BMC Bioinformatics

RESEARCH

Open Access

Sensitivity analysis on protein-protein interaction networks through deep graph networks

Alessandro Dipalma[1*], Michele Fontanesi[1], Alessio Micheli[1], Paolo Milazzo[1] and Marco Podda[1]

*Correspondence:
alessandro.dipalma@phd.unipi.it

[1] Department of Computer Science, University of Pisa, Largo Bruno Pontecorvo, 3, 56125 Pisa, PI, Italy

Abstract

Background: Protein-protein interaction networks (PPINs) provide a comprehensive view of the intricate biochemical processes that take place in living organisms. In recent years, the size and information content of PPINs have grown thanks to techniques that allow for the functional association of proteins. However, PPINs are static objects that cannot fully describe the dynamics of the protein interactions; these dynamics are usually studied from external sources and can only be added to the PPIN as annotations. In contrast, the time-dependent characteristics of cellular processes are described in Biochemical Pathways (BP), which frame complex networks of chemical reactions as dynamical systems. Their analysis with numerical simulations allows for the study of different dynamical properties. Unfortunately, available BPs cover only a small portion of the interactome, and simulations are often hampered by the unavailability of kinetic parameters or by their computational cost. In this study, we explore the possibility of enriching PPINs with dynamical properties computed from BPs. We focus on the global dynamical property of sensitivity, which measures how a change in the concentration of an input molecular species influences the concentration of an output molecular species at the steady state of the dynamical system.

Results: We started with the analysis of BPs via ODE simulations, which enabled us to compute the sensitivity associated with multiple pairs of chemical species. The sensitivity information was then injected into a PPIN, using public ontologies (BioGRID, UniPROT) to map entities at the BP level with nodes at the PPIN level. The resulting annotated PPIN, termed the DyPPIN (Dynamics of PPIN) dataset, was used to train a DGN to predict the sensitivity relationships among PPIN proteins. Our experimental results show that this model can predict these relationships effectively under different use case scenarios. Furthermore, we show that the PPIN structure (i.e., the way the PPIN is "wired") is essential to infer the sensitivity, and that further annotating the PPIN nodes with protein sequence embeddings improves the predictive accuracy.

Conclusion: To the best of our knowledge, the model proposed in this study is the first that allows performing sensitivity analysis directly on PPINs. Our findings suggest that, despite the high level of abstraction, the structure of the PPIN holds enough information to infer dynamic properties without needing an exact model of the underlying processes. In addition, the designed pipeline is flexible and can be easily integrated into drug design, repurposing, and personalized medicine processes.

Keywords: Protein-protein interaction, Sensitivity analysis, Deep graph networks

Introduction

Protein-protein interaction networks (PPINs) represent the set of all known Protein-Protein Interactions (PPIs) within cells. Existing PPIs can be of different nature: *physical* interactions describe known reactions between proteins, *genetic* interactions describe correlations in the expression of genes coding for two proteins, and *predicted* interactions are documented via correlation studies or literature mining.

PPINs are crucial for understanding cellular function; to help in this endeavor, a variety of PPI databases have been developed. These differ in the types of interactions they include, the level of detail provided, and their intended user base [1]. Well-established examples of publicly available PPIN databases include: STRING [2], BioGRID [3] and IntAct [4]. The integration of PPI data with other biological and biomedical information has been a significant research focus, enabling advances in protein interaction prediction, protein complex identification, drug-disease and drug-target associations. These integrations often use shared ontologies to describe biochemical entities, such as UniPROT for proteins [5].

Major PPI repositories depict a static snapshot of the interactome, which is instead inherently dynamic; while interaction networks can provide insights about a biological system organization, they lack explicit modeling of mechanistic insights. The dynamic nature of biological systems is instead modeled by Biochemical Pathways (BPs), which describe a series of interconnected biochemical reactions allowing the investigation of their dynamics. Similarly to PPIs, BPs are collected into publicly available databases, such as Kegg [6], Reactome [7] and BioModels [8]. The behavior of BPs can be described by different dynamical properties, which quantify how the concentrations of molecules involved in the BP change over time in response to different stimuli.

In particular, the focus of this study is on the dynamical property of *sensitivity*, which measures how the change in concentration of an input protein influences the concentration of an output protein. Sensitivity, similarly to other relevant dynamical properties, is often analyzed using ordinary differential equations (ODEs) or stochastic approaches like the Gillespie algorithm [9]. These methods, however, can be applied only if the kinetic parameters of all biochemical reactions are available, which is often not the case. Among the mentioned databases, BioModels contains the largest number of simulation-ready BPs.

In contrast with BPs, PPINs do not contain the kinetic and quantitative information necessary to perform simulations. They can be seen as a high-level abstraction of all the BPs involving the proteins, and this abstraction allows studying interactions belonging to both known and putative BPs. It is also known that these networks exhibit complex structure, the analysis of which allows deriving insights about the dynamics of the biological dynamical system that they depict. In particular, identifying a correlation between the BP dynamics and the topology of a related portion of the PPIN could enable the study of sensitivity (or other dynamical properties) directly from PPINs. In turn, given the extensive interactome coverage of current PPINs, the scope of these analyses could be widened to include PPIN portions for which a detailed description of the

underlying BPs is not available. With these motivations, this work investigates the possibility of inferring the sensitivity property directly at the PPIN level, rather than at the BP level.

The workflow of this study is shown in Fig. 1. It comprises a dataset extraction phase (in blue), a model training phase (in orange), and a final inference phase (in yellow) which pertains to the end user. In the dataset extraction phase, we started by analyzing a set of BPs taken from the BioModels database using ODE simulations. These simulations allowed us to compute the sensitivity of a large set of input/output pairs of molecular species. Then, we devised a mapping from BP proteins and complexes to PPIN nodes, which allowed us to transfer the sensitivity annotations from the BPs to the corresponding portion of the PPIN. The annotated PPIN was used to create a dataset whose examples are labeled PPIN subgraphs. Each subgraph is induced by an input protein and an output protein, and the corresponding label indicates whether the concentration of the output protein is sensitive to changes in the concentration of the input protein.

Given that the examples in the dataset are subgraphs, we choose to tackle the sensitivity prediction task with a class of models specifically designed to natively process graph-structured data, namely Deep Graph Networks (DGNs) [10]. Thus, in the second phase, we trained a DGN to predict the sensitivity from PPIN subgraphs. More precisely, we built a model that takes as input a PPIN subgraph (containing the input/output proteins on which sensitivity needs to be assessed), and outputs the corresponding sensitivity prediction. We experimentally evaluated the performances of the model to establish its generalization capabilities under different training and usage scenarios.

In the final inference phase, the trained DGN can be used to predict the sensitivity of unseen PPIN subgraphs (for example, induced by some input/output proteins under study). A major advantage of our approach is that, once the model is trained, the sensitivity can be predicted directly at the PPIN level, bypassing the need for simulations or predictions at the BP level. Correspondingly, the time to issue a prediction with the trained model is orders of magnitude faster than running numerical simulations, making the developed method suitable for large scale studies.

To ground our findings in real-world scenarios, we applied our methodology to a real-world use case to showcase its flexibility and generalizability. Specifically, we used our

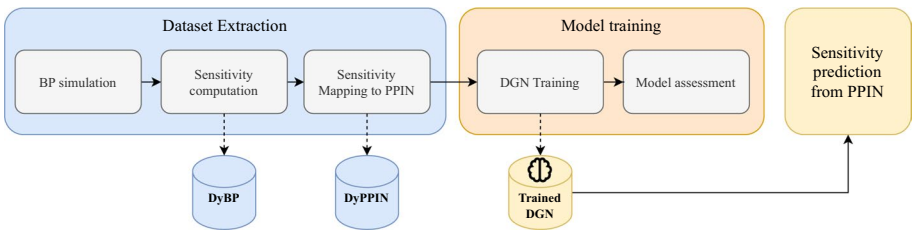


Fig. 1 This work is logically divided into three phases: dataset extraction (in blue), model training (in orange), and sensitivity prediction (in yellow). In the first phase, we performed numerical simulations on a set of manually curated BPs. This process resulted in a preliminary dataset (DyBP) where BPs were annotated with sensitivity information. Then, we mapped the sensitivity relationships in the DyBP dataset to a PPIN, producing a second dataset (DyPPIN). Both datasets are made publicly available. In the second phase, we trained a DGN on the DyPPIN dataset to predict sensitivity relationships, assessing its performances in different use cases. Ultimately (third phase), the end user can use the trained DGN to predict the sensitivity across different portions of the PPIN. The inference phase is visually detailed in Fig. 4

model to predict the sensitivity of diabetes-related proteins (insulin and glucagon) to changes in concentration of known regulatory genes. Crucially, this analysis was performed by only considering the interaction network structure, and purposedly neglecting gene expression annotations. The results highlight that even in this challenging conditions, the predictions of our model align with the biological expectations. To the best of our knowledge, ours is the first method that allows to predict sensitivity from PPINs, opening up to enhancing existing pipelines in practical applications such as drug-target identification, drug repurposing, and personalized medicine. Furthermore, the approach we present is general, and could be seamlessly applied to the prediction of different dynamical properties in forthcoming studies.

