Deep learning of stochastic contagion dynamics on complex networks
— Supplementary Material —

Charles Murphy, Edward Laurence, and Antoine Allard
Département de Physique, de Génie Physique, et d'Optique,
Université Laval, Québec (Québec), Canada G1V 0A6 and
Centre interdisciplinaire en modélisation mathématique,
Université Laval, Québec (Québec), Canada G1V 0A6

(Dated: June 19, 2020)

CONTENTS

I. MODEL

A. Details of the Architecture

We propose the graph neural network (GNN) architecture shown in Fig. 1. First, we apply a sequence of input layers to the state of each node, denoted $x_i(t)$, individually to transform them into features. A single layer l with f_l features is composed of a linear transformation and a rectified linear unit activation function (ReLU, expressed as $\mathrm{ReLU}(x) = \max\{x, 0\}$) and acts on a feature vector $\boldsymbol{u}_i^{l-1}$ such that

$$\boldsymbol{u}_i^l = \mathrm{ReLU}\left(\mathbf{W}^l \boldsymbol{u}_i^{l-1} + \boldsymbol{b}^l\right) \tag{1}$$

where $\mathbf{W}^l \in \mathbb{R}^{f_l \times f_{l-1}}$ and $\boldsymbol{b}^l \in \mathbb{R}^{f_l}$ are trainable parameters. After these first layers, the resulting feature vectors are aggregated using a graph attention module inspired by Ref. [1] (Description in Sec. I B). In short, the graph attention module, illustrated in blue in Fig. 1, is a trainable nonlinear function, denoted $\mathcal{A}$, which combines the node feature vectors using the adjacency matrix $\mathbf{A}$. This layer returns a set of new feature vectors describing *both* the state of nodes and the states of their first neighbors. The aggregated features are then processed into a set of final feature vectors, namely $\boldsymbol{v}_i \in \mathbb{R}^{|\Omega|}$, with another sequence of output layers composed of linear transformations and a ReLU activation functions [similar to Eq. (1)]. From the aggregated and transformed feature vectors $\boldsymbol{v}_i$, we finally compute a discrete probability distribution, using a Softmax function, corresponding to the local transition probabilities (LTPs) of the GNN, where each probability is expressed as following:

$$\hat{p}_{i;\nu} = [\mathrm{Softmax}(\boldsymbol{v}_i)]_\nu = \frac{\exp([\boldsymbol{v}_i]_\nu)}{\sum_{\xi=1}^{|\Omega|} \exp([\boldsymbol{v}_i]_\xi)} \,. \tag{2}$$

Recall that each entry $\hat{p}_{i;\nu}$ of this probability distribution corresponds to the probability that node i transition to state ν, which is conditioned on its state $x_i(t)$ and the state of its first neighbors $x_{\mathcal{N}_i}(t)$ thanks to the GNN architecture. Using the notation that we used in the main paper, we get the following equivalence:

$$\hat{p}_{i;\nu} = \hat{p}_{i;\nu}(t) \equiv \hat{p}^{x_i(t)\to\nu}_{x_{\mathcal{N}_i}(t)} \,. \tag{3}$$

We summarize the architecture of the GNN models for each contagion dynamics we used in the main paper in Tab. I.

B. Attention Mechanism

Our attention mechanism is a variant of the recently introduced graph attention networks (GAT) [1] and corresponds to a nonlinear trainable function $\mathcal{A}$ that combines the feature vectors of the nodes with respect

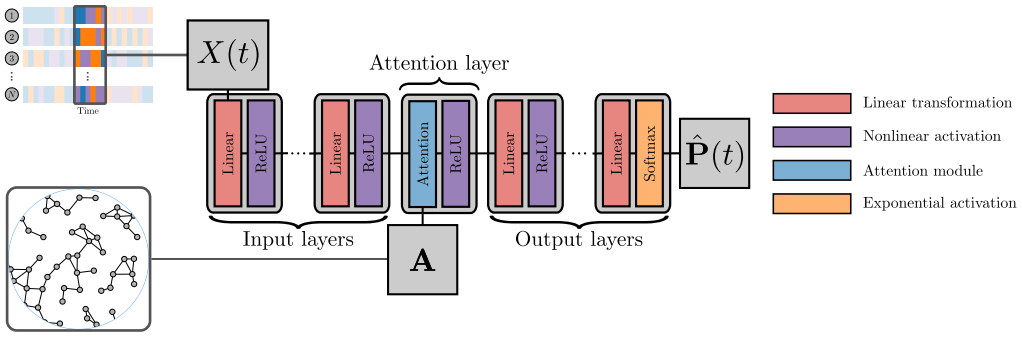

FIG. 1. (color online) **Illustration of the GNN architecture**. The blocks of different colors represent a GNN operations. The red blocks correspond to trainable linear transformation parametrized by weights and biases (see text). The purple blocks represent non-trainable activation function between each layer. The core of the model is the attention module [1], which is represented in blue. The orange block at the end is an exponential Softmax activation that transforms the output into properly normalized transition probabilities.

Dynamics	Simple contagion	Complex contagion	Interacting contagion
Input layers	$\mathrm{Linear}(1,32)$ ReLU $\mathrm{Linear}(32,32)$ ReLU	$\mathrm{Linear}(1,32)$ ReLU $\mathrm{Linear}(32,32)$ ReLU	$\mathrm{Linear}(1,32)$ ReLU $\mathrm{Linear}(32,32)$ ReLU $\mathrm{Linear}(32,32)$ ReLU
Number of attention layers	1	1	2
Output layers	$\mathrm{Linear}(32,32)$ ReLU $\mathrm{Linear}(32,2)$ Softmax	$\mathrm{Linear}(32,32)$ ReLU $\mathrm{Linear}(32,2)$ Softmax	$\mathrm{Linear}(32,32)$ ReLU $\mathrm{Linear}(32,32)$ ReLU $\mathrm{Linear}(32,4)$ Softmax
Number of parameters	2307	2307	6630

TABLE I. **Layer by layer description of the GNN models for each contagion dynamics.** For each sequence, the operations are applied from top to bottom. The operations represented by $\mathrm{Linear}(m,n)$ correspond to linear transformations of the form $f(\boldsymbol{x}) = \mathbf{W}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{x} \in \mathbb{R}^m$ is the input, $\mathbf{W} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{b} \in \mathbb{R}^n$ are trainable parameters. The operations ReLU and Softmax are activation functions given by $\mathrm{ReLU}(x) = \max\{x, 0\}$ and $\mathrm{Softmax}(\boldsymbol{x}) = \frac{\exp(\boldsymbol{x})}{\sum_i \exp(x_i)}$.

