Transfer Learning for Node Regression Applied to Spreading Prediction

Sebastian Mežnar
Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia

Nada Lavrač
Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia
and
University of Nova Gorica, Glavni trg 8, Vipava, Slovenia

Blaž Škrlj
Jožef Stefan Institute
and
Jožef Stefan International Postgraduate School
Jamova 39, Ljubljana, Slovenia

Understanding how information propagates in real-life complex networks yields a better understanding of dynamic processes such as misinformation or epidemic spreading. The recently introduced branch of machine learning methods for learning node representations offers many novel applications, one of them being the task of spreading prediction addressed in this paper. We explore the utility of the state-of-the-art node representation learners when used to assess the effects of spreading from a given node, estimated via extensive simulations. Further, as many real-life networks are topologically similar, we systematically investigate whether the learned models generalize to previously unseen networks, showing that in some cases very good model transfer can be obtained. This paper is one of the first to explore transferability of the learned representations for the task of node regression; we show there exist pairs of networks with similar structure between which the trained models can be transferred (zero-shot) and demonstrate their competitive performance. To our knowledge, this is one of the first attempts to evaluate the utility of zero-shot transfer for the task of node regression.

Keywords: epidemics; neural networks; machine learning; spreading; transfer learning

1. Introduction

Spreading of information or disease spreading are examples of common phenomena of spreading. Modeling the spreading process and spreading prediction has many practical and potentially life-saving applications, including the creation of better strategies for stopping

the spreading of misinformation on social media or stopping an epidemic. Further, companies can analyze spreading to create better strategies for marketing their product [1, 2]. Spreading analysis can also be suitable for analysis of, for example, fire spreading, implying large practical value in terms of insurance cost analysis [3]. Analysis of spreading is commonly studied via extensive simulations [4], exploiting ideas from statistical mechanics to better understand both the extent of spreading and its speed [5].

While offering high utility, reliable simulations of spreading processes can be expensive when performed on larger networks. This issue can be addressed by employing machine learning–based modeling techniques [6]. The contributions of this paper are multifold and can be summarized as follows.

1. We propose an efficient network node regression algorithm named CaBoost, which achieves state-of-the-art performance for the task of spreading prediction against strong baselines such as graph neural networks.

2. We demonstrate that machine learning–based spreading prediction can be utilized for fast screening of potentially problematic nodes, indicating that this branch of methods is complementary to the widely adopted simulation-based approaches.

3. We investigate to what extent the models learned on a given network A are transferable to a network B, and what type of structural features preserves this property the best. This hypothesis assumes that structure-only properties could be sufficient for model transfer in some cases. We demonstrate that transfer learning for node regression is possible, albeit only across topologically similar networks.

This work extends the paper "Prediction of the Effects of Epidemic Spreading with Graph Neural Networks" [7] from the Complex Networks 2020 conference by testing more approaches, using more simulation data, and using different node centralities. Additionally, this work tests if centrality-based features can be used for zero-shot transfer learning.

The remainder of this paper is structured as follows. Section 2 presents the related work that led to the proposed approach. In Section 3 we present the proposed methodology, where we reformulate the task, present centrality data used to create features for our learners and show how we approached transfer learning. In Section 4 we present the datasets, the experimental setting, interpretation of the predictions, and the results of the empirical evaluation and transfer learning. We conclude the paper with the discussion in Section 5.

2. Related Work

This section presents the relevant related work. It starts by discussing the notion of contagion processes, followed by an overview of graph neural networks and transfer learning.

2.1 Analysis of Spreading Processes

The study of spreading processes on networks is a lively research endeavor [2]. Broadly, spreading processes can be split into two main branches, namely, the simulation of epidemics and opinion dynamics. The epidemic spreading models can be classical or network based. The classical models are, for example, systems of differential equations that do not account for a given network's topology. Throughout this paper, we are interested in extensions of such models to real-life network settings.

One of the most popular spreading models extended to networks is the susceptible infected recovered (SIR) model [8]. The spread of the pandemic in the SIR model is dictated by parameters β, known as the infection rate, and γ, known as the recovery rate. Nodes in this model can have one of three states (susceptible, infected, recovered). SIR assumes that if an infected node comes into contact with a susceptible one during a generic iteration, the susceptible node becomes infected with probability β. In each iteration after getting infected, a node can recover with probability γ (only transitions from S to I and from I to R are allowed).

Other related models include, for example, SEIR, SEIS, SWIR. (Where S-Susceptible, I-Infected, R-Recovered, E-Exposed and W-Weakened.) Further, we can also study the role of cascades [9] or the threshold model [10]. For the interested reader, multiple other approaches are summarized in [11].

2.2 Machine Learning on Networks

Learning by propagating information throughout a given network has already been considered by approaches such as label propagation [12]. However, in recent years, approaches that jointly exploit the adjacency structure of a given network alongside features assigned to its nodes are becoming prevalent in the network learning community. The so-called graph neural networks have resurfaced with the introduction of the graph convolutional networks (GCNs) [13], an idea where the normalized adjacency matrix is directly multiplied with the feature space and effectively represents a neural network layer. Multiple such layers can be stacked to obtain better approximation power/performance. One of the most recent methods from this branch is the graph attention networks (GAT) [14], an extension of

GCNs, extended with the idea of neural attention. Here, part of the neural network focuses on particular parts of the adjacency space, offering more robust and potentially better performance.

Albeit being in widespread use, graph neural networks are not necessarily the most suitable choice when learning solely from the network adjacency structure. For such tasks, methods such as node2vec [15], SGE [16], SNoRe [17] and DeepWalk [18] were developed. This branch of methods corresponds to what we refer to as structural representation learning. In our work, we focus mostly on learning this type of representations using network centrality information.

Note that although graph neural networks are becoming the prevailing methodology for learning from feature-rich complex networks, it is not clear whether they perform competitively with the more established structural methods if the feature space is derived solely from a given network's structure.

2.3 Transfer Learning

The main bottleneck of spreading effect prediction is the expensive simulations. While the number of simulations can be reduced by using machine learning, computation of a large fraction of them might still be infeasible on larger networks. One of the solutions for this problem is *transfer learning* [19]. Transfer learning can be performed in at least two main ways: by fine-tuning a pretrained model (few-shot learning) or by using a model trained on a related problem (zero-shot learning). In this paper, we focus on *zero-shot learning*.

In recent years, new approaches were proposed for transfer learning on networks; those adopted mostly fine-tune pretrained graph neural networks to solve the proposed problem. An example of this is prediction of optoelectronic properties of conjugated oligomers [20], where the graph neural network is trained on short oligomers and then fine-tuned by using 100 long ones. The results showed that the fine-tuned model performed much better and needed only a small sample of extra data to improve performance by a margin of 37%. Another approach tries to predict traffic congestion of a network with a small amount of historical data by training a recurrent neural network on a traffic network with a lot of historic data [21]. Zero-shot learning is less popular in the network setting, but the few approaches that exist closely follow the paradigms of zero-shot learning [22]. One such example [23] proposes a dense graph propagation module that adds direct links between distant nodes in the knowledge networks to exploit their hierarchical structure. Finally, pre-training graph neural networks is becoming an active research venue, demonstrating that this problem is possible to solve via systematic selection of pre-training data [24, 25].

