbf{I})\right]_{ij} I_j(0) \approx \exp\left[(\beta\lambda_1 - \gamma)t\right] \Big(\psi^{(1)} \cdot I(0)\Big) \psi_i^{(1)}, \tag{29}$$

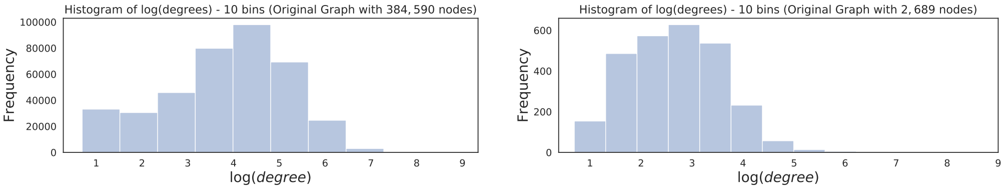
Here, $\mathbf{I}$ is the identity matrix, λ_1 is the largest eigenvalue of A and $\psi^{(1)}$ is the corresponding eigenvector. Averaging over a uniform choice of patient zeros, for the average infection probability we have

$$\langle I(t) \rangle \approx \frac{1}{N} \exp\left[(\beta\lambda_1 - \gamma)t\right] \Big\langle \psi^{(1)} \cdot I(0) \Big\rangle_{\mathrm{P0}} \sum_i \psi_i^{(1)} \geq \frac{1}{N} \exp\left[(\beta\lambda_1 - \gamma)t\right] \tag{30}$$

where we used the inequality between L_1 and L_2 norms to get $\langle \psi^{(1)} \cdot I(0) \rangle_{\mathrm{P0}} = \left\| \psi^{(1)} \right\|_1 \geq \left\| \psi^{(1)} \right\|_2 = 1$. Connectedness means A is irreducible and by the Perron-Frobenius theorem its leading eignvector is positive, hence $\sum_i \psi_i^{(1)} = \left\| \psi^{(1)} \right\|_1 \geq \left\| \psi^{(1)} \right\|_2$. Setting the lower bound of (30) equal to 1 and solving for t we get (28). □

B.5.2 Proof of Theorem 2 P_{tri}

Proof: If P0 is in a triangle, we may miss it $2/3$ of the times. Thus, the probability of detecting P0 is bounded by $P < 1 - P_{tri} \times 2/3$, where P_{tri} is the probability that P0 is in a triangle. Since edges in G are uncorrelated, each having probability p, G_I is also a connected random graph with the same edge probability p. Hence, in G_I all nodes have degree $k \approx p|G_I|$. P_{tri} is one minus the probability that none of the k neighbors of P0 are connected, i.e. $P_{tri} = 1 - (1-p)^{\binom{|G_I|p}{2}}$, which proves the proposition. □



(a) Degree distribution of original network

(b) Degree distribution of subsampled network

Figure 7: Degree distribution of the original co-location network with $384,590$ nodes and the subsampled network with $2,689$ nodes. We subsample the larger network to find a subgraph in order to reduce computational costs of our experiments. We observe that the distribution of our subsampled network is similar to the original graph.

C Covid-19 Data and Simulations

Geolocation data Mobility data are provided by Cuebiq, a location intelligence and measurement platform. Through its Data for Good program (https://www.cuebiq.com/about/data-for-good/), Cuebiq provides access to aggregated and privacy-enhanced mobility data for academic research and humanitarian initiatives. These first-party data are collected from users who have opted in to provide access to their GPS location data anonymously, through a GDPR-compliant framework. Additionally, Cuebiq provides an estimate of home and work census areas for each user. In order to preserve privacy, noise is added to these "personal areas", by upleveling these areas to the Census block group level. This allows for demographic analysis while obfuscating the true home location of anonymous users and preventing misuse of data.

Colocation network The method for constructing the co-location graphs is as follows. First, we split each day into five minute time windows, resulting in 288 time bins per day. For every location event, we use its timestamp to assign it to a time bin, then assign the longitude-latitude coordinate of the observation to an 8-character string known as a *geohash*. A geohash defines an approximate grid covering the earth, the area of which varies with latitude. The largest dimensions of an 8-character geohash are 38m x 19m, at the equator. If a user does not have an observation for a given time bin, we carry the last observation forward until there is another observation. We finally define two users to be co-located — and therefore to have a timestamped edge in the graph — if they are observed in the same geohash in the same time bin. Accordingly, our co-location graph is constructed by observing the greater Boston area over two weeks from 23 March, 2020 to 5 April, 2020 and results in a graph with $N = 384,590$ nodes. To reduce computational costs, we sample a subgraph with $N = 2,689$ nodes and $|E| = 30,376$ edges with similar degree distribution and connectivity patterns as the original graph and can be observed in Fig 7.

Epidemic simulations in real data. We run a SEIR model on the real co-location network. In doing so, we select parameters and modify the structure of the model to resemble the natural history of COVID-19 Chinazzi et al. (2020). At each time step nodes, according their health status, can be in one of five compartments: S, E, I, I_a, or R. Thus, we split infectious nodes in two categories. Those that are symptomatic (I) and those that are asymptomatic (I_a). The first category infects susceptible node, with probability λ per contact. The second category instead with probability $r_a\lambda$. We set $r_a = 0.5$ and consider that probability of becoming asymptomatic as $p_a = 0.5$. The generation time, that is the sum of incubation (α^{-1}) and infectious period (γ^{-1}), is set to be 6.5 days. Specifically, we fix $\alpha^{-1} = 2.5$ and $\gamma^{-1} = 4$ days. In a single, homogeneously mixed, population the basic reproductive number of such epidemic model is $R_0 = (1 - p_a + r_a p_a)\beta/\gamma$ where β is the per capita spreading rate Keeling & Rohani (2011). Here however, the epidemic model unfolds on top of the real co-location network. Hence, infected nodes are able to transmit the disease only via contacts (with susceptible individuals) established during the observation period. As mentioned above, the value of R_0 is defined by the interplay between the disease's parameters as well as the structural properties of the network Pastor-Satorras et al. (2015); Masuda & Holme (2017). For simplicity we approximate $\beta = \langle k\rangle\lambda$, where $\langle k\rangle$ is the average number of connections in the network. We obtain $\lambda = 0.073$ after solving for R_0 and plugging in $\langle k\rangle = 30376/2689 = 11.29$. The simulations start with an initial infectious seed selected uniformly at random among all nodes. We then read and store the time-aggregated network in memory. The infection dynamics, which are catalysed by the contacts between infectious and nodes, take place on such network. The spontaneous transitions instead (i.e. transition from S to E and the recovery process), take place independently of the connectivity patterns. After the infection and recovery dynamics, we print out the status, with respect to the disease, of each node. Finally, we create a dataset with $10,000$ samples and an $80-10-10$ train-validation-test split.