Further relevant contributions of this work are the publication of two datasets describing sensitivity annotations for BPs and for protein pairs in PPINs, as well as the development of a command-line tool to perform the sensitivity prediction on arbitrary PPIN subgraphs.

The paper is organized as follows: Sect. 1.1 introduces the application context of PPIN and an overview of machine learning methods to process and extract knowledge from them. We then discuss how the problem of modeling the dynamics of biological networks has been faced in literature (Sect. 1.1.1). Section 1.2 discusses how dynamical properties of biochemical pathways can be predicted from graphs (Sect. 1.2.1) and introduces the DGN framework (Sect. 1.2.2). Section 2 presents the process for dataset extraction, DGN training, and sensitivity inference. Section 3 presents the outcomes of the developed methodology: the characteristics of the extracted dataset are presented in Sect. 3.1, while the model performances are discussed in Sect. 3.2 and 3.3. Section 4 presents example of how the designed pipeline could be used to perform predictions, along with a discussion of the possible application scenarios that could take advantage of them. Finally, we draw our conclusion in Sect. 5.

Related works

Over the years, a substantial research effort has been put into integrating PPI data with other biological and biomedical information to infer new knowledge. Typical learning tasks are PPI prediction [11], protein complex identification [12], drug-disease [13] and drug-target association [14]. In the following, we briefly survey methods that have researched this integration by making use of protein sequences, structure, functional annotations, and interaction data, with a focus on methods that employ machine learning techniques.

Many techniques have been proposed to predict interaction leveraging protein metadata [15], protein 3D structure [16], and protein amino acidic sequences [17, 18]. Since the knowledge about individual proteins can be incomplete, different methods have been designed to infer protein characteristics by analyzing other proteins that are supposed to interact with the one of interest. For example, Zewen Xiao et al. [19] constructed PPIN embeddings with a DGN that uses Page Rank to capture higher-order relations, with the goal of predicting missing PPI arcs. Their approach has proven effective in dealing with noisy PPINs. Similarly, Palukuri et al. [20] developed a reinforcement learning pipeline to identify protein complexes on large PPINs, while Zhang and Kabuka [21] used a bag-of-words approach to learn features from

pairs of interacting proteins, with the goal of performing protein family annotation. Concurrently, some other works attempt to fuse information coming from pathways and the PPIN topology or to enrich pathways by adding features coming from PPIs. In particular, Zhang et al. [22] merged information coming from gene-gene interactions, PPIs, gene expression, and pathway databases to identify relevant pathways connected to breast cancer. Similarly, Zhang et al. [13] used PPIs and KEGG pathways to identify relevant genes in breast cancer by performing random walk-based network reconstruction. Besides tackling learning problems different from the one presented in this work, all the above works consider the PPIN in a static setting. In contrast, our goal is to build a model that is able to predict dynamic information exploiting the PPIN structure.

Temporal data and PPINs

While PPI repositories concern static graphs, it is well-established that the interactome is not static. Interaction networks are useful for analyzing the structure of systems or the results of their perturbation, but their lack of mechanistic insights and their static nature make them unsuitable for representing dynamic systems [23, 24]. The goal of this study, namely to learn the "dynamics function" of a biological system for which an exact mathematical model cannot be constructed, has been pursued by other works, although with methodologies that can be considered transversal and not directly comparable to ours. For example, Zhang et al. [25] built dynamic graphs from temporal gene expression data of yeast and performed protein complex identification. Similarly, Jing et al. [26] used a dynamic DGN to predict the representation of -omics from reverse phase protein array (RPPA) proteomics and genome-wide expression as features, using a PPIN derived from the STRING database. Costello and Martin [27] used machine learning approaches to learn the function underlying the dynamics of metabolite and protein concentration from metabolic time series, avoiding system modeling in favor of learning the dynamics directly from data. Chow et al. [28] faced the problem of inferring missing time points in dynamic biological networks via network alignment. Furthermore, Cinaglia [29] developed a method based on DGN to evaluate node similarities in multilayer dynamic networks, which supports comparative analysis between different structures of biological networks.

The discussed methodologies build a temporal graph relying on gene expression data, which however tend to be noisy and with changes that are not significant at the PPI level. Additionally, they are bound to introduce additional noise by lacking information on post-transcription that interleaves the gene level with the PPI level. In contrast, we propose to use a different data source (biochemical pathways) to extract dynamical information, and train a model able to predict the dynamical behavior of a PPIN without the need to build a temporal graph. Our findings can also be seen as a method to enrich PPINs and complementary to the use of temporal gene expression data, as we will discuss in the delineation of future research in Sect. 5.

Background

As for any other biological network, PPINs exhibit non-trivial topological features, that have to be taken into account when designing algorithms to process them. In general, biological networks are scale-free, have a high clustering coefficient, may present communities, and have a short diameter despite usually having a large number of nodes. Some of them exhibit a hierarchical structure [30–32]. These characteristics generally make these networks robust to incompleteness, i.e., even in the presence of missing relations, it is possible to recover relevant information from the incomplete network. Both Barabási et al. [33] and Santolini and Barabási [32] pointed out that the complex structure of biological networks allows the study of dynamical properties from static graphs since the dynamics of the system arise from structural patterns. This finding has been verified at a lower level of abstraction by applying network propagation to BPs extracted from the Biomodels repository [8].

PPINs, on the other hand, can be seen as a high-level abstraction of a complex dynamical system, whose functioning can be only partially defined by mathematical rules. Exact analysis of the dynamics of biological systems in this sense is only possible from a handful of well-known BPs, which are studied as isolated systems.

Dynamical properties of BPs and sensitivity analysis

BPs are a series of interconnected biochemical reactions or molecular interactions that occur within a cell or organism to perform a specific biological function. At their finest granularity, these models can describe the kinetics of chemical reactions, allowing the simulation of their dynamics by computing the changes in concentrations at small time intervals. Simulations can be performed by reconstructing Ordinary Differential Equations (ODEs) from the reactions and through any algorithm for ODEs simulation, or via stochastic approaches like the Gillespie algorithm [9]. The biggest repository of computational models apt to simulations is the BioModels database [8]. It contains models of molecular interactions, cellular processes, and whole-organism systems. These models are represented in a variety of formats, including the System Biology Markup Language (SBML) [34], CellML[35], and Synthetic Biology Open Language (SBOL) [36].

In Bove et al. [37], Podda et al. [38, 39], Fontanesi et al. [40], the authors show that it is possible to infer some dynamical properties of BPs by training a DGN to predict a binary label representing whether the property holds or not. In these studies, BPs are represented as Petri Nets [41] where places represent concentrations of molecular species and transitions stand for reactions. To compute the binary labels that serve as ground truth for the training, they simulate the ODE system up to the steady state to assess the robustness (specifically, α-robustness [42]), sensitivity, and monotonicity properties for multiple pairs of biochemical species in the BP. Starting from these findings, in this work we investigate the possibility of inferring a specific dynamical property, and precisely the dynamical property of sensitivity, by leveraging the topology of PPINs rather than that of BPs. Crucially, PPINs describe interactions at the protein level rather than at the molecular species level; therefore, having such an inference mechanism would allow inferring indirect influences (i.e., among proteins having a path of length > 1 between them).

Sensitivity, among other sensitivity analysis methods [43], determines how variations in input factors influence a model output. In particular, the Morris method [44] is an efficient global sensitivity analysis technique that allows for the computation of global sensitivity indices by performing a limited number of model evaluations. This property enables us to analyze if a single species' initial concentration affects the output species' steady-state concentration by performing a reduced number of time-consuming model simulations. However, due to its computational efficiency, the Morris method is mainly used for screening purposes, as it yields sensitivity indices that do not have a direct interpretation in terms of output variance decomposition [45].

The Morris method is based on the concept of Elementary Effect (EE), a measure that quantifies the influence exerted by a perturbation of an input factor s_i over the output of a function f. Formally, given a mathematical model with $k \in \mathbb{N}$ input factors β_k of the form $f(\beta_1, \ldots, \beta_k)$, the EE measuring the influence of the input factor β_i exerted on the output of the function is defined as:

$$EE_i = \frac{f(\beta_1, \ldots, \beta_i + \Delta, \ldots, \beta_k) - f(\beta_1, \ldots, \beta_i, \ldots, \beta_k)}{\Delta} \tag{1}$$

where Δ is a perturbation applied to a single input factor β_i. In the context of BPs, the output $o = f(\beta_1, \ldots, \beta_k)$ is the steady-state concentration of a species, while the inputs are the initial concentration values of the remaining species. To obtain a sensitivity index with the Morris method, several EEs are computed with respect to different points in the input space; these points, along with an appropriate Δ, are chosen through sampling strategies. In this work, we adopt a radial sampling strategy derived from a Sobol sequence [46], as proposed by Campolongo et al. [45].