When transferring knowledge between different networks, one must carefully craft features that are independent of a single network and represent global node characteristics. Because of this, embedding methods such as node2vec [15] and SNoRe [17] cannot be used, since they learn node representation through node indices encountered during random walks within the input network. On the other hand, transfer learning with graph neural networks is an active field of research, with most approaches fine-tuning the pre-trained models.

3. Proposed Methodology

After introducing the task of spreading prediction, a brief methodology overview is presented. After this we present the creation of target variables, extraction of node features from the network, training of machine learning models, and the transfer of models between the networks.

3.1 Task Formulation

In the case of a pandemic, the intensity of disease spreading can be summarized with the following three values: the maximum number of infected people (the peak), the time it takes to reach the maximum number of infected people and the total number of infected people. Let us consider why these three values are important. Knowing the maximum number of infected people during a pandemic will help us to better prepare for the crisis, as it provides a good estimation of how many resources (e.g., hospital beds) will need to be allocated to patients. In another example domain, companies might want to create marketing campaigns on platforms such as Twitter and target specific users to reach a certain number of retweets that are needed to become trending. The time needed to reach the peak is important, for example, to estimate the best time for developing a cure for the disease, or for example, to estimate the maximum time for stopping the spread of misinformation on social media. Finally, the total number of infected nodes during an epidemic is important for assessing the damage, or in another example, for estimating how many computers were infected by some malware.

In this paper, we focus on predicting the maximum number of infected nodes and the time needed to reach this number. We create target data by simulating epidemics from each node with the SIR diffusion model [26] and identify the number of nodes, as well as the time when the maximum number of nodes is infected. We aggregate the generated target data by taking the mean values for each node (expected time and infection numbers). Finally, we normalize the data

as follows. We divide the maximum number of infected nodes by the number of nodes in the observed network. This normalization is suitable since the maximum number of infected nodes cannot exceed the number of nodes in the network. This normalization intuitively offers insight into the percentage of infected nodes, regardless of scale. The upper bound for the time when the maximum number of nodes is infected does not exist. Because of this, we divide the time when the maximum number of nodes is infected with the maximum from the observed data. In practice we might not have this maximum, so a suitable number must be chosen by, for example, examining prior events. Furthermore, in practice, we create the target data as described above and save the number by which we divide (normalization constant). After prediction, we multiply the prediction with this number to find the "real-world" equivalent.

3.2 Methodology Overview

The overview of the proposed methodology is presented in Figure 1. The methodology is composed of three main steps: input data creation, model training and transfer learning.
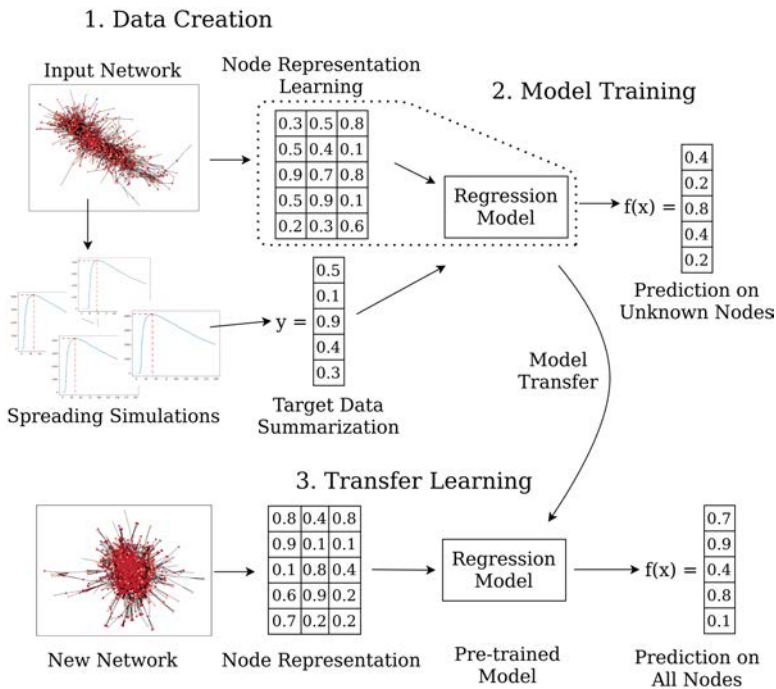

Figure 1. Overview of the proposed methodology.

In the first step, we create target variables and node features from the starting (initial) network. Target variables are created by running simulations on the starting network followed by their summarization and processing. While target variable creation can be performed in the same way on each network, feature creation depends on the learner. Embeddings and feature extraction methods generate features stored in tabular format, which is then used for model training. Graph neural networks are, on the other hand, end-to-end learners, meaning they learn features and perform regression at the same time.

In the second step, we train a regression model using the extracted features and the target variables we created. We use this model to predict the target variables for the unknown nodes.

The model learned in the second step is used in the last step to predict target variables of nodes from a new (unknown/unobserved) network. For such predictions, we first extract features from the new network and then use them on the pre-trained model. These predictions transfer knowledge from the first network to the nodes of the second one.

▌ 3.3 Training Data Creation

The first part of the methodology addresses input data generation. Intuitively, the first step simulates epidemic spreading from individual nodes of a given network to assess the time required to reach the maximum number of infected nodes, as well as the maximum number itself. In this paper, we leverage the widely used SIR model [8, 26] to simulate epidemics, formulated as follows:

$$\frac{dS}{dt} = -\frac{\beta \cdot S \cdot I}{N}$$

$$\frac{dI}{dt} = \frac{\beta \cdot S \cdot I}{N} - \gamma \cdot I$$

$$\frac{R}{dt} = \gamma \cdot I,$$

where S represents the number of susceptible, R the number of recovered and I the number of infected individuals. Spreading is governed by input parameters γ and β. The SIR model is selected due to many existing and optimized implementations that are adapted from systems of differential equations to networks [1]. We use NDlib [26] to simulate epidemics based on the SIR diffusion model.

Target data creation results in two real values for each node. We attempt to predict these two values. The rationale for the construction of such predictive models is that they are potentially much faster than simulating multiple spreading processes for each node (prediction

time is the bottleneck) and can give more insight into why some nodes are more "dangerous."

The predictive task can be formulated as follows. Let $G = (V, E)$ represent the considered network. We are interested in finding the mapping $f : V \to [0, 1]$ from the set of nodes V to the set of real values that represent, for example, the maximum number of infected individuals if the spreading process is started from a given node $u \in V$. Thus, f corresponds to the process of *node regression*.