to the adjacency matrix $\mathbf{A}$. It works by assigning an attention coefficient α_{ij} that modulates the effect of the node j features over the node i resulting features. Considering the pair of connected nodes (i, j) whose input feature vectors are respectively $\boldsymbol{u}_i^l$ and $\boldsymbol{u}_j^l$, we compute the attention coefficient α_{ij} using the following expression:

$$\alpha_{ij} = \sigma\left[\mathbf{W}^{\mathcal{A}}\left(\boldsymbol{u}_i^l \middle|\middle| \boldsymbol{u}_j^l\right) + \boldsymbol{b}^{\mathcal{A}}\right] \tag{4}$$

where

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{5}$$

is the logistic function, $\mathbf{W}^{\mathcal{A}} \in \mathbb{R}^{1\times 2f_l}$ and $\boldsymbol{b}^{\mathcal{A}} \in \mathbb{R}$ are additional trainable parameters corresponding to a linear transformation and $\cdot||\cdot$ denotes a concatenation. The logistic function acts as an activation function guaranteeing that the attention coefficients range between 0 (no attention) and 1 (full attention). The attention coefficients are then used to compute the aggregated features $\boldsymbol{v}_i$ as follows:

$$\boldsymbol{v}_i = \left[\mathcal{A}\Big(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N; \mathbf{A}\Big)\right]_i = \boldsymbol{u}_i + \sum_{j\in\mathcal{N}_i} \alpha_{ij}\boldsymbol{u}_j\,. \tag{6}$$

Note that our attention mechanism is different from the original GAT from Ref. [1] in two ways. First, in the original paper, they used a Softmax function in Eq. (4) instead of a logistic function, which constrains the attention coefficients to be normalized, i.e. $\sum_j \alpha_{ij} = 1$. Under this constraint, Eq. (6) becomes an averaging operation rather than a general weighted sum. We argue that this normalization constraint is detrimental in the case of learning contagion dynamics specifically, because it does not capture the individual contribution of each neighbors in an absolute way. Rather, it captures the relative contributions of the neighbors with respect to each other—the features of an "average" neighbor—which is more desirable in network embedding tasks [2]. This difference also allows us to enforce a complete self-attention ($\alpha_{ii} = 1$), as one can deduce from Eq. (6). Second, contrary to the original GAT architecture, we do not apply an activation function directly after the linear transformation in Eq. (4). The reason for this is that negative-valued features are, for us, of great importance, because they lead to small attention coefficients. Applying a nonlinear activation function, e.g. a ReLU function, inhibits the negative values and, consequently, prevents the attention coefficients to vanish when needed.

In Ref. [1], they also showed how using multiple attention layers in parallel and in series can be beneficial to network embedding tasks. We also consider using attention layers in parallel in our GNN architecture. To do so, we apply multiple attention layers, denoted $\mathcal{A}^q$ with $q = 1..Q$, on each input feature vectors $\boldsymbol{u}_i$, and concatenate the resulting outputs in a single feature vector $\boldsymbol{v}_i$ as follows:

$$\boldsymbol{v}_i = \Big|\Big|_{q=1}^{Q}\left[\mathcal{A}^q\Big(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N; \mathbf{A}\Big)\right]_i \tag{7}$$

Name	Number of nodes	Number of edges	Timespan [s]	Reference
Hypertext 2009	113	2196	212341	[3]
Thiers13	328	43496	380421	[4]
SFHH	403	73557	106541	[4]
LH10	72	1381	259181	[4]
InVS13	95	3915	993561	[4]
InVS15	219	16725	993561	[4]
HS13	327	5818	363561	[5]

TABLE II. **Properties of social contact networks.** We show the number of nodes, the total number of edges observed and the timespan (in seconds) of these time-varying networks. In Refs. [3–5], they collected these networks by monitoring the proximity of pairs of individuals: If two individuals are in close proximity for at least 20 seconds, a contact (or an edge) is recorded at this time.

During experiments, we also considered applying attention layers in series, but it turns out that in our case using more than one attention layer leads to overfitting. Indeed, recall that the first attention layer aggregates the node features with respect to their first neighbors. Therefore, applying a second attention layer indirectly aggregate the features of the second neighbors, which in turn assumes that their contribution matters in the computation of the LTPs. Because only the first neighbors actually contribute to the true LTPs, it forces the GNN model to unwind this assumption during its training, which is actually difficult to do and might explain the overfitting.

II. PROJECTIONS ON REAL-WORLD TEMPORAL NETWORKS

We investigate our GNN architecture capabilities when used on real-world time-varying networks, after they are trained on synthetic data. More precisely, we apply our GNN models on 7 different temporal network datasets, namely networks of social contacts in conferences (labeled Hypertext 2009 and SFHH), workplaces (labeled InVS13 and InVS15), schools (labeled Thiers13 and HS13) and hospitals (labeled LH10) [3–5]: A more detailed description of these datasets is shown in Tab. II. These datasets contain timestamped edges in the form (t_{ij}, i, j), where t_{ij} corresponds to the time at which edge (i, j), representing an active contact between individuals i and j, was observed.

In the main paper, we consider two coarse-grained representations of these temporal networks. For the first one, we aggregate all edges at any given times into a single and static network. We use this more convenient representation in Fig. 2 to compute the steady-state distribution of node states. For each dynamics, we sample 100 points from the LTPs of the GNN models (same model as Fig. 3 of the main

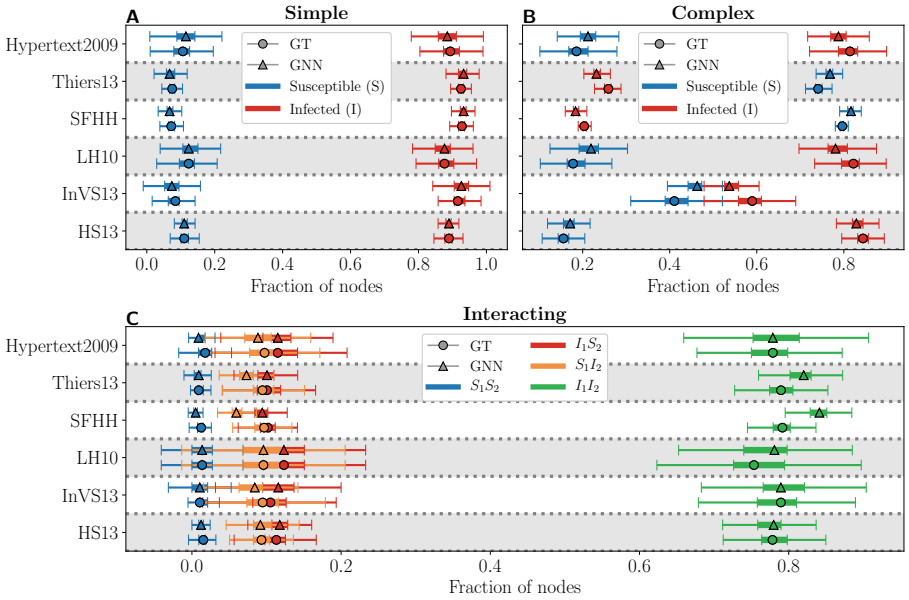

FIG. 2. **Steady-state distributions of (a) the simple, (b) the complex and (c) the interacting contagion dynamics on real-world proximity networks in conferences (labeled Hypertext 2009, SFHH), workplaces (labeled InVS13 and InVS15), schools (labeled Thiers13 and HS13) and hospitals (labeled LH10) [3–5].** The steady-state distribution as generated by the LTPs of the GNN models are indicated by the triangles; those generated by the LTPs of the original dynamics (labeled as GT for "ground truth") are indicated by the circles. The colors of the symbols indicate the state corresponding to the fraction of nodes.

paper) and the LTPs of the original dynamics for comparison, and display the resulting distributions using boxplots.

