ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Biochemical Pathway Robustness Prediction with Graph Neural Networks

Marco Podda[1], Davide Bacciu[1], Alessio Micheli[1], Paolo Milazzo[1] *

[1] University of Pisa - Department of Computer Science
Largo Bruno Pontecorvo 3, 56127, Pisa - Italy

Abstract. The robustness property of a biochemical pathway refers to maintaining stable levels of molecular concentration against the perturbation of parameters governing the underlying chemical reactions. Its computation requires an expensive integration in parameter space. We present a novel application of Graph Neural Networks (GNN) to predict robustness indicators on pathways represented as Petri nets, without the need of performing costly simulations. Our assumption is that pathway structure alone is sufficient to be effective in this task. We show experimentally for the first time that this is indeed possible to a good extent, and investigate how different architectural choices influence performances.

1 Introduction

Biological pathways describe the complex interactions between molecules at the biochemical level. Pathways are usually represented as graphs, while Ordinary Differential Equations (ODEs) are often used to investigate their dynamical properties. Among them, robustness [1] is one of particular relevance. It can be defined on a pair of input and output molecules, and it quantifies the stability of the steady-state concentration of the output against perturbations in the initial concentration of the input (α-robustness, [2]). To assess robustness, the graph representation of the pathway must be translated into a set of ODEs. Then, the property is calculated via integration in parameter space, which requires a massive amount of computational resources. We present a novel and challenging application of Graph Neural Networks (GNNs) [3, 4] in the field of computational biology. Specifically, we introduce the use of GNNs for pathway robustness prediction. The assumption underlying this study is that the structure of a pathway contains sufficient information to predict robustness indicators without having to perform expensive numerical simulations. To test this assumption, we apply GNNs to a dataset of pathways graphically represented as Petri nets [5]. We perform an extensive evaluation of six GNN variants on the task, studying how the number of layers affects performances. Our experiments show that GNNs are indeed capable of predicting robustness to a good extent. These preliminary results can ultimately lead to faster advances in the field: in particular, this approach could relieve researchers from the need of performing expensive simulations in order to evaluate not only robustness, but also dynamical properties such as bistability, monotonicity and oscillations.

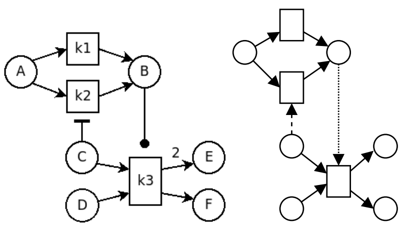
*This work has been partially supported by the Italian Ministry of Education, University, and Research (MIUR) under project SIR 2014 LIST-IT (grant n. RBSI14STDE).

$$A \xrightarrow{-/-} B$$
$$B \xrightarrow{-/C} A$$
$$C + D \xrightarrow{B/-} 2E + F$$

(a)

$$\begin{aligned}
\tfrac{dA}{dt} &= -k_1 A + k_2 \tfrac{B}{C}\\
\tfrac{dB}{dt} &= k_1 A - k_2 \tfrac{B}{C}\\
\tfrac{dC}{dt} &= -k_3 CDB\\
\tfrac{dD}{dt} &= -k_3 CDB\\
\tfrac{dE}{dt} &= 2k_3 CDB\\
\tfrac{dF}{dt} &= k_3 CDB
\end{aligned}$$

(b)



(c) (d)

Fig. 1: a) A set of chemical reactions in arrow notation, with promoters/inhibitors above the arrows. b) The associated ODE system (example). c) The Pathway Petri net corresponding to the chemical reactions. d) An example of training graph: circle nodes are molecules, box nodes are reactions. Different edge styles identify the three possible relations (standard, promoter, inhibitor).

2 Background and Notation

A chemical reaction is a process that transforms a group of molecules (*reactants*) into another (*products*). Reactions are governed by *kinetic constants*, that specify the rate at which the reaction can occur, given the concentrations of the reactants in a chemical solution. In reactions, molecules can also play the roles promoters (reaction facilitators) or inhibitors (reaction blockers). A biochemical pathway is a system of related chemical reactions, where the product of one may become the reactant of another. Petri nets are a common modeling notation for pathways. We consider here a variant of Petri nets that suitably considers promoters and inhibitors, that we call Pathway Petri nets (PPNs). The semantics of a PPN is described by an ODE system that models the associated set of reactions. The state of a PPN (called marking) corresponds to an assignment of positive real values to the variables associated to the system. Given M, the set of possible markings, a PPN is defined as a tuple $\mathcal{P} = (P, T, f, p, h, \delta, m_0)$ where:

- P and T are finite, non empty, disjoint sets of *places* and *transitions*;
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}^{\geq 0}$ defines the set of directed arcs, weighted by non-negative integers (whose value corresponds to the multiplicity of the reactants/products in the reaction);
- $p, h \subseteq (P \times T)$ are the sets of promotion and inhibition arcs;
- $\delta : T \to \Psi$, with $\Psi = M \to \mathbb{R}^{\geq 0}$, is a function that assigns to each transition a function corresponding to the computation of a kinetic formula to every possible marking $m \in M$;
- $m_0 \in M$ is the initial marking.