3.4 Learning on the Same Network: Prediction with Simulation Data

The models we considered can broadly be split into two main categories: graph neural networks and propositional learners. The main difference between the two is that the graph neural network learners, such as GAT [14] and GIN [27], simultaneously exploit the structure of a network, as well as node features, while the propositional learners take as input only the constructed feature space (and not the adjacency matrix). As an example, the feature space is passed through the GIN's structure via the update rule that can be stated as:

$$ h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum_{u \in V(v)} h_u^{(k-1)} \right), $$

where MLP corresponds to a multilayer perceptron, ϵ a hyperparameter, $h_u^{(k)}$ the node u's representations at layer k and $V(v)$ the vth node's neighbors. We test both graph neural networks and propositional learners, as it is to our knowledge not clear whether direct incorporation of the adjacency matrix offers any superior performance, as the graph neural network models are computationally more expensive. The summary of considered learners is presented in Table 1.

Input	Learner	Method Description
A, F	GAT	graph attention networks
A, F	GIN	graph isomorphism networks
A	node2vec + XGBoost	node2vec-based features + XGBoost
A, F	node2vec + features + XGBoost	node2vec-based features + centrality-based features + XGBoost
A	SNoRe + XGBoost	SNoRe-based features + XGBoost
A, F	SNoRe + features + XGBoost	SNoRe-based features + centrality-based features + XGBoost
F	CaBoost	XGBoost trained solely on centrality-based features

Table 1. Summary of the considered learners with descriptions, where *A* denotes the adjacency matrix and *F* the feature matrix.

As the considered complex networks do not possess node attributes, we next discuss which features derived solely from the network structure were used in the considered state-of-the-art implementations of GAT [14] and GIN [27], or concatenated to an embedding generated using node2vec [15] or SNoRe [17] for use in XGBoost. Further, we also test models where only the constructed structural features are considered, as well as a standalone method capable of learning node representations, combined with the XGBoost [28] classifier. In this paper, we explore whether centrality-based descriptions of nodes are suitable for the considered learning task. The rationale for selecting such features is that they are potentially fast to compute and entail global relation of a given node with regard to the remaining part of the network. The centralities, computed for each node, are summarized in Table 2. These centralities are then normalized and concatenated to create features used with some learners. In Section 4.3 we refer to the XGBoost model trained with these features as CaBoost, which is one of the contributions of this paper.

Centrality	Time Complexity	Description
degree centrality [29]	$O\left(|E|\right)$	The number of edges of a given node.
eigenvector centrality [29]	$O\left(|V|^3\right)$	Importance of the node, where nodes are more important if they are connected to other important nodes. This can be calculated using the eigenvectors of the adjacency matrix.
PageRank [30]	$O\left(|E|\right)$	Probability that a random walker is at a given node.
average out-degree	$O\left(|V| \cdot w \cdot \bar{s}\right)$	The average out-degree of nodes encountered during w random walks of mean length $\bar{s}$.
number of second neighbors	$O\left(|V| \cdot |E|\right)$	Number of nodes that are neighbors to neighbors of a given node. This number is between 0 and $|V|$.

Table 2. Summary of the centralities considered in our work.

During model training, we minimize the mean squared error (MSE) between the prediction $f(u)$ and the observed state y_u, which is defined for the uth node as follows:

$$\text{MSE} = \frac{1}{|N|} \sum_{u \in N} \left(f(u) - y_u\right)^2.$$

In Section 4.3 we use the root mean squared errors (RMSE), defined as follows:

$$\text{RMSE} = \sqrt{\text{MSE}},$$

to present the results.

▍ 3.5 Transfer Learning from Other Networks

In Section 3.4 we use the centrality data to create the features used for model training. Since these features represent (normalized) global characteristics of nodes and not the specific relations between them (as for example in node2vec or SNoRe), they have the advantage of being transferable between different networks. This gives us the ability to train a model on one network and use it for prediction on a different network.

In this paper, we use the approach outlined in the following paragraphs to train and test a regression model for transfer learning. We will use the term *training network* to highlight the network used for training the model, and *test network* as the network composed of nodes used in prediction of spreading effects.

First we create simulations with nodes from the training network as patient(s) zero, and create target variables as shown in Sections 3.1 and 3.3. After this, we create centrality-based features and use them to train the CaBoost model from Section 3.4. To predict target variables of the test network, we generate its centrality-based features and use them with the previously trained model.

In Section 4.4 we use the following methodology to benchmark the performance of transfer learning models. First we create target variables and centrality-based features of all networks. Then we normalize the features and use all instances to train one CaBoost model for each network. Transfer learning scores are then calculated for each model and each (different) network as the RMSE between the predicted values and target variables. We use five-fold cross-validation as the baseline score for each network.

We represent the performance of transfer learning as a heatmap. The columns of the heatmap represent the test networks, while the rows represent the training network used to create the model. The values on the diagonal represent the RMSE values of the five-fold cross-validation. The other values represent the transfer learning score (score on the test network) divided by the baseline score. The values can be interpreted as the decrease in performance if we use a model trained on another network relative to the estimated performance on the initial network obtained in the process of five-fold cross-validation.

4. Empirical Evaluation

In this section, we present the baselines and datasets used for evaluation and show the empirical results of the approaches outlined in Section 3. We also present how predictions from the SNoRe+features model can be explained with SHAP [31].

4.1 Baselines for Regression (Initial Network)

We compared the results of the proposed method to the following five baselines:

- *Random baseline* creates an embedding of size $|N| \times 64$ with random numbers drawn from Unif(0, 1). We use this embedding as the input data for the XGBoost model.

- *node2vec* [15] learns a low-dimensional representation of nodes that maximizes the likelihood of neighborhood preservation using random walks. During testing, we use the default parameters.

- *SNoRe* [17] learns an interpretable representation of nodes based on the similarity between their neighborhoods. These neighborhoods are created with short random walks. During testing, we use the default parameters.

- *GAT* [14] includes the attention mechanism that helps learn the importance of neighboring nodes. In our tests, we use the implementation from PyTorch Geometric [32].

- *GIN* [27] learns a representation that can provably achieve the maximum discriminative power. In our tests, we use the implementation from PyTorch Geometric [32].

For comparison we also add the averaged simulation error. We calculate this error with the RMSE formula, where $y = 0$ and $f(u)$ is the mean absolute difference between simulation results and their mean value. This baseline corresponds to the situation, where only a single simulation would be used to approximate the expected value of multiple ones (the goal of this work).