For the second coarse-grained representation, we aggregate edges in time windows of 12 hours. For instance, a network for which edges are timestamped across 24 hours would result in two different networks —one for the first 12 hours and a second for the remaining 12 hours. We then generate 100 trajectories in which we run the GNN and the ground truth models for 10 time steps per network, i.e. per 12 hours, consequently setting a difference between the network's and the contagion dynamics' time scales. It is important to note that we could have chosen arbitrary time scales and window sizes, as setting these time scales differently would change the evolution of the projections. We chose this setting to obtain clear trajectories with potentially small and large variations.

III. LOCAL TRANSITION PROBABILITIES OF THE INTERACTING CONTAGION DYNAMICS

For the interacting contagion dynamics, the support of the state of nodes is $\Omega = \{S_1S_2, I_1S_2, S_1I_2, I_1I_2\}$. The LTPs therefore consist of 16 probabilities—one for each possible transition—, minus the redundant

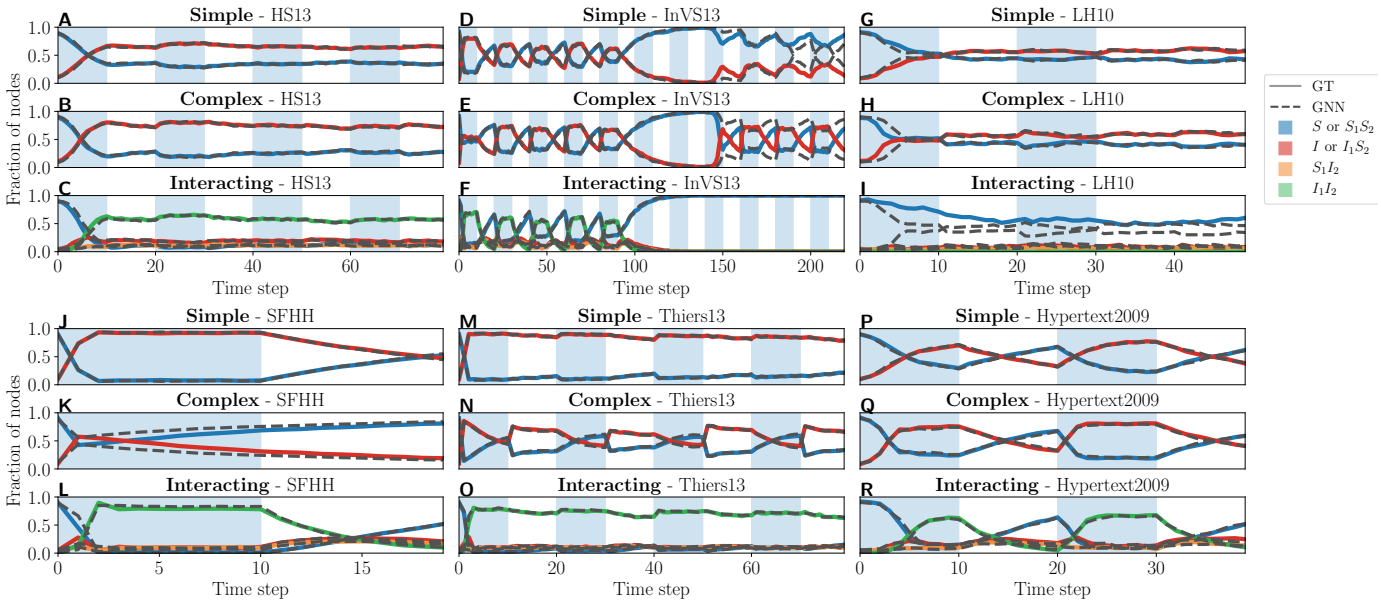

FIG. 3. **Projection of (a) the simple, (b) the complex and (c) the interacting contagion dynamics on real-world temporal networks.** Panels are organized in triple —one panel per contagion dynamics— corresponding to the same dataset. For instance, panels (a–c) correspond to projections on the HS13 temporal network. Dashed lines correspond to projections when using the LTPs predicted by the GNN model (same as in Fig. 2), while the projections of solid lines used the LTPs of the original dynamics (labeled GT for "ground truth"). Similarly to Fig. 2, colors indicate the state of the fraction of nodes. Changes in shaded area indicate where the network transitioned to its next configuration.

ones obtained from the normalization,

$$\sum_{\nu\in\Omega} p_{\boldsymbol{\sigma}}^{\mu\to\nu} = 1\,, \quad \forall\,\mu\in\Omega\,,\,\forall\,\boldsymbol{\sigma}\in\Omega^k\,,\,\forall\,k\in\mathbb{Z}^+\,. \tag{8}$$

Thus there are 12 probabilities in total. Recall that the mechanisms leading to these probabilities are the following: a node infected by disease g transmit the disease to its neighbors susceptible to g with probability τ_g, and recovers from it with probability γ_g, at each time step. If more than one disease is involved in the transmission (from the receiver or the transmitter), the transmission probability is changed to $\zeta\tau_g$, where ζ encodes the coupling the two diseases. From these mechanisms, we obtain the complete distribution of

LTPs. For the transitions from the completely susceptible state, S_1S_2, the LTPs are

$$p_{\boldsymbol{\ell}}^{S_1S_2\to I_1S_2} = \left[1-(1-\tau_1)^{\ell^{I_1S_2}}(1-\zeta\tau_1)^{\ell^{I_1I_2}}\right](1-\tau_2)^{\ell^{S_1I_2}}(1-\zeta\tau_2)^{\ell^{I_1I_2}}\,, \tag{9a}$$

$$p_{\boldsymbol{\ell}}^{S_1S_2\to S_1I_2} = (1-\tau_1)^{\ell^{I_1S_2}}(1-\zeta\tau_1)^{\ell^{I_1I_2}}\left[1-(1-\tau_2)^{\ell^{S_1I_2}}(1-\zeta\tau_2)^{\ell^{I_1I_2}}\right], \tag{9b}$$

$$p_{\boldsymbol{\ell}}^{S_1S_2\to I_1I_2} = \left[1-(1-\tau_1)^{\ell^{I_1S_2}}(1-\zeta\tau_1)^{\ell^{I_1I_2}}\right]\left[1-(1-\tau_2)^{\ell^{S_1I_2}}(1-\zeta\tau_2)^{\ell^{I_1I_2}}\right], \tag{9c}$$

$$p_{\boldsymbol{\ell}}^{S_1S_2\to S_1S_2} = 1-\left[p_{\boldsymbol{\ell}}^{S_1S_2\to I_1S_2}+p_{\boldsymbol{\ell}}^{S_1S_2\to S_1I_2}+p_{\boldsymbol{\ell}}^{S_1S_2\to I_1I_2}\right]. \tag{9d}$$