Finally, the mean μ of the distribution of the absolute values of the EEs and the variance σ^2 of the distribution of the EEs are computed. The mean μ is interpreted as the overall influence that β_i has on o, while the variance σ^2 estimates the effect that β_i has on o due to interaction with other inputs. Both μ and σ^2 are compared to a threshold; if either exceeds the threshold, the output is declared sensitive to that input factor. A sensitivity of 1 indicates the output is highly influenced by the input species, while a sensitivity of 0 indicates resistance to changes in that input.

Thus, sensitivity allows us to assess the importance of each parameter within the model domain and understand how changes in input concentrations influence the system's behavior. Being a global property that involves all the species concentrations in its computation, sensitivity suits our need to use information computed at the BP level to perform predictions over a PPIN. A single protein of the PPIN can be a component of multiple species in a BP because proteins bind to other molecules creating complexes formalized as distinguished entities in the BP. As described in Sect. 2.1.2, we define a dynamical property over a pair of proteins considering all the values of their complexes. Therefore, we choose a global property, that looks at all the species in the BP and not strictly to the input and output ones.

Deep graph networks for graph classification

A directed graph is a tuple

$$\mathcal{G} = \langle \mathscr{V}_{\mathcal{G}}, \mathscr{E}_{\mathcal{G}} \rangle,$$

where $\mathscr{V}_{\mathcal{G}}$ is the set of nodes and $\mathscr{E}_{\mathcal{G}} \subseteq \mathscr{V}_{\mathcal{G}} \times \mathscr{V}_{\mathcal{G}}$ is the set of directed edges. The node-wise connectivity of a graph can be expressed with a function:

$$\overrightarrow{\mathcal{N}_{\mathcal{G}}} : \mathscr{V}_{\mathcal{G}} \rightarrow 2^{\mathscr{V}_{\mathcal{G}}} : -\{u \mid (u, v) \in \mathscr{E}_{\mathcal{G}}\}.$$

The set of reversed edges of $\mathcal{G}$ is defined as $\bar{\mathscr{E}}_{\mathcal{G}} = \{(v, u) \mid (u, v) \in \mathscr{E}_{\mathcal{G}}\}$. Similarly, the function:

$$\overleftarrow{\mathcal{N}_{\mathcal{G}}} : \mathscr{V}_{\mathcal{G}} \rightarrow 2^{\mathscr{V}_{\mathcal{G}}} : -\{u \mid (v, u) \in \bar{\mathscr{E}}_{\mathcal{G}}\}$$

specifies the graph connectivity with all edge directions reversed. A subgraph $\mathcal{H}_{[\mathcal{G}]} = \langle \mathscr{V}_{\mathcal{H}}, \mathscr{E}_{\mathcal{H}} \rangle$ of $\mathcal{G}$ is a graph such that $\mathscr{V}_{\mathcal{H}} \subseteq \mathscr{V}_{\mathcal{G}}$ and $\mathscr{E}_{\mathcal{H}} \subseteq \mathscr{E}_{\mathcal{G}}$. The graphs used in this work are attributed, meaning that each graph node $v \in \mathscr{V}_{\mathcal{G}}$ is associated with a vector $\boldsymbol{x}_v \in \mathbb{R}^d$ of node features, with $d \in \mathbb{N}$. We indicate with $\mathscr{X}_{\mathcal{G}}$ the set of all node feature vectors of $\mathcal{G}$. We oftentimes use the term *skeleton* to refer to the structure of a graph without node features, i.e., considering only the sets $\mathscr{V}$ and $\mathscr{E}$.

Although graphs are very flexible and expressive data structures, they require careful handling to be used as inputs to machine learning models, since they can have varying sizes (i.e., different number of nodes) and complex connectivity patterns. DGNs learn from graph data by building a vectorial representation for each node of the graph called *embedding*, which can be used to tackle classification and regression tasks on graphs. In this work, we focus on the *graph classification* task, which can be defined informally as learning a mapping from graphs to discrete labels. Typically, DGNs for graph classification are composed of three main modules: message passing, pooling, and readout.

The *message passing module* takes an input graph and maps it to an isomorphic graph where each node is associated with an embedding. This mapping is achieved iteratively: at each iteration, the node embedding is updated as a function of itself and its neighboring embeddings. As the number of iterations increases, the receptive field of the nodes (i.e., the portion of other graph nodes that contribute to the embedding computation) grows, allowing the capture of global information from the graph [47]. Here, we focus on *convolutional* variants of message passing, which structure the iterative mapping as a sequence of neural layers (see, e.g., Bacciu et al. [10], Ye et al. [48] and the references therein).

The *pooling module* takes as input the node embeddings computed with message passing and aggregates them into a single vector representing the entire graph. This step ensures that each graph is encoded as a fixed-size embedding, regardless of its dimension and connectivity. Generally speaking, pooling can be performed using any permutation-invariant function operating on multi-sets of embeddings.

The *readout module* takes as input the pooled graph representation and outputs a class prediction. In general, any standard machine learning classifier can be used; usual choices are logistic regression or multilayer perceptron (MLP) classifiers.

In convolutional DGN architectures, these three modules are typically differentiable, allowing for gradient-based learning with backpropagation.

Methods

Before proceeding, let us restate the goal of this study for clarity. Our aim is to build a machine learning model able to predict the dynamical property of sensitivity from PPIN subgraphs. To this end, Sect. 2.1 describes how we created a dataset to train the proposed model. This process involved performing simulations on a set of BPs and computing the sensitivity among pairs of molecular species (Sect. 2.1.1), mapping the sensitivity information to the PPIN (Sect. 2.1.2), and augmenting the PPIN subgraphs with node features (Sect. 2.1.3). Then, Sect. 2.2 describes the architecture of the proposed machine learning model, including how it is used during training and inference. Finally, Sect. 2.3 describes the setup of the experiments we performed with the proposed DGN.

Dataset creation

To the best of our knowledge, there currently is no public dataset about dynamical properties prediction over PPINs. This section explains how we obtained one for our purposes. As stated in Sect. 1.2.1 we focus on sensitivity, although the method we propose is general and can be applied seamlessly to other dynamical properties. To help the reader understand the workflow, we show in Fig. 2 the process to obtain a training data sample starting from a single BP.

BP simulations and sensitivity computation

We downloaded all 1063 manually curated models in SBML format [34] from the Biomodels repository [8]. Each model is represented as a set of reactions, as exemplified in Fig. 2, box (1). We then converted the reactions set into a system of ODEs as shown in Fig. 2, box (2), and simulated them up to steady state using the `libroadrunner` library [49], which applies a numerical integration method to the ODE system. The simulation schema follows the one presented in [37] and [40]. We did not simulate BPs composed of assignment rules exclusively, and/or delayed differential equations, as they are not supported by the library. Furthermore, we discarded simulations that failed completely, due to numerical instability introduced by variations at too different scales, or partially, whenever we could not obtain at least 10 EEs to compute the sensitivity.

After running the simulations, we obtained a set $\mathbb{S}$ of 842 successfully simulated BPs. Formally, a BP obtained after the simulations is completely specified by the triplet $\mathscr{S} = \langle \mathcal{S}_{\mathscr{S}}, \mathcal{P}_{\mathscr{S}}, \phi_{\mathscr{S}} \rangle$ where:

- $\mathcal{S}_{\mathscr{S}}$ is the set of molecular species interacting in the pathway. We use the letter s to indicate elements of $\mathcal{S}_{\mathscr{S}}$, which can be single proteins or protein complexes;
- $\mathcal{P}_{\mathscr{S}} \subseteq \mathcal{S}_{\mathscr{S}} \times \mathcal{S}_{\mathscr{S}}$ is the set of input/output molecular pairs on which the sensitivity is calculated. We use the notation (s_{in}, s_{out}) to denote elements of $\mathcal{P}$;
- $\phi_{\mathscr{S}} : \mathcal{P}_{\mathscr{S}} \rightarrow \{0, 1\}$ is a function that computes the sensitivity for a given input/output pair using the Morris method (see Sect. 1.2.1). More specifically, $\phi_{\mathscr{S}}(s_{in}, s_{out}) = 1$ if s_{out} is sensitive to s_{in}, and 0 otherwise.