4.2 Experimental Setting

We used the following datasets for testing (available at github.com/smeznar/Epidemic-spreading-CN2020): Hamsterster [33], Advogato [34], Wikipedia Vote [35], FB Public Figures [36] and HEP-PH [37] taken from the Network Repository website [38]. Some basic statistics of the networks we used can be seen in Table 3. Two of the networks used during testing are visualized in Figure 2. The network nodes in this figure are colored based on the values of the target variables.

| | | | | Percentage of Nodes in |
Name	Nodes	Edges	Components	Largest Component
Wikipedia Vote [35]	889	2914	1	1.00
Hamsterster [33]	2426	16630	148	0.82
Advogato [34]	6551	43427	1441	0.77
FB Public Figures [36]	11565	67114	1	1.00
HEP-PH [37]	12008	118521	278	0.93

Table 3. Basic statistics of the networks used for testing.

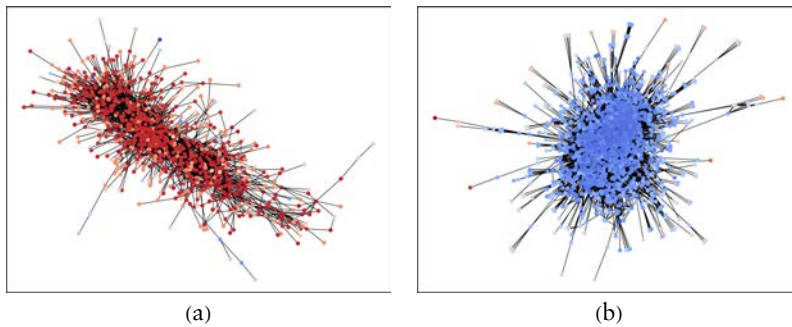(a)   (b)

Figure 2. (a) Visualization of Advogato and (b) Hamsterster networks. The color represents the target value we get when spreading starts from a given node. Color on the Advogato dataset represents the maximum number of infected nodes, while on the Hamsterster dataset time until the maximum number of infected nodes is reached is shown. Blue colors represent low values, while red ones represent high ones. Since nodes with similar centrality values have similar characteristics, these nodes should be colored similarly.

We used the following approach to test the proposed method as well as the baselines mentioned in Section 4.1. We created the target data by simulating 10 epidemics starting from each node of every dataset. We created each simulation using the SIR diffusion model from the NDlib [26] Python library with parameters $\beta = 5\%$ and $\gamma = 0.5\%$. We then created the target variables by identifying and aggregating the maximum number of infected nodes and the time when this happens. We used these target variables to test the methods using five-fold cross-validation. We used XGBoost [28] with default parameters as the regression model with proposed features based on the mentioned centralities, the random baseline, SNoRe [17], SNoRe+centrality features, node2vec [15] and node2vec+centrality features baselines. Baselines GIN and GAT were trained for 200 epochs using the Adam optimizer [39]. Since GIN and GAT are primarily used for node classification, we changed the output layer to a ReLU [40] layer, so they perform regression.

▌ 4.3 Results of Models Trained with Simulation Data

The results of the evaluation described in Section 4.2 are presented in Tables 4–7. Tables 4 and 5 show the results on all the nodes from the network, while Tables 6 and 7 show results only on the nodes from the network's largest component. The results show that the learners significantly outperform the random baseline and the averaged simulation error, especially when predicting effects on networks with several components. Models CaBoost, node2vec+features and SNoRe+features perform significantly better than others and all use centrality-based features to train the XGBoost model. These best approaches achieve RMSE scores around 0.05, which corresponds to an error of around 5% of nodes on average.

Dataset Learner	Advogato	Hamsterster	HEP-PH
CaBoost	0.0519 (± 0.0045)	**0.0429 (± 0.0116)**	0.0481 (± 0.0017)
GAT	0.1748 (± 0.0072)	0.1534 (± 0.0024)	0.1761 (± 0.0019)
GIN	0.0646 (± 0.0238)	0.0712 (± 0.0597)	0.1753 (± 0.0806)
Random	0.3156 (± 0.0024)	0.2915 (± 0.0029)	0.2107 (± 0.0012)
SNoRe	0.1743 (± 0.0057)	0.1591 (± 0.0053)	0.1611 (± 0.0048)
SNoRe+features	**0.0515 (± 0.0044)**	0.0438 (± 0.0114)	**0.0467 (± 0.0018)**
node2vec	0.0673 (± 0.0054)	0.0841 (± 0.0143)	0.0835 (± 0.0031)
node2vec+features	0.0574 (± 0.0037)	0.0431 (± 0.0114)	0.0494 (± 0.0031)
Simulation error	0.0644	0.0576	0.0796

Dataset Learner	FB Public Figures	Wikipedia Vote
CaBoost	0.0521 (± 0.0007)	0.0600 (± 0.0020)
GAT	0.0594 (± 0.0010)	0.0608 (± 0.0013)
GIN	0.0579 (± 0.0015)	0.2076 (± 0.2531)
Random	0.0625 (± 0.0003)	0.0732 (± 0.0039)
SNoRe	0.0600 (± 0.0003)	0.0667 (± 0.0032)
SNoRe+features	**0.0514 (± 0.0004)**	0.0597 (± 0.0009)
node2vec	0.0575 (± 0.0005)	0.0690 (± 0.0021)
node2vec+features	0.0515 (± 0.0004)	**0.0590 (± 0.0010)**
Simulation error	0.0982	0.1064

Table 4. Cross-validation results for maximum number of infected nodes on the whole network.

The results for the prediction of the maximum number of infected nodes on the whole network are shown in Table 4. The results show that the SNoRe+features model has the lowest RMSE on most networks, but that this is mostly because of the centrality-based features, since all the learners that use them give similar results. We also see that graph neural networks perform poorly, on most networks only beating the random baseline. This might be because we use features extracted from the network and a small amount of training data. It is also worth mentioning that the three best-performing models perform

notably better than the averaged simulation error and that the node embedding methods node2vec and SNoRe perform much worse when used without the centrality-based features.

Table 5 shows the performance results for prediction of the time needed to reach the maximum number of infected nodes on the whole network. We see that the SNoRe+features model performs the best overall. This is probably due to features that represent both the similarity between neighborhoods of nodes and their global characteristics. The results also show that GIN and GAT are not suitable for such a task, since they often perform much worse than some other learners (especially GAT) and in some cases worse than the simulation error.