For the transitions from the state I_1S_2, i.e. infected by the disease 1, we have

$$p_{\boldsymbol{\ell}}^{I_1S_2\to S_1S_2} = \gamma_1(1-\zeta\tau_2)^{\ell^{S_1I_2}+\ell^{I_1I_2}}\,, \tag{10a}$$

$$p_{\boldsymbol{\ell}}^{I_1S_2\to S_1I_2} = \gamma_1\left[1-(1-\zeta\tau_2)^{\ell^{S_1I_2}+\ell^{I_1I_2}}\right], \tag{10b}$$

$$p_{\boldsymbol{\ell}}^{I_1S_2\to I_1I_2} = (1-\gamma_1)\left[1-(1-\zeta\tau_2)^{\ell^{S_1I_2}+\ell^{I_1I_2}}\right], \tag{10c}$$

$$p_{\boldsymbol{\ell}}^{I_1S_2\to I_1S_2} = 1-\left[p_{\boldsymbol{\ell}}^{I_1S_2\to S_1S_2}+p_{\boldsymbol{\ell}}^{I_1S_2\to S_1I_2}+p_{\boldsymbol{\ell}}^{I_1S_2\to I_1I_2}\right], \tag{10d}$$

and similarly, for the transitions from the state S_1I_2, i.e. infected by the disease 2,

$$p_{\boldsymbol{\ell}}^{S_1I_2\to S_1S_2} = (1-\zeta\tau_1)^{\ell^{I_1S_1}+\ell^{I_1I_2}}\gamma_2\,, \tag{11a}$$

$$p_{\boldsymbol{\ell}}^{S_1I_2\to I_1S_2} = \left[1-(1-\zeta\tau_1)^{\ell^{I_1S_1}+\ell^{I_1I_2}}\right]\gamma_2\,, \tag{11b}$$

$$p_{\boldsymbol{\ell}}^{S_1I_2\to I_1I_2} = \left[1-(1-\zeta\tau_1)^{\ell^{I_1S_1}+\ell^{I_1I_2}}\right](1-\gamma_2)\,, \tag{11c}$$

$$p_{\boldsymbol{\ell}}^{S_1I_2\to S_1I_2} = 1-\left[p_{\boldsymbol{\ell}}^{S_1I_2\to S_1S_2}+p_{\boldsymbol{\ell}}^{S_1I_2\to I_1S_2}+p_{\boldsymbol{\ell}}^{S_1I_2\to I_1I_2}\right]. \tag{11d}$$

Finally, the LTPs corresponding to the transitions from the state I_1I_2, i.e. infected by both diseases are

$$p_{\boldsymbol{\ell}}^{I_1I_2\to S_1S_2} = \gamma_1\gamma_2\,, \tag{12a}$$

$$p_{\boldsymbol{\ell}}^{I_1I_2\to I_1S_2} = (1-\gamma_1)\gamma_2\,, \tag{12b}$$

$$p_{\boldsymbol{\ell}}^{I_1I_2\to S_1I_2} = \gamma_1(1-\gamma_2)\,, \tag{12c}$$

$$p_{\boldsymbol{\ell}}^{I_1S_2\to I_1SI_2} = 1-\left[p_{\boldsymbol{\ell}}^{I_1I_2\to S_1S_2}+p_{\boldsymbol{\ell}}^{I_1I_2\to I_1S_2}+p_{\boldsymbol{\ell}}^{I_1I_2\to S_1I_2}\right]. \tag{12d}$$

IV. LOSS OPTIMIZATION PATTERNS

The machine learning problem we address is similar to a classification problem: For a given input, the model learns to assign it the correct label, i.e. the discrete state to which the node transition to. Contrary to standard classification problems, for a given input $(\mu, \boldsymbol{\sigma})$, the label ν is not assigned deterministically, but

stochastically with LTP $p_{\boldsymbol{\sigma}}^{\mu\to\nu}$, which from the classification point of view can be seen as a label distribution. This difference changes how the loss decreases during the learning phase. For example, when we consider a cross entropy objective function and deterministic labels, we expect the loss to descend gradually to zero because the entries of the label distributions are either zeros or ones, i.e. $p_{\boldsymbol{\sigma}}^{\mu\to\nu} \in \{0,1\}$. For stochastic labels, the model achieves maximal accuracy when the cross entropy is equal to the entropy of the label distribution, i.e. $H\left(p_{\boldsymbol{\ell}}^{\mu\to\nu}\right)$. This is obtained by pointing out that minimizing the cross entropy with respect to the parameters $\boldsymbol{\Theta}$ is completely equivalently to minimizing the Kullback-Liebler (KL) divergence between the ground truth $p_{\boldsymbol{\sigma}}^{\mu\to\nu}$ and the model $\hat{p}_{\boldsymbol{\sigma}}^{\mu\to\nu} = \hat{p}_{\boldsymbol{\sigma}}^{\mu\to\nu}(\boldsymbol{\Theta})$ defined as follows:

$$\mathcal{D}\left(p_{\boldsymbol{\sigma}}^{\mu\to\nu}\middle|\middle|\hat{p}_{\boldsymbol{\sigma}}^{\mu\to\nu}\right) = -\sum_{\nu\in\Omega} p_{\boldsymbol{\sigma}}^{\mu\to\nu}\log\hat{p}_{\boldsymbol{\sigma}}^{\mu\to\nu} - H\left(p_{\boldsymbol{\sigma}}^{\mu\to\nu}\right) . \tag{13}$$

Because the KL divergence is non-negative, it follows that its minimum value, corresponding to maximal accuracy, is exactly zero, hence

$$\min_{\boldsymbol{\Theta}}\left[-\sum_{\nu\in\Omega} p_{\boldsymbol{\sigma}}^{\mu\to\nu}\log\hat{p}_{\boldsymbol{\sigma}}^{\mu\to\nu}\right] = H\left(p_{\boldsymbol{\sigma}}^{\mu\to\nu}\right) . \tag{14}$$

This motivates the monitoring of the entropy of the GNN predicted LTPs alongside the loss function during training. Indeed, because we expect $\hat{p}_{\boldsymbol{\sigma}}^{\mu\to\nu} \approx p_{\boldsymbol{\sigma}}^{\mu\to\nu}$ after a while, the entropy of the GNN predicted LTPs should be close to the true value of $H(p_{\boldsymbol{\sigma}}^{\mu\to\nu})$, which in turn should remain of the same magnitude as the loss function if the training is going smoothly.

In Fig. 4, we show the evolution of different metrics throughout the training, namely the loss function, the GNN prediction entropy and the Jensen-Shannon distance (JSD) between the GNN predicted LTPs and the ones given by the maximum likelihood estimators (MLE). As expected, we see the loss decreasing as the training goes on, but does not descend to zero because of the stochasticity of the labels. In fact, it remains of the same order of magnitude as the GNN prediction entropy, as expected. However, the JSD clearly confirms that the model gets closer to the MLE, regardless of the fact that the loss is far from zero. Note that the JSD for the interacting contagion remains larger than the other contagion models because the MLE quality is quite poor in this case.