Given a PPN $\mathcal{P}$, we can cast it into a graph $G_{\mathcal{P}} = \langle V, E \rangle$ as follows. First we define the node sets V_{mol} for molecules, and V_{react} for reactions, and set

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

$V_{mol} = P$, $V_{react} = T$. We then define three edge sets: E_{std} for standard edges, E_{pro} for promoting edges, and E_{inh} for inhibition edges; and set $E_{std} = \{\langle u, v\rangle \in (P \times T) \cup (T \times P) \mid f(\langle u,v\rangle) > 0\}$, $E_{pro} = p$, $E_{inh} = h$. Finally, we impose $V = V_{mol} \cup V_{react}$ and $E = E_{std} \cup E_{pro} \cup E_{inh}$[1]. Note that information about kinetic formulae, reactant multiplicities, the transition function δ, and the initial marking m_0 are not included in the graph. This corresponds to leveraging only the topology of the pathway, discarding all information related to the mechanics of the reactions. Figure 1 shows a set of reactions modeled as a PPN and the corresponding graph. Given a graph $G_{\mathcal{P}}$ let us define its *augmented* version $\bar{G}_{\mathcal{P}} = \langle \bar{V}, \bar{E}\rangle$ as follows: $\bar{V} = V$, and $\bar{E} = \bar{E}_{std} \cup E_{pro} \cup E_{inh}$, with $\bar{E}_{std} = E_{std} \cup \{\langle v, u\rangle \mid \langle u, v\rangle \in E_{std}, u \in V_{mol}, v \in V_{react}\}$. Informally, for each standard edge in E connecting a molecule to a reaction, we add to $\bar{E}$ the same edge with reversed direction. This encodes the notion that changing reaction rates directly influences the consumption of the reactant. Intuitively, this graph represents influence relations, rather than reaction dynamics. Furthermore, given a graph $G_{\mathcal{P}}$ and a pair of nodes $u, v \in V_{mol}$, let us define $X^{uv}_{G_{\mathcal{P}}}$, the *subgraph of* $G_{\mathcal{P}}$ *induced by (u,v)*, informally as follows: $X^{uv}_{G_{\mathcal{P}}}$ is the smallest subgraph of $G_{\mathcal{P}}$ whose node set contains u, v, as well as nodes in every possible oriented path from u to v in $\bar{G}_{\mathcal{P}}$. Note that, although its node set is computed using $\bar{G}_{\mathcal{P}}$, $X^{uv}_{G_{\mathcal{P}}}$ is a subgraph of $G_{\mathcal{P}}$. Finally, let us define the neighborhood function of $G_{\mathcal{P}}$ as $\mathcal{N}(u) = \{v \in V \mid \langle u, v\rangle \in E\}$.

3 Task

We start from a set of graphs $\mathcal{G} = \{G_{\mathcal{P}_1}, G_{\mathcal{P}_2}, \ldots, G_{\mathcal{P}_N}\}$ with $N = 706$, representing PPNs taken from the BioModels [6] database[2] (we will hereafter drop the dependency on $\mathcal{P}$ for ease of notation). Each graph G_i is associated with a set of tuples $\mathcal{T}_{G_i} = \{(u, v, r) \mid u, v \in V_{mol}\}$. Our targets $r \in [0, 1] \subseteq \mathbb{R}$ are the robustness values[3] associated to the pair of input/output nodes (u, v), where 1 is the maximum robustness possible. We frame the problem as a classification task, transforming these values into indicators $y = \mathbb{I}(r)$ by rounding them to the nearest integer. We then extract, for each graph, the set of its induced subgraphs $\mathcal{X}_{G_i} = \{(X^{uv}_{G_i}, y) \mid (u, v, r) \in \mathcal{T}_{G_i}, y = \mathbb{I}(r)\}$. Our dataset is thus defined as $\mathcal{D} = \bigcup_{G_i \in \mathcal{G}} \{\mathcal{X}_{G_i}\}$. In this work, induced subgraphs with number of nodes < 40 are used, which allows us to work with a set of graphs of homogeneous size. The resulting dataset is made of 7013 induced subgraphs. With this newly defined dataset, the learning problem we face is to find a model $f(X_G)$ which, given an unseen induced subgraph, predicts the associated robustness indicator with "good" accuracy. In other words, we wish to minimize $\mathcal{L}(\mathcal{D}) = 1/|\mathcal{D}| \cdot \sum_{(X_G, y) \in \mathcal{D}} \epsilon_1 \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$, with $\hat{y} = f(X_G)$, that is, the weighted binary cross-entropy between the predicted output of the model

[1]Note that $V_{mol} \cap V_{react} = \emptyset$ and $E_{std} \cap E_{pro} \cap E_{inh} = \emptyset$.