Dataset Learner	Advogato	Hamsterster	HEP-PH
CaBoost	0.0571 (± 0.0032)	**0.0540 (± 0.0037)**	0.0459 (± 0.0012)
GAT	0.1411 (± 0.0021)	0.1027 (± 0.0012)	0.0971 (± 0.0006)
GIN	0.0782 (± 0.0274)	0.0766 (± 0.0162)	0.0717 (± 0.0142)
Random	0.2073 (± 0.0013)	0.1209 (± 0.0014)	0.1095 (± 0.0003)
SNoRe	0.1463 (± 0.0033)	0.1007 (± 0.0022)	0.0904 (± 0.0006)
SNoRe+features	**0.0557 (± 0.0038)**	0.0545 (± 0.0014)	**0.0451 (± 0.0006)**
node2vec	0.0758 (± 0.0014)	0.0824 (± 0.0039)	0.0634 (± 0.0016)
node2vec+features	0.0602 (± 0.0032)	0.0600 (± 0.0035)	0.0467 (± 0.0015)
Simulation error	0.0840	0.0906	0.0839

Dataset Learner	FB Public Figures	Wikipedia Vote
CaBoost	0.0448 (± 0.0005)	0.0647 (± 0.0027)
GAT	0.0642 (± 0.0008)	0.0760 (± 0.0021)
GIN	0.0497 (± 0.0035)	0.0701 (± 0.0069)
Random	0.0817 (± 0.0004)	0.0992 (± 0.0019)
SNoRe	0.0646 (± 0.0011)	0.0753 (± 0.0073)
SNoRe+features	**0.0434 (± 0.0004)**	0.0641 (± 0.0037)
node2vec	0.0590 (± 0.0009)	0.0845 (± 0.0018)
node2vec+features	0.0440 (± 0.0004)	**0.0638 (± 0.0016)**
Simulation error	0.0847	0.1178

Table 5. Cross-validation results for time when maximum number of infected nodes is reached on the whole network.

Similarly to Table 4, Table 6 shows the prediction scores for the maximum number of infected nodes on the largest component of the network. Results for networks Wikipedia vote and FB Public Figures are the same, since they have only one component. Contrary to scores on the whole network, scores on the biggest component show that node2vec+features performs the best overall. We also see that the random baseline performs much better on the single component than on the whole network. This is because the maximum number of infected nodes is usually smaller in smaller components, which makes the mean value of target data smaller and the variance higher. Because of

high variance of target data, the random baseline predicts scores with higher error, since the range of predictions is bigger.

Dataset Learner	Advogato	Hamsterster	HEP-PH
CaBoost	0.0556 (± 0.0011)	**0.0437 (± 0.0017)**	0.0496 (± 0.0004)
GAT	0.0668 (± 0.0088)	0.0455 (± 0.0014)	0.0536 (± 0.0014)
GIN	0.0651 (± 0.0039)	0.0566 (± 0.0057)	0.1147 (± 0.0355)
Random	0.0622 (± 0.0007)	0.0513 (± 0.0014)	0.0556 (± 0.0005)
SNoRe	0.0588 (± 0.0005)	0.0520 (± 0.0014)	0.0551 (± 0.0003)
SNoRe+features	0.0552 (± 0.0007)	0.0447 (± 0.0008)	**0.0482 (± 0.0004)**
node2vec	0.0592 (± 0.0008)	0.0504 (± 0.0012)	0.0520 (± 0.0002)
node2vec+features	**0.0548 (± 0.0010)**	**0.0437 (± 0.0016)**	0.0489 (± 0.0003)
Simulation error	0.0975	0.0769	0.0883

Dataset Learner	FB Public Figures	Wikipedia Vote
CaBoost	0.0521 (± 0.0007)	0.0600 (± 0.0020)
GAT	0.0594 (± 0.0010)	0.0608 (± 0.0013)
GIN	0.0579 (± 0.0015)	0.2076 (± 0.2531)
Random	0.0625 (± 0.0003)	0.0732 (± 0.0039)
SNoRe	0.0600 (± 0.0003)	0.0667 (± 0.0032)
SNoRe+features	**0.0514 (± 0.0004)**	0.0597 (± 0.0009)
node2vec	0.0575 (± 0.0005)	0.0690 (± 0.0021)
node2vec+features	0.0515 (± 0.0004)	**0.0590 (± 0.0010)**
Simulation error	0.0982	0.1064

Table 6. Cross-validation results for maximum number of infected nodes on the biggest component of the network.

Table 7 shows the prediction score of time needed to reach the maximum number of infected nodes on the biggest component of the network. As with the other results, CaBoost, node2vec+features and SNoRe+features give the best performance on all datasets. Compared to the results in Table 5, we see that the difference between the random baseline and other learners is smaller and that the random baseline results are in some cases only around 50% worse than the best-performing learner. Interestingly, the random baseline gives better results overall than the averaged simulation error. This is probably because spreading is "highly" stochastic and simulations can end before spreading begins. In such a case, the averaged simulation error increases significantly, while the random baseline is not affected much, since the model is trained with already processed target data. If we chose a random value $x \in [0, 1]$ as the prediction for the node, the result would be much worse.

We can see that predictions with the proposed learners on all datasets give better results than a single simulation. This shows that such models are useful because they can estimate the joint distribution of spreading across multiple simulations, which is better than a random simulation run.

Dataset Learner	Advogato	Hamsterster	HEP-PH
CaBoost	0.0529 (± 0.0018)	**0.0442 (± 0.0020)**	0.0436 (± 0.0006)
GAT	0.0790 (± 0.0063)	0.0883 (± 0.0031)	0.0643 (± 0.0008)
GIN	0.0614 (± 0.0016)	0.0536 (± 0.0058)	0.0538 (± 0.0118)
Random	0.0855 (± 0.0010)	0.0907 (± 0.0014)	0.0845 (± 0.0003)
SNoRe	0.0702 (± 0.0011)	0.0680 (± 0.0026)	0.0651 (± 0.0005)
SNoRe+features	**0.0517 (± 0.0014)**	0.0454 (± 0.0015)	**0.0425 (± 0.0003)**
node2vec	0.0708 (± 0.0010)	0.0702 (± 0.0018)	0.0557 (± 0.0006)
node2vec+features	0.0519 (± 0.0017)	0.0471 (± 0.0030)	0.0431 (± 0.0002)
Simulation error	0.0941	0.0789	0.0816

Dataset Learner	FB Public Figures	Wikipedia Vote
CaBoost	0.0448 (± 0.0005)	0.0647 (± 0.0027)
GAT	0.0642 (± 0.0008)	0.0760 (± 0.0021)
GIN	0.0497 (± 0.0035)	0.0701 (± 0.0069)
Random	0.0817 (± 0.0004)	0.0992 (± 0.0019)
SNoRe	0.0646 (± 0.0011)	0.0753 (± 0.0073)
SNoRe+features	**0.0434 (± 0.0004)**	0.0641 (± 0.0037)
node2vec	0.0590 (± 0.0009)	0.0845 (± 0.0018)
node2vec+features	0.0440 (± 0.0004)	**0.0638 (± 0.0016)**
Simulation error	0.0847	0.1178

Table 7. Cross-validation results for time when maximum number of infected nodes is reached on the biggest component of the network.