V. DATASET GENERATION

A. Algorithm

We generate data from each dynamics using the following algorithm:

1. Sample a graph G from a given generative model (e.g. the Erdős-Rényi or the Barabási-Albert models).

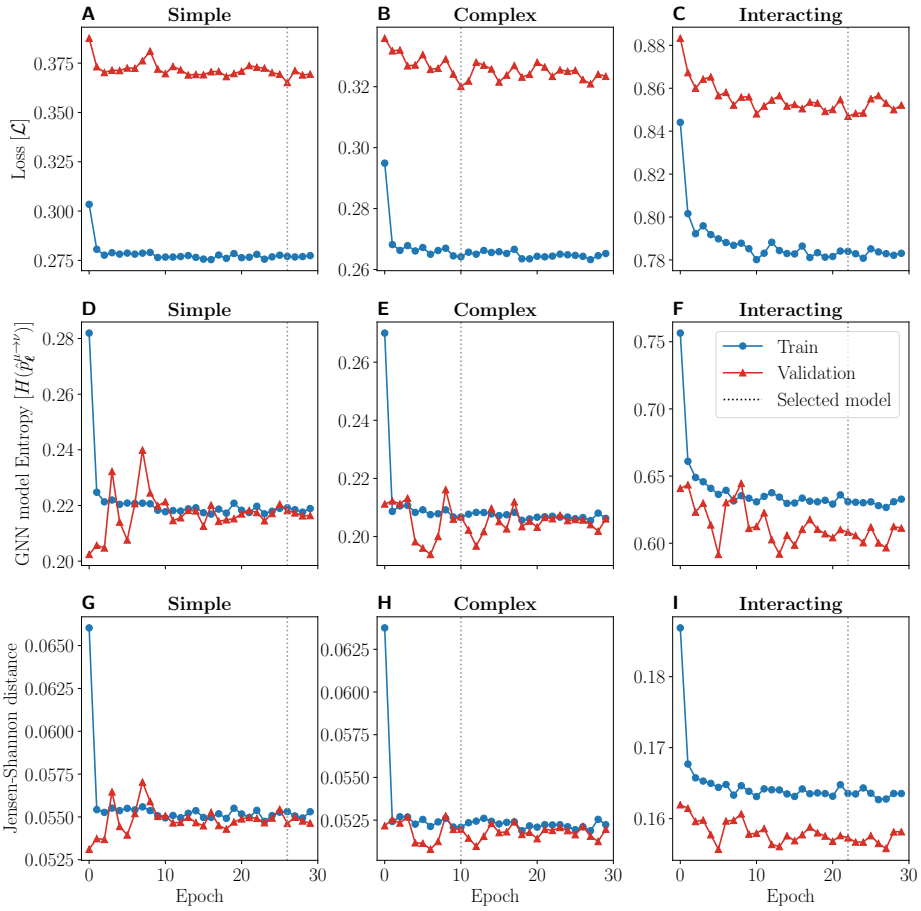

FIG. 4. **Loss optimization patterns during training.** (a–c) Loss as expressed by Eq. (3) in the main text, (d–f) average entropy of the GNN model predictions, (g–i) average Jensen-Shannon distance (JSD) between the GNN predicted LTPs and the ones given by the MLE. We show the results obtained when using Barabási-Albert networks to generate the data; similar conclusions are obtained when using data generated with Erdős-Rényi networks. All measures shown by these plots are approximated using the importance sampling strategies used to compute the loss. The vertical dotted lines show the minimum value of the validation loss, corresponding to the criterion for our model selection.

2. Initialize the state of the system $X(0) = \big(x_i(0)\big)_{i=1..N}$. That is, for disease g, sample n_g uniformly between 0 and N and select uniformly and without replacement n_g nodes from the network. Then, assign a state to each node according to this selection: For instance, if node i is selected to have disease 1 and disease 2, it is assigned the state $I_1 I_2$.

3. At time t, sample $X(t + \Delta t)$ with the LTP conditioned on $X(t)$, where $\Delta t = 1$ without loss of generality. We record the states $X(t)$ and $X(t + \Delta t)$ for training as inputs and targets, respectively.

4. Repeat step 3 until $(t \mod t_s) = 0$, where t_s denotes a resampling time. At this moment, apply

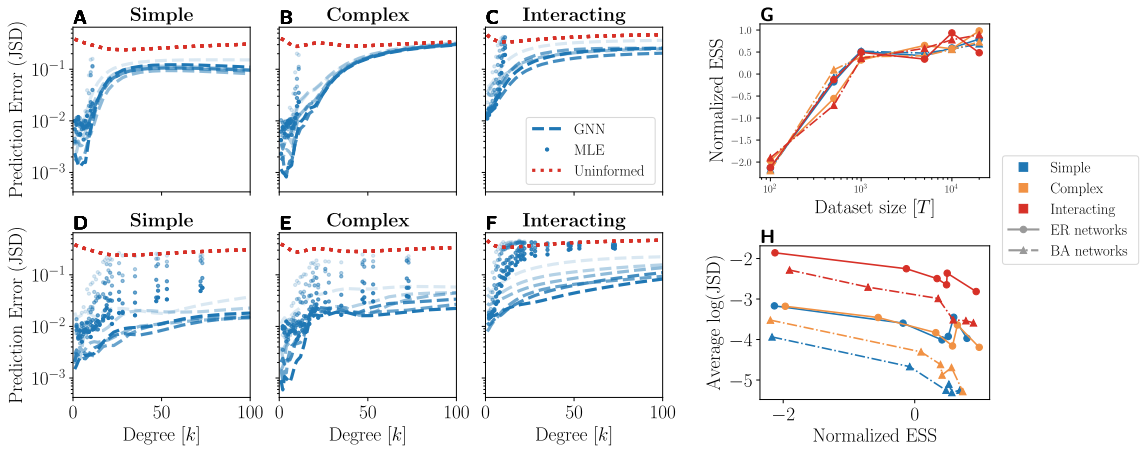

FIG. 5. (color online) **Impact of the dataset size on the prediction error.** (a–c) Models trained on ER networks, (d–f) models trained on BA networks, (g) normalized effective sample size (ESS), (h) average logarithm of the JSD error against the ESS. The specific dynamics are indicated on top of each column. We show the prediction error—calculated from the Jensen Shannon distance (JSD)— for different dataset size $T \in \{100, 500, 1000, 5000, 10000, 20000\}$. Solid lines correspond to the error of the GNN predictions, dotted lines denote the error of the uninformed baseline and symbols, the error of the MLE computed from the training dataset. On Figs. (a–f), the shade of blue indicates the value of T: darker blue means larger T. On Fig. (g–h), the colors indicate the type of dynamics used for training, and the lines and symbols, the type of networks. All hyperparameters are listed in Sec. IV.C of the Supplementary Material.

step 2 to reinitialize the states $X(t)$ and repeat step 3.