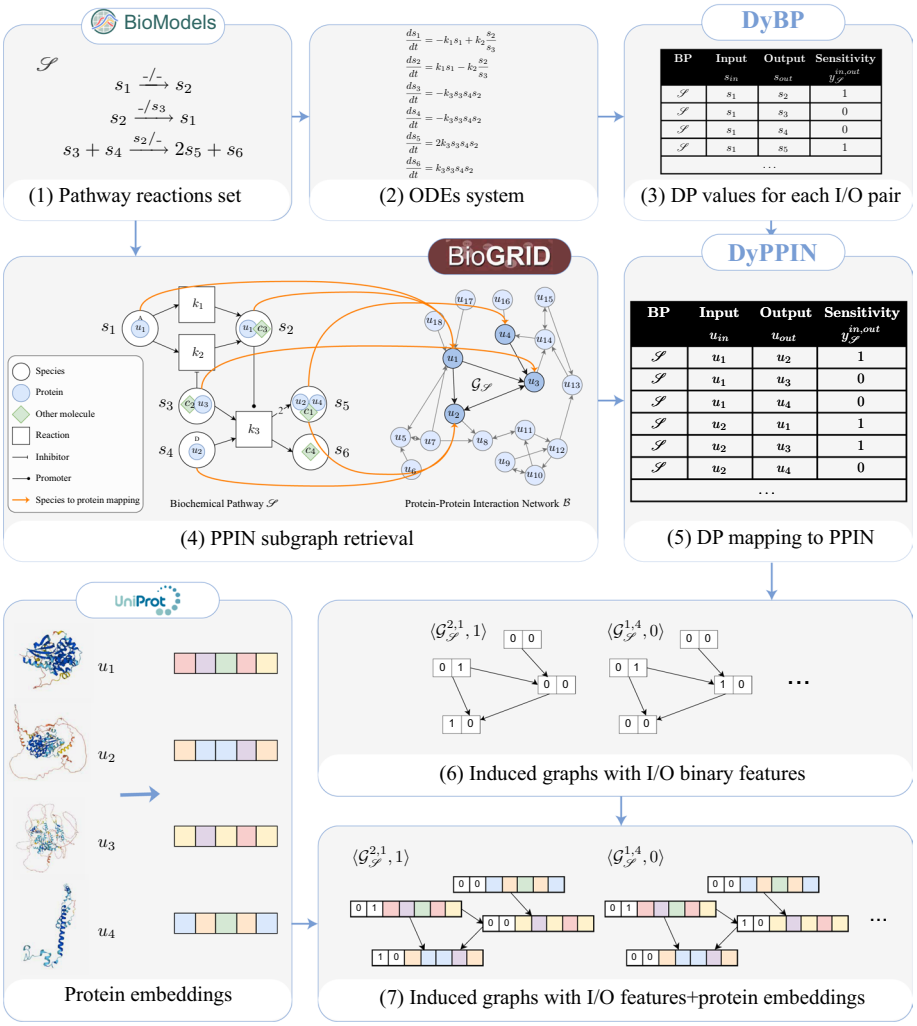


Fig. 2 Dataset extraction process for a single BP $\mathscr{S}$. (1) The BP's model is downloaded from the Biomodels repository. (2) The BP is converted to ODEs and simulated to steady state multiple times. (3) The sensitivity is computed from the simulations' results for each possible input/output pair in the BP. (4) The BP-related protein interaction graph $\mathcal{G}_{\mathscr{S}}$ is built by retrieving the BioGRID interactions among the proteins in the BP; note that multiple species can be mapped to the same protein (orange arrows), e.g. s_1, s_2 to u_1, and a single species can be mapped to multiple proteins, e.g. s_5 to u_2 and u_4. (5) The DyPPIN dataset consists of the DPs for each I/O species pair that are mapped to each protein-protein pair. (6) Each DyPPIN data sample induces a graph $\mathcal{G}_{\mathscr{S}}^{in,out}$ for the DGN training: the skeleton conveys the PPIN subgraph topology, while the node features $\mathscr{X}_{\mathscr{S}}^{in,out}$ represent whether the node is the input u_{in} or the output u_{out}. Optionally, the node features can be augmented with protein embeddings from UniProt (7). The graphs $\mathcal{G}_{\mathscr{S}}^{in,out}$ and their sensitivity labels $y_{\mathscr{S}}^{in,out}$ are used as input and the target variables of our graph classifier

We assembled this information into a dataset which we termed **DyBP** (Dynamics of Biochemical Pathways), defined as follows:

$$\mathbf{DyBP} = \bigcup_{\mathscr{S} \in \mathbb{S}} \{\langle s_{in}, s_{out}, \phi_{\mathscr{S}}(s_{in}, s_{out})\rangle \mid (s_{in}, s_{out}) \in \mathcal{P}_{\mathscr{S}}\}.$$

A portion of the **DyBP** dataset in tabular form is shown in Fig. 2, box (3).

Mapping of BP information onto the BioGRID PPIN

The next step entailed mapping the sensitivity information contained in the **DyBP** dataset to the interactome. This mapping is not trivial, since species in a BP can be complexes containing multiple proteins, and the same protein can be a part of multiple complexes. After taking into consideration different PPI databases (IntACT, STRING, BioGRID), we chose BioGRID as the reference interactome since it contains the largest number of curated physical interactions involving BP proteins. We consider all the PPIs in BioGRID independently from the organism they refer to.

Let $\mathcal{B} = \langle \mathcal{V}_{\mathcal{B}}, \mathcal{E}_{\mathcal{B}} \rangle$ be the graph representing the BioGRID interactome, where nodes $u \in \mathcal{V}_{\mathcal{B}}$ are proteins and edges $\mathcal{E}_{\mathcal{B}} \subseteq \mathcal{V}_{\mathcal{B}} \times \mathcal{V}_{\mathcal{B}}$ are PPIs. The connection between BPs and PPINs stems from the fact that BP species $s \in \mathcal{S}$ are essentially individual proteins or protein complexes that we represent as sets of proteins, some of which belong to the interactome,[1] i.e., $s \subset \mathcal{V}_{\mathcal{B}}$. Having made this crucial observation, for each BP $\mathcal{S}$ in the **DyBP** dataset we induce a BioGRID subgraph $\mathcal{G}_{\mathcal{S}}^{in,out} = \langle \mathcal{V}_{\mathcal{S}}, \mathcal{E}_{\mathcal{S}}, \mathcal{X}_{\mathcal{S}}^{in,out} \rangle$ where:

- The set of nodes is composed of all the interactome proteins belonging to at least one molecular species in $\mathcal{S}_{\mathcal{S}}$. More formally, $\mathcal{V}_{\mathcal{S}} = \{u \in \mathcal{V}_{\mathcal{B}} \mid \exists s \in \mathcal{S}_{\mathcal{S}} : u \in s\}$.

  As a further restriction, we filtered out proteins that are not associated with a UniPROT identifier,[2]
- The set of edges is composed of all the PPIs in the BioGRID graph having proteins in $\mathcal{V}_{\mathcal{S}}$ as vertices,
- $\mathcal{X}_{\mathcal{S}}^{in,out}$ is the node features set that is constructed depending on the choice of an input protein u_{in} and an output protein u_{out} (see Sect. 2.1.3).

We remark that even though the notation does not explicitly specify it, $\mathcal{G}_{\mathcal{S}}$ is to be considered as a proper subgraph of the BioGRID PPIN, in the sense that $\mathcal{V}_{\mathcal{S}} \subseteq \mathcal{V}_{\mathcal{B}}$ and $\mathcal{E}_{\mathcal{S}} \subseteq \mathcal{E}_{\mathcal{B}}$. An example of mapping a BPs to the corresponding BioGRID subgraphs is shown in Fig. 2, box (4). In practical applications, this mapping could produce a disconnected BioGRID subgraph, as there is no guarantee that nodes in $\mathcal{V}_{\mathcal{S}}$ all share a direct interaction. To avoid this issue, we complete $\mathcal{V}_{\mathcal{S}}$ and $\mathcal{E}_{\mathcal{S}}$ by adding all the proteins and interactions in $\mathcal{B}$ that belong to a path of minimal length between any pair of connected components to restore full connectivity. Lastly, we associate the sensitivity information for any two pairs of proteins in the subgraph with the following rule:

$$y_{\mathcal{S}}^{in,out} = \begin{cases} 1 \text{ if } \exists\, (s_i, s_j) \in \mathcal{P}_{\mathcal{S}}, u_{in} \in s_i, u_{out} \in s_j : \phi_{\mathcal{S}}(s_i, s_j) = 1, \\ 0 \text{ otherwise.} \end{cases}$$

In other words, if at least one species including protein u_{in} influences at least another species including protein u_{out} at the BP level we claim the same influence at the PPIN level.

[1] Technically, a species can include other molecules, but we do not consider them since we are interested in proteins only.

[2] UniPROT was chosen since it is the most widely used ontology for proteins in PPI databases.

During the mapping process, several BPs were mapped into trivial BioGRID subgraphs containing a single node or a completely disconnected set of nodes. This was caused by different reasons, such as lack of corresponding PPIs or due to protein identifiers missing in the SBML model. To prevent adding trivial subgraphs to the **DyPPIN** dataset, we discarded all BPs with trivial structure as well as those who mapped to trivial BioGRID subgraphs; as a consequence, we were able to extract BioGRID subgraphs from 279 BPs out of the original 842.