▌ 4.4 Results of Transfer Learning

In this section, we show the results of transfer learning between the presented networks. The results are represented in the form of a heatmap where the values on the diagonal represent the baseline RMSE of five-fold cross-validation and the nondiagonal values represent the RMSE of the dataset in the column with the model trained on the dataset in the row. The error of nondiagonal cells is calculated on all nodes and divided by the baseline score and thus shows how much worse the RMSE we get from transfer learning is when compared with the RMSE we get with the five-fold cross-validation.

The transfer learning results for the prediction of the maximum number of infected nodes can be seen on the heatmap in Figure 3. We can see that most errors are 1–3 times higher than the baseline. The two major exceptions are the results of the Advogato dataset with the FB Public Figures model and the result of the Wikipedia Vote network with the FB Public Figures model. The 5.4 times higher RMSE on the FB Public Figure dataset is probably caused by the big difference between the number of components, since the large number of components lowers the highest number of infected nodes. It is interesting to see that the FB Public Figures model works better than the baseline for the Wikipedia network. This is probably because both

networks have similar structure but Wikipedia vote has fewer nodes and thus less training data. These results show that transfer learning between two topologically similar networks is possible without additional data and can yield good results.

On the heatmap in Figure 4, we see transfer learning results for prediction of the time needed to reach the maximum number of infected nodes. We can see that overall these results are better than those in Figure 3 and that the Advogato dataset performs much worse with other models. This is probably because Advogato has 1441 components while the other networks have significantly fewer. We can also see that the FB Public Figures and Wikipedia vote datasets give good predictions (below two times worse) with all the models, especially in the case where the error is the same as with the baseline.
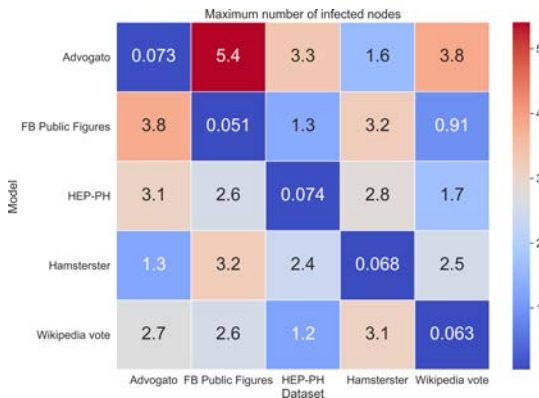
Figure 3. Heatmap with transfer learning results for predictions of maximum number of infected nodes.
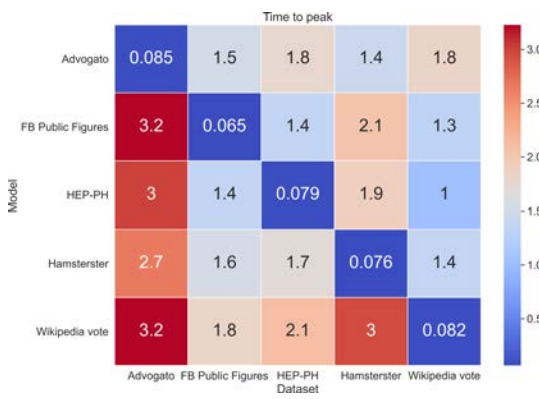

Figure 4. Heatmap with transfer learning results for predictions of time when maximum number of nodes were infected.

The results of transfer learning can be better explained with the distribution plot of target values shown in Figure 5. The first row shows the distribution of the maximum number of infected nodes. We can see that the distributions of FB Public Figures and Wikipedia vote are very similar. This reflects the results, where the Wikipedia vote network performs better with the FB Public Figures model than with the five-fold cross-validation. We also see that the Advogato and FB Public Figures networks have very different distributions. This matches the results, since the transferred model performs very poorly.
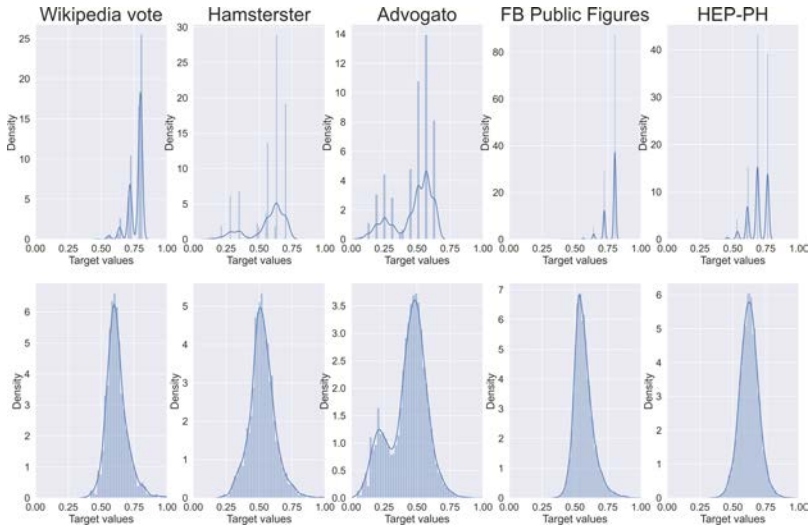
Figure 5. Distribution of target values for the maximum number of infected nodes and the time when this happens. The first row shows the target values for the number of infected nodes, while the second one shows the time when this happens. The x axis represents the value of the target variable. On the other hand, the values of the y axis represent the density at some value.

The second row shows the distribution of time needed to reach the maximum number of infected nodes. As with the maximum number of infected nodes, these distributions also show that the distribution is closely related to how well the model performs. We see that the distribution of target values on the Advogato network vastly differs from the distributions on other networks and that this reflects the results where transfer learning models have higher RMSE. Similarly, the distributions of datasets FB Public Figures, HEP-PH and Wikipedia vote are similar and have transfer learning results that do not differ much from the five-fold cross-validation results.

▌ 4.5 Interpretation of a Prediction

We can explain predictions using model explanation approaches such as SHapley Additive exPlanations (SHAP) [31, 41]. SHAP is a game-theoretic approach for explaining classification and regression models. The algorithm perturbs subsets of input features to take into account the interactions and redundancies between them. The explanation model can then be visualized, showing how the feature values of an instance impact a prediction.