5. Stop when $t = T$, where T is the targeted number of samples.

The resampling step parametrized by t_s indirectly controls the diversity of the training dataset. We allow t_s to be small to emphasize on the performance of the GNN rather than the quality of the training dataset, while acknowledging that large values of t_s could lead to poor training. Because of the very nature of contagion dynamics, which contains absorbing and endemic steady states, it is expected that letting the dynamics evolve on its own could lead to datasets with high redundancy composed largely of degenerate states. This artificial mechanism has the advantage of increasing the effective sample size of the dataset by reducing this naturally occurring redundancy.

B. Impact of the Data Generation Hyperparameters

We investigate the impact of several hyperparameters involved in the data generation on the performance of the GNN, namely the dataset size T, the resampling time t_s and the number of nodes N. In general, we

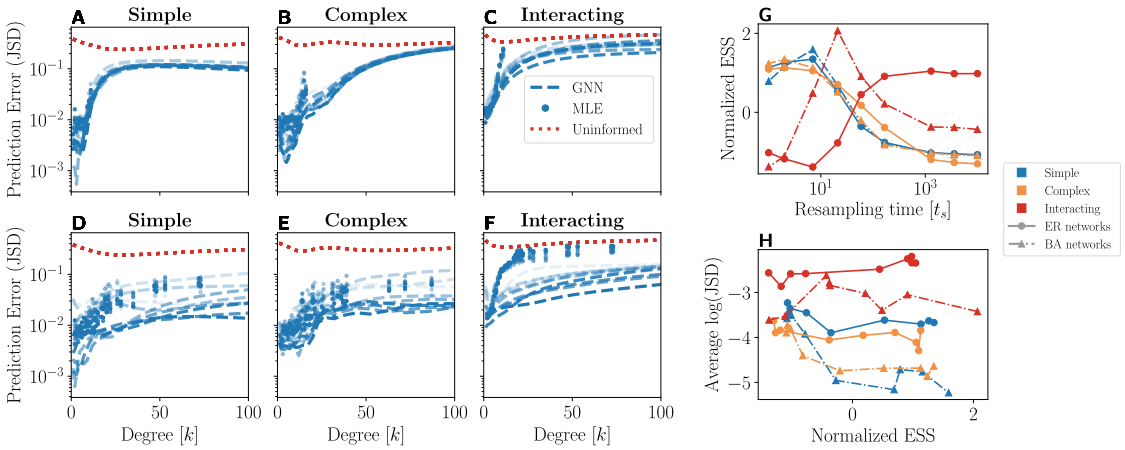

FIG. 6. (color online) **Impact of the resampling time on the prediction error.** (a–c) Models trained on ER networks, (d–f) models trained on BA networks, (g) normalized effective sample size (ESS), (h) average logarithm of the JSD error against the ESS. In these experiments, we fixed the resampling times to $t_s \in \{1, 2, 7, 21, 59, 166, 464, 1291, 3593, 10000\}$. For both the GNN and MLE errors, the shade of blue indicates the value of t_s: darker blue means smaller t_s. See the caption of Fig. 5 of the Supplementary Material for further details.

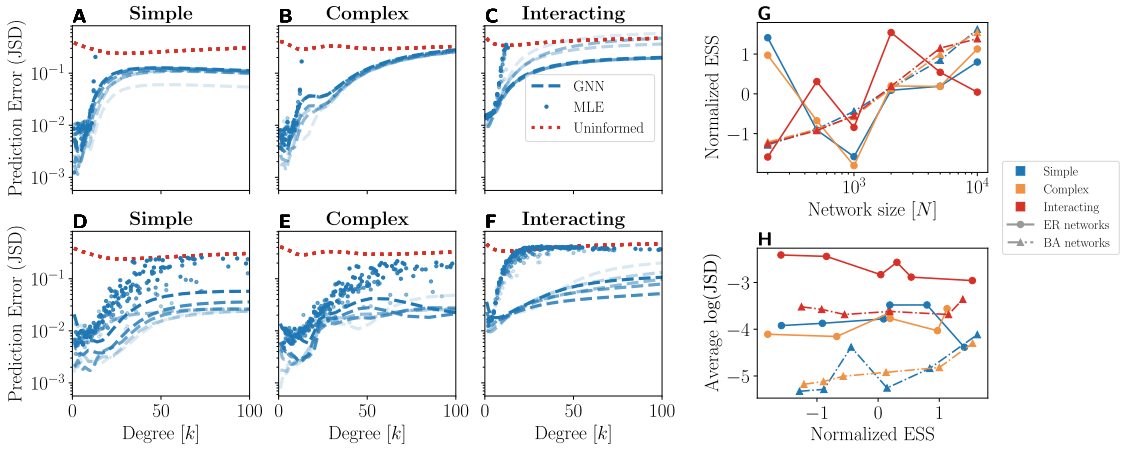

FIG. 7. (color online) **Impact of the number of nodes of training networks on the average error.** (a–c) Models trained on ER networks, (d–f) models trained on BA networks, (g) normalized effective sample size (ESS), (h) average logarithm of the JSD error against the ESS. In these experiments, we fixed the network sizes to $N \in \{200, 500, 1000, 5000, 10000\}$. In order to better appreciate the impact of increasing N, we fixed $T = 10^7/N$ to maintain a comparable number of samples, and fixed $\langle k \rangle = 4$ in all experiments. For both the GNN and MLE errors, the shade of blue indicates the value of N: darker blue means larger N. See the caption of Fig. 5 of the Supplementary Material for further details.

expect the performance to increase with the variety of the dataset. To quantify this variety, we use the effective sample size (ESS), given by

$$n_{\text{eff}} = \frac{\left(\sum_{\mu,\boldsymbol{\ell}} n_{\boldsymbol{\ell}}^{\mu}\right)^2}{\sum_{\mu,\boldsymbol{\ell}} \left(n_{\boldsymbol{\ell}}^{\mu}\right)^2} \tag{15}$$

where

$$n_{\boldsymbol{\ell}}^{\mu} = \sum_{t=1}^{T} \sum_{i \in \mathcal{V}} I\left[x_i(t) = \mu\right] I\left[\boldsymbol{\ell}_i(t) = \boldsymbol{\ell}\right] , \quad \boldsymbol{\ell}_i(t) = \left(\ell_i^{\mu}(t)\right)_{\mu \in \Omega} = \left(\sum_{j \in \mathcal{N}_i} I\left[x_j(t) = \mu\right] \right)_{\mu \in \Omega} . \tag{16}$$

where $I[c]$ is an indicator function where $I[c] = 1$ if c is true, and $I[c] = 0$ otherwise. We expect that larger ESS will lead to a better approximated loss, in turn yielding better trained models. To compare all experiments, we consider the normalized ESS, denoted by

$$\frac{n_{\text{eff}} - \mathrm{E}(n_{\text{eff}})}{\sqrt{\mathrm{var}(n_{\text{eff}})}} \tag{17}$$

where the expectation and the variance are taken over the values of hyperparmeters.