[2]Specifically, all the manually-curated models in the BioModels database at the time we performed our experiments.

[3]Pre-computed as a normalized relative variant of α-robustness [2] from results of numerical simulations performed using the `libRoadRunner` Python library [7].

$\hat{y}$ and the actual indicator y. Weighting with $\epsilon_1 = \frac{\#\text{negative examples}}{\#\text{positive examples}}$ is needed to mitigate the imbalance in favor of the positive class (72%-28%).

4 Model

Our model uses GNNs to embed the structure of subgraphs X_G. GNNs are based on the notion of state, a vector associated to each node in the graph which is updated iteratively according to a message-passing schema [8]. The initial state is set to be a vector of features: in our case, each subgraph node contains a binary feature vector of size 3, where the first position encodes node type (0 for molecule, 1 for reaction); the second position encodes whether a node is an input node (1) or not (0); the third position encodes whether a node is a output node (1) or not (0). In general, the ℓ-th layer of a GNN updates the state of a node v as $\mathbf{h}_v^{\ell+1} = \sigma(\mathbf{w}^\ell\ U(\mathbf{h}_v^\ell, C(\{\mathbf{h}_u^{\ell+1} \mid u \in \mathcal{N}(v)\}))$, where C is a function that aggregates nodes in the neighborhood of the current node; U is an update function that combines $\mathbf{h}_v$, the current state of the node with the aggregated state of its neighbors; $\mathbf{w}^\ell$ are adaptive weights; and σ is a nonlinearity (ReLU in our case). To build a graph-level representation at layer ℓ, node representations are aggregated by a permutation-invariant readout function $\mathbf{h}^\ell = R(\{\mathbf{h}_v^\ell \mid v \in V\})$. Note that different GNNs can be derived choosing different C, U, and R. The final representation of the graph is obtained concatenating the representations of each layer; more formally, $\mathbf{h}_G = [\mathbf{h}^1; \mathbf{h}^2; \ldots; \mathbf{h}^L]$, where ; denotes concatenation, and L is the number of GNN layers. We denote the process of obtaining the final graph representation as $\mathbf{h}_G = \text{GNN}(X_G)$. The graph representation is then fed to a Multi-Layer Perceptron (MLP) classifier with two hidden layers with ReLU activations, and sigmoid outputs that compute the robustness probability associated to the input subgraph. Summing up, our model f is implemented as the following neural network: $f(X_G) = \text{MLP}(\text{GNN}(X_G))$, where we omit the parameterization for ease of notation.

5 Experiments

We thoroughly assess the performance of the model described in Section 4 using nested Cross-Validation (CV) [9] for generalization accuracy estimation, with an outer 5-fold CV for performance assessment and an internal holdout split (90% training, 10% validation) for model selection. The model is selected with grid search on the following hyper-parameters: size of the GNN embedding (choosing between 64 and 128 per layer), number of GNN layers (choosing from $\{1, \ldots, 8\}$), and type of readout function (choosing among element-wise mean, max or sum). We evaluate 6 different models, determined by the type of convolution adopted and whether or not edge features are learned. Specifically, we evaluate three different convolutions: Graph Convolutional Network (GCN) [10], Graph Isomorphism Network (GIN) [11] and Weisfeiler-Lehman GCN (WLGCN) [12]. For each of them, we evaluate a vanilla variant which uses the state update function described in Section 4, as well as an edge-aware variant (following [13]) where each edge type has its own set of adaptive weights. Formally, the edge-aware variants update the state as $\mathbf{h}_v^{\ell+1} = \sigma(\sum_{k \in K} \mathbf{w}_k^{\ell}\ U(\mathbf{h}_v^{\ell},\ C(\{\mathbf{h}_u^{\ell+1} \mid u \in \mathcal{N}(v,k)\}))$, where k ranges over all K possible edge types, and $\mathcal{N}(v,k)$ is a neighborhood function that only selects neighbors of v connected by an edge of type k. All models are trained with the Adam optimizer, using a learning rate of 0.001 and scheduled annealing with a shrinking factor of 0.6 every 50 epochs. The size of the two hidden layers of the MLP is 128 and 64, with dropout rate of 0.1. All models are trained for a total of 500 epochs; we use early stopping on the validation accuracy with 100 epochs of patience. In the assessment phase, the model is trained three different times to account for random initialization effects; the resulting accuracies are averaged to obtain the final test fold accuracy. The experiments are implemented using the `PyTorch Geometrics` library [14].