Having obtained the subgraphs and their respective sensitivity indicators, we assembled this information to create a dataset, which we termed **DyPPIN** (Dynamics of Protein-Protein Interaction Networks), as follows:

$$\textbf{DyPPIN} = \{\langle \mathcal{G}_{\mathscr{S}}^{in,out}, y_{\mathscr{S}}^{in,out} \rangle \mid (u_{in}, u_{out}) \in (\mathscr{V}_{\mathscr{S}} \times \mathscr{V}_{\mathscr{S}})\}.$$

A subset of the **DyPPIN** dataset in tabular form is displayed in Fig. 2, box (5). Ultimately, the **DyPPIN** dataset contains 17169 training pairs.

Adding node features to subgraphs

Before being processed by the DGN, each BioGRID subgraph in the **DyPPIN** dataset is augmented by adding a vector of node features. More precisely, given a subgraph $\mathcal{G}_{\mathscr{S}}^{in,out}$ as defined in Sect. 2.1.2, we associate each node v to a vector $\boldsymbol{x}_v \in \mathbb{R}^d = [I; O; \tilde{\boldsymbol{x}}_v]$, with [;] indicating concatenation, where:

$$I = \begin{cases} 1 \text{ if } v = u_{in} \\ 0 \text{ otherwise,} \end{cases} \quad O = \begin{cases} 1 \text{ if } v = u_{out} \\ 0 \text{ otherwise.} \end{cases}$$

Basically, the first two components of $\boldsymbol{x}_v$ indicate whether the current node is an input protein, an output protein, or neither. We call these two components the *I/O features*. The remaining component $\tilde{\boldsymbol{x}}_v$ is not fixed, but varies depending on the experiments. Specifically, it can be:

- *Empty*, to let the DGN learn from the structure alone and assess the binding between graph structure and sensitivity;
- a *Protein sequence embedding* predicted by the protein language model ProtT5, as downloaded from UniProt [17]. These embeddings are produced by tokenizing protein sequences and applying positional encoding. The sequences are then passed through a transformer model to generate context-aware embeddings from the last hidden state of the transformer's attention stack. We compress the embeddings via principal component analysis from the original dimension of 1024 to 128. The optimal number of components has been selected by taking the kneading point of the explained variance ratio curve [50];
- a *One-hot encoding* of the possible protein identifiers in **DyPPIN**, (1029 in total). These features will serve as "protein information baseline": as they are just an identifier, they are useful to check whether we actually need the rich information stored in the protein embedding.

Finally, we arrange all the node feature vectors into a set $\mathscr{X}_{\mathscr{S}}^{in,out}$ to be used by the DGN during training or inference. An example of BioGRID subgraph encoded with the

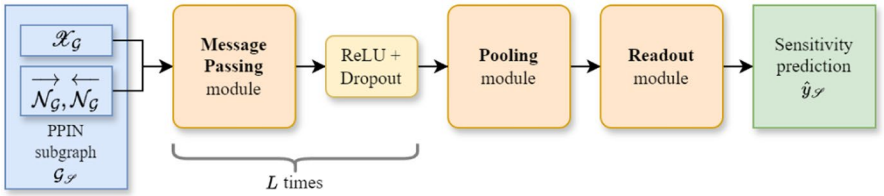


Fig. 3 The proposed DGN architecture for sensitivity prediction. Given a PPIN subgraph $\mathcal{G}_{\mathscr{S}}$, its node features $\mathscr{X}_{\mathcal{G}}$ and connectivity functions $\overrightarrow{\mathcal{N}_{\mathcal{G}}}$ and $\overleftarrow{\mathcal{N}_{\mathcal{G}}}$ are provided as input to the message passing module, which computes the node embeddings by applying L graph convolutional layers to the node features. After each convolutional layer, the embeddings are passed through a ReLU non-linearity and a dropout layer. Then, the final embeddings are aggregated into a single graph representation via the pooling module. Lastly, the readout module takes in the aggregated graph representation and computes a sensitivity prediction $\hat{y}_{\mathscr{S}}$

corresponding node features (with empty $\tilde{\boldsymbol{x}}_v$) is shown in Fig. 2, box (6). Similarly, Fig. 2, box (7) shows the same graph but with $\tilde{\boldsymbol{x}}_v$ containing ProtT5 embeddings taken from UniProt.

DGN architecture

In this section, we present the details of the DGN architecture used to learn sensitivity from PPINs. We refer the reader back to Sect. 1.2.2 for a general understanding of the main DGN concepts. At a high level, the DGN receives and processes a **DyPPIN** BioGRID subgraph $\mathcal{G}_{\mathscr{S}}$, encoded as described in Sect. 2.1.3, to produce a sensitivity prediction $\hat{y}_{\mathscr{S}} \in (0, 1)$ as output. Notice that we drop the *in*, *out* superscript for conciseness. The overall architecture is shown in Fig. 3; below, we describe its main modules (message passing, pooling, and readout) in detail.

Message passing module The message passing module takes as input a graph $\mathcal{G}$, and specifically its node features $\mathscr{X}_{\mathcal{G}}$ and the connectivity functions $\overrightarrow{\mathcal{N}_{\mathcal{G}}}$ and $\overleftarrow{\mathcal{N}_{\mathcal{G}}}$, giving as output a graph isomorphic to $\mathcal{G}$ where each node v is associated to an embedding $\boldsymbol{h}_v^L \in \mathbb{R}^h$, where $h \in \mathbb{N}$ is the embedding dimension. In practice, the message passing module maps the input to the output with a stack of L subsequent graph convolutional layers. The computation is initialized with $\boldsymbol{h}_v^0 = \boldsymbol{x}_v$, i.e., by setting the node features as initial embeddings. Then, each intermediate graph convolutional layer processes its input as follows (for $\ell \geq 1$):

$$\overrightarrow{\boldsymbol{h}}_v^{\ell} = \overrightarrow{\boldsymbol{W}}_1 \boldsymbol{h}_v^{\ell-1} + \overrightarrow{\boldsymbol{W}}_2 \bigoplus_{v' \in \overrightarrow{\mathcal{N}_{\mathcal{G}}}(v)} \boldsymbol{h}_{v'}^{\ell-1}, \tag{2}$$

$$\overleftarrow{\boldsymbol{h}}_v^{\ell} = \overleftarrow{\boldsymbol{W}}_1 \boldsymbol{h}_v^{\ell-1} + \overleftarrow{\boldsymbol{W}}_2 \bigoplus_{v' \in \overleftarrow{\mathcal{N}_{\mathcal{G}}}(v)} \boldsymbol{h}_{v'}^{\ell-1}, \tag{3}$$

$$\boldsymbol{h}_v^{\ell} = \alpha\, \overrightarrow{\boldsymbol{h}}_v^{\ell} + (1 - \alpha)\, \overleftarrow{\boldsymbol{h}}_v^{\ell}. \tag{4}$$

In Eq. 2, the current node embedding $\boldsymbol{h}_v^{\ell-1}$ is updated as a function of itself and a permutation-invariant aggregation (denoted with $\bigoplus$) of its *incoming* neighbors, as selected through $\overrightarrow{\mathcal{N}_{\mathcal{G}}}$. This computation is parameterized by learnable weights $\overrightarrow{\boldsymbol{W}}_1, \overrightarrow{\boldsymbol{W}}_2 \in \mathbb{R}^{h \times h}$. Analogously, Eq. 3 updates the current node embedding as a function of itself and its *outgoing* neighbors, as selected through $\overleftarrow{\mathcal{N}_{\mathcal{G}}}$. This computation is parameterized by a different set of learnable weights $\overleftarrow{\boldsymbol{W}}_1, \overleftarrow{\boldsymbol{W}}_2 \in \mathbb{R}^{h \times h}$. Finally, in Eq. 4, the two intermediate

embeddings are combined into a final embedding as a convex combination with coefficient α. This implementation combines ideas from previous contributions in the field of DGNs [51, 52] which are particularly suited to our task. Specifically, learning the contribution of incoming and outgoing neighbors with separate sets of weights reflects the fact that in a PPIN, a node can play the role of source (resp. target) of the interaction. Thus, having distinct sets of weights creates two distinct flows of information depending on the role taken by each node, where the "strength" of each flow is modulated by α. Also, remark that the weights are adjusted by the network during the learning phase in order to better approximate the relationship between the input PPINs and the output sensitivity.

After L graph convolutional layers, the resulting embeddings $\boldsymbol{h}_v^L$ are further passed into an element-wise ReLU non-linearity and a final Dropout layer [53]. Dropout is a model regularization technique that deactivates neurons with a certain probability during training, encouraging the neural network to learn more robust features.

The message passing module includes several hyperparameters that have been tuned during model selection: the number of graph convolutional layers L, the embedding dimension h, and the convex combination coefficient α. We refer to Sect. 2.3.2 for an in-depth discussion.