On Figs. 5-6-7, we show the prediction error, measured by the JSD, on star graphs while changing the different data generation hyperparmeters (T, t_s and N, respectively). As we can see from Figs. 5-6, increasing T and reducing t_s tend to help the models achieve higher performance, with diminished prediction errors. In most cases, this is well correlated with the ESS which tend to be large when the error is small, while the effect is more subtle for the resampling time than for the dataset size. It is interesting to note that small values of t_s are not always ideal, as we observe increase in ESS for the interacting contagion dynamics when t_s is larger.

Surprisingly, increasing the network size N does not seem to increase the performance of our models. First, for ER networks, increasing N does not tend to increase the ESS. This is expected because the maximum degree only slighting increase when the number of nodes is increased, for fixed the average degree $\langle k \rangle$. Hence, we do not observe additional degree classes when N is marginally increased and the training dataset variety remains similar. For BA networks, we observe something different: While the increase in N leads to higher ESS, there is still no substantial gain in performance. This can be explained by looking at the degree distribution. As more nodes are added to the network, the degree classes get more populated, resulting in increased ESS. However, because the degree distribution is scale-free (with exponent -3), these are not populated evenly and more degree classes are created as N increases. This has the effect of raising the problem difficulty, because increasing the number of accessible degree classes also increases the number of cases—LTPs with specific inputs—the GNN model needs to fit.

VI. TRAINING SETTINGS

A. Optimization

We use the Rectified Adam optimization algorithm presented in Ref. [6]. Similarly to Ref. [7], this algorithm minimizes an objective function by estimating the average and variance of its gradient from moving averages. These moving averages are parametrized by $b_1, b_2 \in [0, 1)$ for the average and the variance respectively, which we specify below.

B. Validation Dataset Selection

Validating the model is a crucial step of the learning pipeline where the performance of the model is objectively evaluated. In most cases, it is done by selecting a subset of the training dataset, called the validation dataset, from which the model will not learn from. The performance is then evaluated by computing the average loss with respect to this validation dataset, which allows us to monitor the learning process.

The selection of the validation set is usually done by sampling randomly a small percentage, about 20%, of samples in the training dataset. In general, this is not an issue in high dimensional problems where each data point is virtually unique. However, in our case, the data points correspond to each individual input $(\mu, \boldsymbol{\sigma})$, which is likely to be repeated multiple times. Therefore, it is not recommended to proceed by splitting the training dataset at random. In fact, this could jeopardize the whole validation procedure if the training and validation datasets distribution are too similar.

Instead, we propose to sample the transitions by importance similarly to the importance sampling scheme approximating the loss function presented in the main text. Let us consider $\mathcal{W}(t) \subset \mathcal{V}$, a subset of nodes at time t selected for validation. The probability that the transition of node i at time t belongs to the validation dataset is equal to

$$\Pr\left[i \in \mathcal{W}(t)\right] = \frac{\rho\big(x_i(t), \boldsymbol{\ell}_i(t)\big)^{-\delta}}{\sum_{i \in \mathcal{V}} \rho\big(x_i(t), \boldsymbol{\ell}_i(t)\big)^{-\delta}} \tag{18}$$

where $\rho(\mu, \boldsymbol{\ell})$ denotes the input distribution, and $\delta \geq 1$ is a parameter controlling the bias towards rare inputs. During training, the loss function is then approximated by

$$\mathcal{L}(\boldsymbol{\Theta}) \simeq \sum_{t \in \mathcal{T}'} \sum_{i \in \mathcal{V}'(t)} \frac{\omega_i(t)}{|\mathcal{T}'|\,|\mathcal{V}'(t)|} \left[-\log \hat{p}^{x_i(t) \to x_i(t+\Delta t)}_{x_{\mathcal{N}_i}(t)} \right] \tag{19}$$

where $\mathcal{V}'(t) = \mathcal{V} \backslash \mathcal{W}(t)$. Sampling the validation dataset by importance allows us to validate the model on all available inputs with equal weights, i.e. when $\delta \to 1$. It also helps to minimize the similarity between the training and the validation datasets, as the rarest inputs will be likely only be available in the latter.

C. Hyperparameters

For all experiments, we fix the optimizer parameters to $b_1 = 0.9$ and $b_2 = 0.999$ as was suggested in Ref. [7] and we schedule the learning rate ϵ to reduce by a factor 2 every 10 epochs, that is $\epsilon_{i+1} = 2^{\lfloor \frac{i \mod 10}{10} \rfloor} \epsilon_i$ with initial value $\epsilon_0 = 0.0005$. A weight decay of 10^{-4} is used as well to help regularize the training. We set the number of epochs to 20 and choose the model with the lowest loss on validation datasets as our best model. We fix the importance sampling bias exponents for the training and the validation to $\lambda = 0.6$ and $\delta = 0.8$, respectively. For the data generation, we fix the sequence size $t_s = 2$ and we used $T = 10000$ samples for the experiments presented in the main paper.

VII. MEAN FIELD FRAMEWORK

A. Approximate master equations

The mean field framework we use is inspired by [8] and provides an approximate solution to the stationary distribution of contagion dynamics using a set of discrete-time approximate master equations (AME). To construct the AME, we consider a set of LTPs $p_{\boldsymbol{\ell}}^{\mu\to\nu}$ and a state matrix $\boldsymbol{Q}(t)$, where the entry $[\boldsymbol{Q}(t)]_{\mu,k}$ corresponds to the probability that a node of degree k is in state μ at time t. Then, the AME that describes the evolution of $\boldsymbol{Q}(t)$ is expressed as follows:

$$[\boldsymbol{Q}(t+\Delta t)]_{\mu,k} = \sum_{\nu} [\boldsymbol{Q}(t)]_{\nu,k} \sum_{|\boldsymbol{\ell}|=k} M_k(\boldsymbol{\ell}; t) p_{\boldsymbol{\ell}}^{\mu\to\nu} \tag{20}$$

where $M_k(\boldsymbol{\ell})$ is the probability that the neighborhood of a node of degree k has state $\boldsymbol{\ell}$ at time t, recalling that $[\boldsymbol{\ell}]_\mu = \ell^\mu$ corresponds to the number of neighbors in state μ. The probability $M_k(\boldsymbol{\ell})$ is approximated by a multinomial distribution,

$$M_k(\boldsymbol{\ell}; t) = k! \prod_{\nu} \frac{\phi^\nu(t)^{\ell^\nu}}{\ell^\nu!} \tag{21}$$

where $\phi^\nu(t)$ is the probability that a node at the end of a randomly selected edge is in state ν at time t. The underlying assumption behind this parameterization of $M_k(\boldsymbol{\ell})$ is that the neighbors of a given node are assumed dynamically and structurally uncorrelated. These assumptions lead us to the following closed form for $\phi^\nu(t)$:

$$\phi^\nu(t) = \frac{\sum_k k \rho_k [\boldsymbol{Q}(t)]_{\nu,k}}{\langle k \rangle} \tag{22}$$

where $\rho_k = \Pr(K = k)$ is the probability that a node has degree k and $\frac{k\rho_k}{\langle k \rangle}$ is the probability to reach a node of degree k from a randomly selected edge.