6 Results and Discussion

Model	Test accuracy
Baseline	0.7322 ± 0.0000
GCN-vanilla	0.8573 ± 0.0087
GIN-vanilla	0.8567 ± 0.0137
WLGCN-vanilla	0.8624 ± 0.0088
GCN	0.8692 ± 0.0140
GIN	0.8684 ± 0.0078
WLGCN	0.8687 ± 0.0117

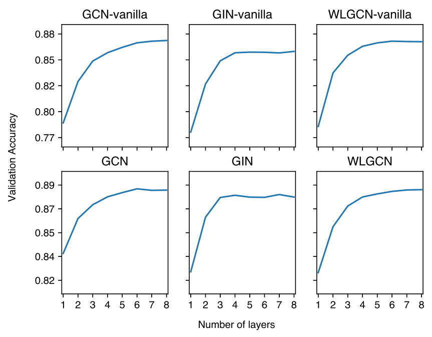


Fig. 2: Left: evaluation results. Right: plot of the change in validation accuracy as a function of the number of GNN layers.

Figure 2 (left) shows the results of the assessed models, corresponding to the mean and standard deviation of the accuracy obtained in the 5 evaluation folds. To double-check the significance of our results, we also report the results of a baseline that simply predicts the most frequent class in the dataset. From the analysis, three results emerge: *a*) the convolution type does not seem to be a relevant factor for good performances: in fact, all models perform very similarly despite using different types of graph convolutions; *b*) as expected, learning edge features provides a slight increase in performances: in fact, all variants that leverage edge features obtain an improvement, up to approximately 1.2% in the case of the GCN-based model; *c*) all examined GNNs significantly outperform the baseline. This demonstrates the validity of our assumption, i.e. the structure of the pathway contains enough information to predict robustness to some extent, and GNNs are effective at extracting such information. The fact that the accuracy plateaus can be related to the influence of the reaction parameters (which we do not consider). As an additional contribution, we study how GNN layering affects performances. To do so, we perform an ablation study where we stratify the results of the model selection by number of GNN layers, and report the mean and standard deviations of the related validation accuracies. Note that, although validation accuracy is in general an over-estimate of the true accuracy, the relative difference in performance as the number of layers change stays proportional independently of the data used. Figure 2 (right) shows that, for all considered GNNs, increasing the number of layers improves accuracy, up to a certain depth where it becomes stable. This provides evidence that "deep" GNNs are necessary to obtain good performances in this task.

7 Conclusions

In this work we have shown, for the first time, that GNNs can be effective at predicting the dynamical property of robustness of pathway networks, leveraging only their structure. Future works will be aimed to extend this result to larger graphs as well as to other interesting dynamical properties.

References

[1] H. Kitano. Biological robustness. *Nature Reviews Genetics*, 5(11):826, 2004.

[2] L. Nasti, R. Gori, and P. Milazzo. Formalizing a notion of concentration robustness for biochemical networks. In *STAF Workshops*, volume 11176 of *LNCS*, pages 81–97. Springer, 2018.

[3] F. Scarselli, M. Gori, et al. The Graph Neural Network Model. *Trans. Neur. Netw.*, 20(1):61–80, 2009.

[4] A. Micheli. Neural Network for Graphs: A Contextual Constructive Approach. *Trans. Neur. Netw.*, 20(3):498–511, 2009.

[5] V. N. Reddy, M. L. Mavrovouniotis, et al. Petri net representations in metabolic pathways. In *ISMB*, volume 93, pages 328–336, 1993.

[6] C. Li, M. Donizelli, et al. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4:92, 2010.

[7] E. T. Somogyi, J. M. Bouteiller, et al. libRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics*, 31(20):3315–3321, 2015.

[8] J. Gilmer, S. S. Schoenholz, et al. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017.

[9] S. Varma and R. Simon. Bias in Error Estimation When Using Cross-Validation for Model Selection. *BMC bioinformatics*, 7:91, 2006.

[10] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

[11] K. Xu, W. Hu, et al. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

[12] C. Morris, M. Ritzert, et al. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proceedings of the 33rd Conference on Artificial Intelligence*, AAAI '19, pages 4602–4609, 2019.

[13] M. S. Schlichtkrull, T. N. Kipf, et al. Modeling Relational Data with Graph Convolutional Networks. In *Proceedings of the 15th International Conference on The Semantic Web*, ESWC '18, pages 593–607, 2018.

[14] M. Fey and J. E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.