Pooling module The pooling module takes as input the node embeddings at the last message passing layer and aggregates them into a single vector representing the entire graph. This step ensures that each graph is encoded as a fixed-size embedding, regardless of its dimension and connectivity. In this work, we only consider add pooling, which aggregates using the sum. Therefore, the pooling module produces a graph representation $\boldsymbol{h}_g \in \mathbb{R}^h$ by aggregating the node embeddings at layer L as follows:

$$\boldsymbol{h}_{\mathcal{G}} = \sum_{v \in \mathcal{V}_{\mathcal{G}}} \boldsymbol{h}_v^L. \tag{5}$$

Readout module The readout module takes as input the pooled graph representation and outputs a sensitivity prediction $\hat{y}_{\mathcal{G}} \in (0, 1)$. In this work, the readout module is a simple logistic regression classifier:

$$\hat{y}_{\mathcal{G}} = \text{sigm}(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{h}_{\mathcal{G}} + \text{b}), \tag{6}$$

where $\boldsymbol{w} \in \mathbb{R}^h$ and $b \in \mathbb{R}$ are learnable weights, and sigm is the sigmoid function.

Training and inference

During training, the DGN receives mini-batches $B \subset$ **DyPPIN** of training samples, where each training sample is a tuple $\langle \mathcal{G}_{\mathcal{G}}, y_{\mathcal{G}} \rangle$. The graph $\mathcal{G}_{\mathcal{G}}$ is processed by the DGN to obtain a prediction $\hat{y}_{\mathcal{G}}$, which is compared to the true sensitivity $y_{\mathcal{G}}$ through a binary cross entropy function as follows:

$$\mathcal{L}(y_{\mathcal{G}}, \hat{y}_{\mathcal{G}}) = \frac{1}{|B|} \sum_{\langle \mathcal{G}_{\mathcal{G}}, y_{\mathcal{G}} \rangle \in B} y_{\mathcal{G}} \log(\hat{y}_{\mathcal{G}}) + (1 - y_{\mathcal{G}}) \log(1 - \hat{y}_{\mathcal{G}}). \tag{7}$$

The DGN parameters are then adjusted to reduce the average loss of the batch using stochastic gradient descent.

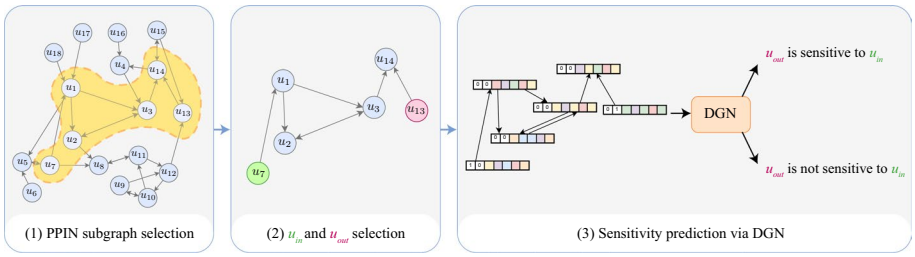


Fig. 4 An example of how sensitivity is predicted on a trained DGN. The user has to (1) select the subgraph containing the proteins of interest, here marked in yellow, (2) the input protein u_{in} to perturb, and the output protein u_{out} on which the sensitivity needs to be assessed. (3) The DGN model processes the induced graph and predicts whether u_{out} is sensitive to u_{in}

Once the DGN has been trained, it can be used for inference, i.e., to predict BioGRID subgraphs that were not seen during training, as shown in Fig. 4. In this case, a user only needs to select a portion of interest in the BioGRID interactome, as shown in box (1); choose an input and output protein whose sensitivity needs to be predicted, as depicted in box (2); and finally feed the corresponding subgraph $\mathcal{G}_{\mathcal{S}}$ to the DGN, as shown in box (3). We provide a command line tool that implements this inference pipeline in the attached code repository.

Experimental design and performance evaluation

When dealing with any prediction task over groups of proteins, it is crucial to properly split the available data to avoid unrealistic performance estimations [54]. As proteins form a densely connected graph due the presence of hubs, splitting nodes and edges to form disjoint sets is infeasible, especially for a dataset covering a small part of the interactome like ours (see Sect. 3.1). Rather, we can quantify the expected data leakage and avoid overestimating graph models performances by designing proper validation strategies [55, 56]. We defined different data splitting strategies to evaluate performances in three different use cases (UCs). The UCs reflect how the test graphs overlap with those in the training set.

UC1 **Unknown input/output pair**. In this case, the data samples of **DyPPIN** were split with a standard random sampling. Therefore, the same PPIN subgraph can be present both in the training and test sets but with different input/output pairs. This corresponds to the scenario where the user wants to predict the sensitivities of new input/output pairs within a group of proteins that could be in BPs in **DyBP**.

UC2 **Unknown protein**. In this case, we performed a k-fold split over the proteins in **DyPPIN**, inducing a split over the data samples. Basically, we ensure that the held out subset of proteins in the test set does not appear in any training pair (either as input or output). This corresponds to the scenario where the user wants to predict the sensitivity for proteins that are not in **DyPPIN**. Notice that this setup is more challenging than UC1 since a model is trained without all the sensitivity information about some proteins.

UC3 **Unknown subgraph**. In this case, we performed a k-fold split over $\mathcal{S}$, inducing a split of the data samples with respect to their skeletons. This corresponds

to the scenario where the user wants to assess the sensitivity within a group of proteins that induce a totally new PPIN subgraph. This is the most challenging scenario, because the model needs to generalize over unobserved topologies containing mostly unobserved nodes.

We remark that the inference tool allows the user to evaluate in which UC a prediction falls. This is done by computing the overlap between the imputed proteins and the proteins in **DyPPIN** on which the DGN model has been trained. Notice that UC2 and UC3 can be seen as information ablation tests with respect to the base case UC1.

Performance metrics

Performances have been assessed using accuracy, F1-score, Area Under the Receiving Operator Curve (AUROC), and Matthews Correlation Coefficient (MCC). In particular, the accuracy defined as:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}},$$

where TP, TN, FP, FN stand for true positives, true negatives, false positives, and false negatives, respectively. Accuracy was chosen since it is a standard metric for classification tasks. However, accuracy is less interpretable when class imbalance is at play (our case). Therefore, we computed additional metrics that are more informative under class imbalance. Specifically, the F1-score is computed as:

$$\mathrm{F1} = \frac{2\,\mathrm{TP}}{2\,\mathrm{TP} + \mathrm{FP} + \mathrm{FN}},$$

which corresponds to the geometric mean between precision (i.e., positive predictive value), and recall (i.e., sensitivity). F1-score is more advantageous since it is imbalance-aware (it is 0 in case of a null classifier that only predicts the majority class). The AUROC measures the probability that the model ranks a randomly chosen sensitive example higher than a randomly chosen non-sensitive example [57]. AUROC ranges from 0 to 1, and a null classifier obtains an AUROC of 0.5. However, AUROC can provide a deceptively high score when the classifier is biased towards the majority class. Lastly, we computed the MCC, defined as:

$$\mathrm{MCC} = \frac{\mathrm{TP}\,\mathrm{TN} - \mathrm{FP}\,\mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}}.$$

Essentially, the MCC measures how much the predictions correlate with the true sensitivity labels, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation). An MCC score of 0 indicates that the predictions are not correlated with the targets. It is worth noting that MCC is imbalance-agnostic and symmetric (i.e., it treats the positive and negative class equally) [58, 59].

Model selection and evaluation

We evaluate the model through a grouped 4-fold cross-validation, stratified by the sample labels $y_{\mathscr{G}}$. Specifically, we split the **DyPPIN** dataset into four subsets (or

folds). In turn, one of the four folds is held out for testing, while the remaining three folds are further partitioned into a training set (used to fit the model parameters) and a validation set (used to tune the model hyperparameters). We use grouping for UC2 (resp. UC3) to build the folds so that all data samples for a protein (resp. BP) do appear either in the training, validation, or test set, hence preventing information leakage and ensuring that the model performances are properly evaluated.

Since the number of samples per BP is highly variable across the **DyPPIN** dataset (a characteristic further analyzed in Sect. 3.1), we carefully selected the optimal number of groups in order to have a coherent number of samples for training, validation, and test set across the different folds. The groups are also stratified with respect to the sensitivity labels to homogenize the distribution of the classes across the folds.