An example of such an explanation is shown in Figure 6 using the model SNoRe+features. We can see that the prediction is impacted mostly by the eigenvector centrality, node 1696, number of second neighbors and the degree centrality. We can also see that a very small value of eigenvector centrality raises the prediction value and that the low values of the number of second neighbors and the degree centrality lower it. This is expected because the low value of eigenvector centrality usually shows that the node is not that "important" and is in a neighborhood with many nodes. Similarly it is expected that the low value of degree centrality and low number of second neighbors lower the prediction because having fewer nodes gives a smaller chance of infection. Lastly, the high similarity between neighborhoods of node 1696 and the instance we try to predict lowers the prediction.
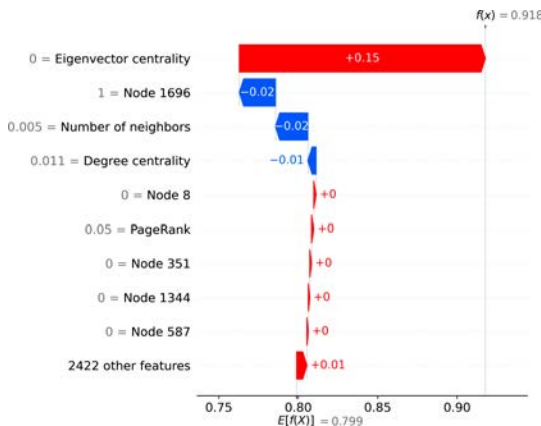

Figure 6. An example of a model explanation for an instance using the SNoRe+features model. Blue arrows indicate how much the prediction is lowered by some feature value, while the red ones indicate how much it is raised. Prediction starts at model's expected value 0.799 and finishes at 0.918. Features and their values are shown on the left. The visualization shows, for example, that the prediction rose from 0.788 to 0.918 because of the low value of eigenvector centrality.

▌5. Discussion and Conclusion

In this paper, we showcase that machine learning methods can be used for fast estimation of epidemic spreading effect from a given node. We show that by reformulating the task as node regression, we can obtain realistic estimations much faster than by performing computationally expensive simulations, even though such simulations are initially used to fine-tune the machine learning models. Further, employment of predictive modeling instead of relying on a single simulation also shows promising results. We also demonstrate that transfer learning can be used to predict spreading effects between networks with similar characteristics without big performance loss.

We show that while graph neural networks outperform the random baseline and can give us great results, centrality scores and embedding feature representation methods coupled with XGBoost mostly outperform them. We also see that machine learning models might overall give a more accurate representation of an epidemic than data gathered from a small number of simulations. This makes the machine learning approach faster and more reliable, while also giving an interpretation of why a node was predicted as it was. Further, this paper demonstrates the complementarity between the accepted simulation-based spreading modeling and fast machine learning–based screening in data-scarce regimes.

A crucial part of our paper shows that transfer of knowledge between networks is possible. This implies that our features capture characteristics that are crucial and transferable between different networks. Since we derive features for models from centralities that are explainable, machine learning models can be used to study which characteristics of the networks play a crucial role in epidemic spreading and how they affect it.

An obvious limitation of the proposed task is that the spreading is probabilistic and even the best classifiers might make significant errors. Similarly, when observing prediction results of the maximum number of infected nodes we must be careful, since we predict an average outcome from some nodes and not the true maximum. This gives us the ability to predict which nodes are the most "dangerous" as patient zero. When trying to predict an outcome of an epidemic that has already spread, one must adjust data accordingly and get rid of simulations where epidemics have not spread.

In future work, we plan to research different centralities and algorithms to better describe network structure and achieve more accurate results. The proposed approach lowers the number of simulations needed to create good approximations, but the approach might still not be scalable to larger networks. In the future, we would like to develop methods to further reduce the number of simulations needed, making the solution more scalable. Another area of our

interest is the ability to solve such tasks by using unsupervised algorithms. Finally, as the current work is focused on the node-level aspects, we believe that similar ideas could be adopted to model higher-order structures and their spreading potential, including convex skeletons and communities.

Acknowledgments

References

[1] A. Guille, H. Hacid, C. Favre and D. A. Zighed, "Information Diffusion in Online Social Networks: A Survey," *ACM Special Interest Group on Management of Data Record*, **42**(2), 2013 pp. 17–28. doi:10.1145/2503792.2503797.

[2] C. Nowzari, V. M. Preciado and G. J. Pappas, "Analysis and Control of Epidemics: A Survey of Spreading Processes on Complex Networks," *IEEE Control Systems Magazine*, **36**(1), 2016 pp. 26–46. doi:10.1109/MCS.2015.2495000.

[3] A. Kacem, C. Lallemand, N. Giraud, M. Mense, M. De Gennaro, Y. Pizzo, J.-C. Loraud, P. Boulet and B. Porterie, "A Small-World Network Model for the Simulation of Fire Spread Onboard Naval Vessels," *Fire Safety Journal*, **91**, 2017 pp. 441–450. doi:10.1016/j.firesaf.2017.04.009.

[4] S. Kesarev, O. Severiukhina and K. Bochenina, "Parallel Simulation of Community-Wide Information Spreading in Online Social Networks," in *Proceedings of the 4th Russian Supercomputing Days (RuSCDays 2018)*, Moscow (V. Voevodin and Sergey Sobolev, eds.), Cham, Switzerland: Springer, 2018 pp. 136–148. doi:10.1007/978-3-030-05807-4_12.

[5] S. Dong, F.-H. Fan and Y.-C. Huang, "Studies on the Population Dynamics of a Rumor-Spreading Model in Online Social Networks," *Physica A: Statistical Mechanics and Its Applications*, **492**, 2018 pp. 10–20. doi:10.1016/j.physa.2017.09.077.

[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, **32**(1), 2020 pp. 4–24. doi:10.1109/TNNLS.2020.2978386.

[7] S. Mežnar, N. Lavrač and B. Škrlj, "Prediction of the Effects of Epidemic Spreading with Graph Neural Networks," in *Complex Networks & Their Applications IX* (R. M. Benito, C. Cherifi, H. Cherifi, E. Moro, L. M. Rocha and M. Sales-Pardo, eds.), Cham, Switzerland: Springer International Publishing, 2021 pp. 420–431. doi:10.1007/978-3-030-65347-7_35.

[8] W. O. Kermack, A. G. McKendrick, "A Contribution to the Mathematical Theory of Epidemics," *Proceedings of the Royal Society of London A*, **115**(772), 1927 pp. 700–721. doi:10.1098/rspa.1927.0118.

[9] D. Kempe, J. Kleinberg and É. Tardos, "Maximizing the Spread of Influence through a Social Network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, New York: Association for Computing Machinery, 2003 pp. 137–146. doi:10.1145/956750.956769.

[10] M. Granovetter, "Threshold Models of Collective Behavior," *American Journal of Sociology*, **83**(6), 1978 pp. 1420–1443. doi:10.1086/226707.

[11] M. Li, X. Wang, K. Gao and S. Zhang, "A Survey on Information Diffusion in Online Social Networks: Models and Methods," *Information*, **8**(4), 2017 118. doi:10.3390/info8040118.

[12] Z. Xiaojin and G. Zoubin, "Learning from Labeled and Unlabeled Data with Label Propagation," Technical Report CMU-CALD-02–107, Carnegie Mellon University, Pittsburgh, 2002.

[13] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France, Conference Track Proceedings, 2017. openreview.net/forum?id=SJU4ayYgl.

[14] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph Attention Networks," in *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, Vancouver, Canada, Conference Track Proceedings, 2018. openreview.net/forum?id=rJXMpikCZ.