We use this mean field framework to compute the stationary distributions of the three contagion dynamics we investigated in the main text, as well as the stationary distributions predicted by the trained GNN models. To simplify our analysis, we consider that $\rho_k = e^{-\kappa}\kappa^k/k!$ is a Poisson distribution with parameter and κ with a truncated support $\mathcal{K} = [k_{\min}, k_{\max}] \subset \mathbb{Z}^+$. In our experiments, we consider that the average degree $\langle k \rangle$ is our control parameter, rather than κ. Therefore, to ensure that $\langle k \rangle$ remains fixed, we fix the support using

$$k_{\min} = \max\left\{0, \lfloor \langle k \rangle \rfloor - 7\right\}, \quad k_{\max} = \max\left\{14, \lceil \langle k \rangle \rceil + 7\right\} \tag{23}$$

where the operations $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ are the floor and ceil functions, respectively. This guarantees that $|\mathcal{K}| = k_{\max} - k_{\min} + 1 = 15$. Then, we fix κ by solving numerically the following equation:

$$\sum_{k \in \mathcal{K}} k\rho_k = \sum_{k \in \mathcal{K}} k \frac{e^{-\kappa}\kappa^k}{k!} = \langle k \rangle \ . \tag{24}$$

To apply the AME framework to a GNN model, we begin by extracting its LTPs beforehand using the prescription described in the main text involving the star graph of $k+1$ nodes, whose central node can generate any desired input $(\mu, \boldsymbol{\ell})$ given its degree k. Then, Eq. (20) is solved numerically using a relaxation method [9].

It is possible to obtain a more refined version of Eq. (20), as prescribed in Ref. [8]. However, even in this simple case, iterating Eq. (20) becomes rapidly tedious as the number of available states increases. Specifically, to update Eq. (20), one needs to enumerate

$$|\Omega| \binom{k + |\Omega| - 1}{|\Omega| - 1} \tag{25}$$

terms, which scales poorly with $|\Omega|$ and k. The scaling is even worst for more refined frameworks.

B. Numerical Evaluation of the Thresholds

Contagion dynamics are known to have phase transitions which delineate different dynamical behaviors. More specifically, beyond some threshold values of the dynamical and structural parameters, contagion dynamics shift abruptly from an absorbing phase, where all nodes are susceptible in the steady state—noted $\boldsymbol{Q}^*$ where $[\boldsymbol{Q}^*]_{\mu,k} = I[\mu = S]$—, to an endemic phase in which a nonzero fraction of nodes remains infected over time [10–12]. We note the endemic state $\boldsymbol{Q}^\dagger$, where $[\boldsymbol{Q}^\dagger]_{\mu,k} > 0$ is obtained numerically from Eq. (20) using a relaxation method with the initial condition $[\boldsymbol{Q}(0)]_{\mu,k} = (1-\epsilon)I[\mu \neq S] + \epsilon I[\mu = S]$ assuming $\epsilon \ll 1$ [13]. In the case of the susceptible-infected-susceptible dynamics (SIS), the phase transition occurs at the point where the infection and recovery probabilities, τ and γ, are related to the first

and second moments of the degree distribution as follows:

$$\frac{\tau}{\gamma} = \frac{\langle k \rangle}{\langle k^2 \rangle} \,. \tag{26}$$

One obtains this relation by computing the stability $\lambda(\boldsymbol{Q}^*)$ of the absorbing state $\boldsymbol{Q}^*$, which corresponds to the largest eigenvalue of the Jacobian matrix $\mathbf{J}(\boldsymbol{Q}^*)$ evaluated at that point. The entries of the Jacobian matrix, indexed by the pairs (μ, k) and (μ', k'), evaluated at an arbitrary point $\boldsymbol{Q}$ are computed as follows:

$$[\mathbf{J}(\boldsymbol{Q})]_{(\mu,k),(\mu',k')} = \left. \frac{\partial [\boldsymbol{Q}(t+\Delta t)]_{\mu,k}}{\partial [\boldsymbol{Q}(t)]_{\mu',k'}} \right|_{\boldsymbol{Q}(t)=\boldsymbol{Q}} . \tag{27}$$

A fixed point $\boldsymbol{Q}$ is stable when $\lambda(\boldsymbol{Q}) < 1$, and unstable otherwise. As we consider that the parameters of contagion dynamics to remain unchanged, we compute the thresholds of the phase transitions with respect to the average degree $\langle k \rangle$. For simple contagion dynamics, the threshold is obtained numerically by solving for the value of $\langle k \rangle$ for which the stability of the absorbing state yields $\lambda(\boldsymbol{Q}^*) = 1$. For complex and interacting dynamics, we must find two thresholds that delineate a bistable regime where both the absorbing state $\boldsymbol{Q}^*$ and the endemic state $\boldsymbol{Q}^\dagger$ are stable. Therefore, we apply the same strategy and solve numerically for $\langle k \rangle$ the two equations $\lambda(\boldsymbol{Q}^*) = 1$ and $\lambda(\boldsymbol{Q}^\dagger) = 1$.

[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations* (2018).

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation Learning on Graphs: Methods and Applications," (2017), arXiv:1709.05584.

[3] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck, "What's in a crowd? Analysis of face-to-face behavioral networks," J. Theor. Biol. **271**, 166–180 (2011).

[4] M. Génois and A. Barrat, "Can co-location be used as a proxy for face-to-face contacts?" EPJ Data Sci. **7**, 11 (2018).

[5] R. Mastrandrea, J. Fournet, and A. Barrat, "Contact Patterns in a High School: A Comparison between Data Collected Using Wearable Sensors, Contact Diaries and Friendship Surveys," PLOS ONE **10**, e0136497 (2015).

[6] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the Variance of the Adaptive Learning Rate and Beyond," arXiv , 1908.03265 (2020).

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," (2014), arXiv:1412.6980.

[8] P. G. Fennell and J. P. Gleeson, "Multistate Dynamical Processes on Networks: Analysis through Degree-Based Approximation Frameworks," SIAM Rev. **61**, 92–118 (2019).

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. (Cambridge University Press, USA, 2007).

[10] I. Z. Kiss, J. C. Miller, and P. L. Simon, *Mathematics of Epidemics on Networks* (Springer, 2017) p. 598.

[11] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," Rev. Mod. Phys. **87**, 925–979 (2015).

[12] J. Sanz, C.-Y Xia, S. Meloni, and Y. Moreno, "Dynamics of Interacting Diseases," Phys. Rev. X **4**, 041005 (2014).

[13] Equivalently, in the case of the interacting contagion dynamics, we use $[\boldsymbol{Q}(0)]_{\mu,k} = (1 - \frac{\epsilon}{4})I[\mu \neq S_1 S_2] + \epsilon I[\mu = S_1 S_2]$ as an initial condition.