The best DGN hyperparameters of the DGN were chosen using grid search. Specifically, we first defined a grid of possible values for each hyperparameter, and then we exhaustively evaluated all combinations of from the grid on the validation set, choosing the set of hyperparameters which scored the highest F1. Tuned hyperparameters include:

- Number of layers L, chosen in the set $\{1, \dots, 8\}$;
- Embedding dimension h, chosen in the set $\{32, 64, 128, 256, 512, 1024\}$;
- Convex combination coefficient α, chosen in the set $\{1.0, 0.9, 0.5\}$;
- Learning rate, chosen in $\{1e-3, 5e-4, 1e-4, 1e-5\}$;
- L2 regularization coefficient λ, chosen in $\{1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 0\}$;
- Units dropout percentage, chosen in $\{0, 0.25, 0.5\}$;
- Neighborhood aggregation operator $\bigoplus$, chosen between the *sum* and *mean* operators.

Baseline models Since we are interested in investigating whether the PPIN structure plays a role in inferring sensitivity, we compared the proposed architecture to a baseline where the graph connectivity is discarded. Basically, we kept the same architecture shown in Fig. 3, but we replaced the message passing module with a multilayer perceptron with one hidden layer, which is applied to the node features, similarly to the DeepSets architecture [60]. In practice, the graph embedding is computed as follows:

$$\boldsymbol{h}_G = \sum_{v \in \mathscr{V}_G} \mathrm{MLP}(\boldsymbol{x}_v^0), \tag{8}$$

and fed to the readout module, similarly to the proposed model. Basically, the role of the baseline is to check if the graph structure is needed to accomplish the task, following the current best practices for the evaluation of graph classifiers [61].

We also included a "null" baseline which always predicts the majority class, to check whether the model provides any meaningful learning beyond the trivial prediction of the most frequent class.

Furthermore, we performed grid searches using other graph convolutional architectures, GCN [62] and GIN [63], whose results are provided as supplementary material.

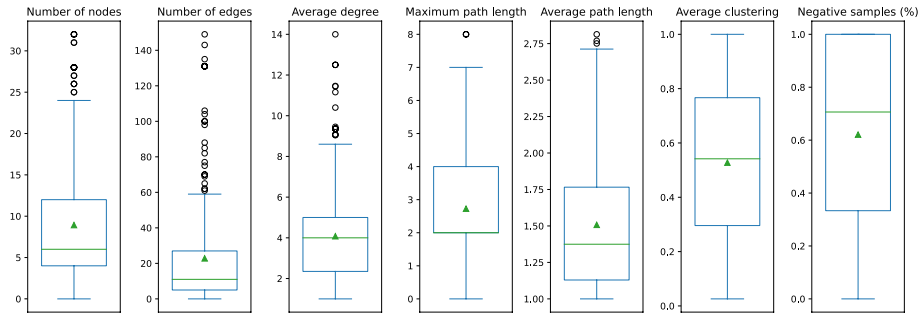
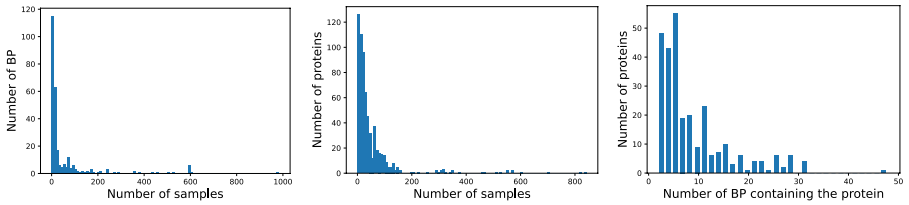


Fig. 5 Statistics about the 279 PPIN subgraphs in the dataset. The graphs are generally small, i.e. they have 9 nodes and 23 edges on average. The average and maximum path length among any two nodes in the graphs are short (~1.5 and ~3), which is an expected property of PPINs. The clustering coefficient has a mean of 0.5, but there is a not negligible part of strongly connected and loosely connected graphs

Other details about model training We use weighted batch sampling to mitigate the bias induced by the imbalance in the distribution of sensitivity indicators (see Sect. 3.1). The training was performed using the Adam optimizer [64]; we use early stopping with 100 epochs patience and tolerance 10^{-6}. The model has been implemented in PyTorch [65], PyTorch Lightning [66], and PyTorch Geometric [67] for the model development, Ray Tune [68] and Wandb [69] for the grid search management. Experiments were run on a single machine with two NVIDIA® A30 GPUs with 24 GB of dedicated memory, and two 28-core Intel® Xeon® Gold 6238R CPU @ 2.20GHz, and 250 GB of RAM. Due to early stopping, training times show a great variance depending on the UC and the node features. An epoch took 6 s on average. The final configurations took 12 to 26 min in UC3, while in UC1 and UC2 they range from 17 to 248 min. Usage of node features led to earlier stopping in all UCs (2 to 10 times less epochs), at the cost of a larger memory consumption. In terms of scalability, we remark that our method works on small PPIN subgraphs, each of which covers a small interactome portion of interest. Therefore, we reasonably expect that the training times will scale proportionally with the number of training subgraphs, thus remaining into feasible ranges. Details about training times can be found in the model selection notebook in the code repository (see the "Availability of data and materials" section).

Results and discussion

In this section, we present a comprehensive analysis of the **DyPPIN** dataset and evaluate the performance of the proposed model on various use cases. In Sect. 3.1, we analyze the data to give insights into its structure and characteristics, which is crucial for understanding the context of our experiments. In Sect. 3.2, we discuss the results obtained from different model configurations and their performance across different experimental setups. Finally, in Sect. 3.3, we perform an error analysis to identify patterns in the misclassified samples and understand the limitations of our models.



(a) Amount of BP by number of data samples. (b) Amount of proteins by number of data samples (c) Amount of BP in which a protein is present.

Fig. 6 Distribution of proteins and BP with respect to the amount of data samples. For example, the highest bar of Fig. 6(c) can be interpreted as: *"there are 40 proteins that are present in 5 different BPs"*. Note that these 5 BP can be different for each protein, so we are referring to up to 800 different BPs

Dataset analysis

The **DyPPIN** dataset features 17169 samples coming from 279 different BPs. The dataset is imbalanced towards the non-sensitivity class (i.e., $y_{\mathscr{S}} = 0$), which covers 67.6% of the sample labels. The class distribution is not uniform for each BP. Figure 5 shows a high variance of the number of nodes, edges, maximum path length, and clustering coefficient.[3] In particular, it can be noticed that most of the graphs are quite small, having a number of nodes between 4 and 10 and between 4 and 25 edges. Observing the clustering coefficient, the average degree, and the maximum path length reveals that in general, every node is connected to half of the nodes in the network. Therefore, the topological features of the graphs in **DyPPIN** are aligned with those of the typical biological networks discussed in Sect. 1.1.1. This is crucial since our desire is to learn from graphs matching the generic PPIN topology.

Since we compute the sensitivity between any pair of species in a BP, the number of data samples grows exponentially with the number of species. Therefore, BP having larger graphs will also have many more samples (see Fig. 6a), which could cause the model to overfit the larger graphs. Also, proteins belonging to larger graphs will be involved in more I/O pairs, causing further imbalance. From Fig. 6b we can observe that the number of data samples for each protein follows a power law distribution.

The **DyPPIN** dataset encompasses 1009 different proteins, covering 1.14% of all the proteins in the BioGRID PPIN. Similarly, the PPIs among them (2826) cover only 0.13% of the BioGRID PPIN. This is not surprising as we know that the set of BPs in the BioModels repository has minimal coverage. By considering the interactions of the sole proteins in the dataset, we cover 5.35% of the reactions involving them. Clearly, this implies that the set of subgraphs used in this study mostly covers well-studied BPs, which could be perceived as a bias. However, it is crucial to remark that our approach is topology-based, and that our experimental setup is designed to explicitly stress-test the model to predict unseen interactome regions (as detailed in Sect. 3.2), which may as well contain less-studied BPs. Therefore, as long as the PPIN subgraphs are extracted with the methodology detailed in Sect. 2.1, whether the BP is less-studied does not constitute a concern. Despite the small coverage, most of the proteins in the **DyPPIN** dataset (97.5%) are present in more than one BP, as shown in Fig. 6(c). This is a much-desired property

[3] A measure of how likely it is that nodes form tightly knit groups [70]

Table 1 Average test scores (± standard deviation) obtained by the different models in the random hold out use case (UC1). The suffix +emb indicates that protein sequence embeddings were concatenated to the I/O features. Best results are **boldfaced**

Model	MCC	F1	ACC	AUROC
Null	.000±.000	.000±.000	.676±.000	.500±.000
DeepSets	.062±.043	.284±.140	.574±.167	.650±.004
DeepSets+emb	.472±.006	.647±.004	.764±.004	.817±.002
DGN	.625±.009	.750±.006	.830±.009	.907±.004
DGN+emb	**.767±.004**	**.843±.002**	**.897±.003**	**.957±.002**

since it will allow us to verify whether we can learn from some BPs in which the protein interacts and infer the sensitivity of new ones.

Model performances

In this section, we review the results obtained for each use case presented in Sect. 2.3. For clarity and conciseness, we focus the discussion of the results on the MCC metric; however, similar considerations can be extended to the other reported metrics without loss of generality.