[15] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, San Francisco, CA (B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen and R. Rastogi, eds.), New York: Association for Computing Machinery, 2016 pp. 855–864. doi:10.1145/2939672.2939754.

[16] B. Škrlj, N. Lavrač and J. Kralj, "Symbolic Graph Embedding Using Frequent Pattern Mining," *Discovery Science (DS 2019)*, Split, Croatia (P. K. Novak, T. Šmuc and S. Džeroski, eds.), Cham, Switzerland: Springer International Publishing, 2019 pp. 261–275. doi:10.1007/978-3-030-33778-0_21.

[17] S. Mežnar, N. Lavrač and B. Škrlj, "SNoRe: Scalable Unsupervised Learning of Symbolic Node Representations," *IEEE Access,* **8**, 2020 pp. 212568–212588. doi:10.1109/ACCESS.2020.3039541.

[18] B. Perozzi, R. Al-Rfou and S. Skiena, "Deepwalk: Online Learning of Social Representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, New York (S. A. Macskassy, C. Perlich, J. Leskovec, W. Wang and R. Ghani, eds.), New York: Association for Computing Machinery, 2014 pp. 701–710. doi:10.1145/2623330.2623732.

[19] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong and Q. He, "A Comprehensive Survey on Transfer Learning," *Proceedings of the IEEE,* **109**(1), 2020 pp. 43–76. doi:10.1109/JPROC.2020.3004555.

[20] C.-K. Lee, C. Lu, Y. Yu, Q. Sun, C.-Y. Hsieh, S. Zhang, Q. Liu and L. Shi, "Transfer Learning with Graph Neural Networks for Optoelectronic Properties of Conjugated Oligomers," *The Journal of Chemical Physics,* **154**(2), 2021 024906. doi:10.1063/5.0037863.

[21] T. Mallick, P. Balaprakash, E. Rask and J. Macfarlane, "Transfer Learning with Graph Neural Networks for Short-Term Highway Traffic Forecasting," arxiv.org/abs/2004.08038.

[22] W. Wang, V. W. Zheng, H. Yu and C. Miao, "A Survey of Zero-Shot Learning: Settings, Methods, and Applications," *ACM Transactions on Intelligent Systems and Technology*, **10**(2), 2019 pp. 1–37. doi:10.1145/3293318.

[23] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang and E. P. Xing, "Rethinking Knowledge Graph Propagation for Zero-Shot Learning," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)*, Long Beach, CA, Los Alamitos, CA: IEEE Computer Society, 2019 pp. 11479–11488. doi:10.1109/CVPR.2019.01175.

[24] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang and J. Tang, "Gcc: Graph Contrastive Coding for Graph Neural Network Pre-training," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*, 2020 pp. 1150–1160. doi:10.1145/3394486.3403168.

[25] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande and J. Leskovec, "Strategies for Pre-training Graph Neural Networks," arxiv.org/abs/1905.12265.

[26] G. Rossetti, L. Milli, S. Rinzivillo, A. Sîrbu, D. Pedreschi and F. Giannotti, "NDLIB: A Python Library to Model and Analyze Diffusion Processes over Complex Networks," *International Journal of Data Science and Analytics,* **5**(1), 2018 pp. 61–79. doi:10.1007/s41060-017-0086-6.

[27] K. Xu, W. Hu, J. Leskovec and S. Jegelka, "How Powerful Are Graph Neural Networks?," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, New Orleans, LA, 2019. openreview.net/forum?id=ryGs6iA5Km.

[28] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, San Francisco, New York: Association for Computing Machinery, 2016 pp. 785–794. doi:10.1145/2939672.2939785.

[29] F. A. Rodrigues, "Network Centrality: An Introduction," *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems* (E. Macau, ed.), Cham, Switzerland: Springer, 2019 pp. 177–196. doi:10.1007/978-3-319-78512-7_10.

[30] L. Page, S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Technical Report 422, Stanford InfoLab. ilpubs.stanford.edu:8090/422.

[31] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*, Long Beach, CA (U. von Luxburg, I. Guyon, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan and R. Garnett, eds.), Red Hook, NY: Curran Associates, Inc., 2017 pp. 4768–4777. dl.acm.org/doi/10.5555/3295222.3295230.

[32] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. (Nov 3, 2021) rlgm.github.io/papers/2.pdf.

[33] Hamsterster. "Hamsterster Social Network." (Oct 4, 2021) networkrepository.com/soc-hamsterster.php.

[34] P. Massa, M. Salvetti and D. Tomasoni, "Bowling Alone and Trust Decline in Social Network Sites," in *Proceedings of the Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC'09)*, Chengdu, China (B. Yang, W. Zhu, Y. Dai, L. T. Yang and J. Ma, eds.), Los Alamitos, CA: IEEE Computer Society, 2009 pp. 658–663. doi:10.1109/DASC.2009.130.

[35] J. Leskovec, D. P. Huttenlocher and J. M. Kleinberg, "Signed Networks in Social Media," in *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI 2010)*, Atlanta, GA (E. D. Mynatt, D. Schoner, G. Fitzpatrick, S. E. Hudson, W. K. Edwards and T. Rodden, eds.), New York: Association for Computing Machinery, 2010 pp. 1361–1370. doi:10.1145/1753326.1753532.

[36] B. Rozemberczki, R. Davies, R. Sarkar and C. Sutton, "GEMSEC: Graph Embedding with Self Clustering, in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '19)*, Vancouver, Canada (F. Spezzano, W. Chen and X. Xiao, eds.), New York: Association for Computing Machinery, 2019 pp. 65–72. doi:10.1145/3341161.3342890.

[37] J. Leskovec, J. Kleinberg and C. Faloutsos, "Graph Evolution: Densification and Shrinking Diameters," *ACM Transactions on Knowledge Discovery from Data*, **1**(1), 2007 p. 2–es. doi:10.1145/1217299.1217301.

[38] R. A. Rossi and N. K. Ahmed, "The Network Data Repository with Interactive Graph Analytics and Visualization," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, TX (B. Bonet and S. Koenig, eds.), AAAI Press, 2015 pp. 4292–4293. www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/download/ 9553/9856.

[39] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the 3rd International Conference for Learning Representations (ICLR 2015)*, San Diego, CA (Y. Bengio and Y. LeCun, eds.), Conference Track Proceedings, 2015.

[40] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML '10)*, Haifa, Israel (J. Fürnkranz and T. Joachims, eds.), Madison, WI: Omnipress, 2010 pp. 807–814. dl.acm.org/doi/10.5555/3104322.3104425.

[41] E. Štrumbelj and I. Kononenko, "Explaining Prediction Models and Individual Predictions with Feature Contributions," *Knowledge and Information Systems*, **41**(3), 2014 pp. 647–665. doi:10.1007/s10115-013-0679-x.