UC1: Unknown input/output pair. We remind the reader that in this use case, our aim is to test to what extent the models generalize to unseen input/output pairs (which however belong to proteins that are seen during training). The results of this experiment are reported in Table 1. We notice that the vanilla DeepSets variant predicts essentially at random, as indicated by the MCC metric which is close to 0 (indicating that its predictions are not correlated with the true sensitivities). This suggests that when both the graph structure and the protein embeddings are not used, the resulting model is unable to generalize.

When protein embeddings are added to the node features, the resulting model starts to properly generalize, as indicated by the significant 7.6x improvement in the MCC metric obtained by the DeepSets+emb variant with respect to the vanilla variant. The next leap in performance is achieved by the vanilla DGN variant, which improves the MCC metric upon the DeepSets+emb variant by a further 32% despite not using additional protein embeddings as node features. This suggests that in this task, the graph structure (i.e., the way the PPIN subgraph is "wired") has a stronger impact than the protein embeddings on performances, and that DGN are able to exploit it to improve predictions. Lastly, adding protein embeddings to the DGN nodes further improves performances, as demonstrated by the 23% MCC improvement achieved by the DGN+emb variant over the vanilla DGN variant. Overall, these results align to our initial hypothesis that the combination of the graph structure and the protein embeddings (which also encode structural and sequential patterns) provide the strongest signal to learn the sensitivity prediction task.

UC2: Unknown protein. This use case tests whether the models generalize to input/output pairs containing unseen proteins. The results of the experiments are reported in Table 2. As explained in Sect. 2.3, this is a more challenging setup than UC1, where we expected a slight decrease in performance. From the comparison, we notice that

Table 2 Average test scores (± standard deviation) obtained by the different models in the protein hold out use case (UC2). The suffix +emb indicates that protein sequence embeddings were concatenated to the I/O features. Best results are **boldfaced**

Model	MCC	F1	ACC	AUROC
Null	.000±.000	.000±.000	.680±.024	.500±.000
DeepSets	.021±.036	.218±.200	.572±.169	.560±.142
DeepSets+emb	.415±.031	.605±.038	.741±.013	.783±.033
DGN	.404±.077	.604±.043	.728±.052	.791±.038
DGN+emb	**.512±.040**	**.667±.032**	**.788±.019**	**.844±.025**

Table 3 Average test scores (± standard deviation) obtained by the different models in the pathway hold out use case (UC3). The suffix +emb indicates that protein sequence embeddings were concatenated to the I/O features. Best results are **boldfaced**

Model	MCC	F1	ACC	AUROC
Null	.000±.000	.000±.000	.678±.014	.500±.000
DeepSets	.048±.037	.259±.169	.565±.174	.634±.037
DeepSets+emb	.104±.094	.320±.186	.637±.047	.554±.087
DGN	.230±.074	**.528±.029**	.581±.098	.631±.086
DGN+emb	**.277±.058**	.515±.040	**.682±.030**	**.657±.041**

by adding protein embeddings the DeepSets+emb variant scores a significant 19.8x improvement (with respect to the vanilla DeepSets variant) in the MCC metric, indicating that protein embeddings are beneficial to generalization. As regards DGNs, we observe that the vanilla variant does not improve upon the DeepSets+emb variant. However, the DGN+emb variant, which combines protein embeddings with the graph structure, is able to improve in the MCC metric by 23% over the DeepSets+emb variant. This result indicates that, even when transitioning to a more challenging task, using the graph structure together with informative node embeddings allows the resulting model to generalize, and makes it applicable to perform predictions on proteins whose behavior cannot be studied at the BP level.

UC3: Unknown subgraph. In our last use case, we test whether the models are still able to generalize to novel PPIN subgraphs. The results of this experiment are reported in Table 3. As mentioned in Sect. 2.3, this is the most challenging scenario, which requires the models to be able to generalize to possibly unseen nodes and topologies. In this case, DeepSets variants, which do not exploit the graph structure, are unable to generalize even with protein embeddings. In fact, they achieve an MCC close to 0 or display high variance across the test folds (in particular the DeepSets+emb variant). In contrast, even in this challenging setup, the structure-aware DGN variants are able to generalize to unseen topologies to some extent: in particular, the DGN+emb (resp. DGN) improves the MCC metric by up to 2.7x (resp. 2.2x) with respect to the DeepSets+emb variant.

Several insights can be drawn from these results. First of all, the fact that in this challenging case DeepSets+emb is unable to generalize, while the DGN variants show better performances, suggests that the graph structure contains crucial information for generalization, and that DGN variants are able to exploit it. Moreover, on the basis of the poor

Table 4 Average text scores (± standard deviation) of DeepSets and DGN variants when using one-hot encodings as protein identifiers

Model	UC	MCC	F1	ACC	AUROC
DeepSets	1	.498±.009	.661±.008	.778±.007	.834±.003
DGN	1	**.764±.008**	**.842±.005**	**.896±.004**	**.952±.001**
DeepSets	2	.427±.032	.595±.047	.759±.015	.794±.026
DGN	2	**.543±.043**	**.690±.039**	**.801±.018**	**.851±.014**
DeepSets	3	.146±.076	.284±.107	.677±.028	.622±.042
DGN	3	**.253±.058**	**.486±.047**	**.677±.027**	**.651±.049**

performances of DeepSets+emb in this use case, we can conjecture why it performed reasonably in UC1 and UC2, where only input/output pairs and individual proteins, respectively, were held out from the training set. Our intuition is that in those cases DeepSets learns the correlation between the sensitivity label and the group of proteins, so that when either a new input/output pair for the same group or the same group with a new protein are considered, the learned correlation can be exploited. However, learning this correlation alone is not enough when, as in UC3, the data samples come from different BPs than those used during training. Indeed, different BPs include (mostly) different groups of proteins, making the correlation learned by DeepSets useless. In contrast, the proposed DGN learns useful signals from the graph structure, which allows it to generalize even in more challenging use cases, and exploits the protein embeddings as a means to distinguish among the different proteins. Indeed, as we will show in Sect. 3.2.1, the DGN is able to achieve similar levels of generalization in all use cases even when using orthogonal (i.e., one-hot) protein identifiers instead of protein embeddings.

Finally, it is worth remarking that the generalization signals shown by DGN are particularly relevant by considering the small coverage of our dataset compared to the entire interactome. As discussed in Sect. 3.1, our dataset includes only 1.14% of the proteins and 0.14% of the interactions in BioGRID. Hence, the whole interactome very likely contains a large variety of topologies not included in the dataset and on which it was not possible to train the DGN. We anticipate that performances are likely to increase further once a larger coverage of the interactome will be obtained, thus increasing the representation of different topologies in the dataset.

Performances with one-hot

In this additional experiment, we trained the DGN model by replacing the protein embeddings with one-hot encoded identifiers. This test has been motivated by the improvements we obtained by adding the protein embeddings as a feature. In particular, we wanted to investigate whether the model benefited from using the structural information encoded by the embeddings or it used them as protein identifiers. Thus, we resorted to one-hot encodings, which are simple identifiers that do not contain structural information. As reported in Table 4, the performances are always on par with the ones obtained with the protein embeddings. Furthermore, we observed a faster training convergence (in terms of number of epochs), which suggests that the one-hot features bring information that is easier for the model to learn from. This outcome indicates that

the model mostly needs to distinguish proteins from each other to obtain high performance. Therefore, we conclude that the largest impact on the performances is brought forth by the way the subgraphs are "wired", i.e., by their topology, which the proposed DGN could exploit successfully.

Besides these considerations, the one-hot features serve as a probing tool rather than a modeling solution as they are not generalizable to new proteins. In fact, the one-hot vector can only encode as many proteins as known at training time, and any unknown protein at test time would require non-trivial adjustments.

Error analysis

To complement the evaluation experiments, we conducted an analysis to uncover patterns in misclassified samples. This analysis serves the purpose of understanding what is the contribution of different graph topological features to the sensitivity predictions. To achieve this objective, we stratified the performance metrics with respect to four different characteristics of the input graphs: number of nodes, number of edges, average clustering coefficient, length of the shortest path from the input protein to the output protein, and backward (following the graph connectivity). In this section, we pick the MCC measure for the discussion, but the observations hold for all the other performance measures adopted in this study.

We identified a general trend that holds regardless of the UC or characteristics under study: predictions are more accurate with moderate to high clustering coefficient (CC). This is expected, since dynamical properties are typically inferable on graphs that exhibit some complex structure [32], so it is harder for the model to make accurate predictions at the two edges of the connectivity spectrum, i.e., when the CC is close to 1 (strongly connected graph) or to 0 (weakly connected graph). Figures 7a–c show that performances are lower for graphs with a CC that tends to 0, and decreases slightly when reaching 1. When using protein embeddings, this trend gets smoothed for large CC, because even when the graph is fully connected the model can discern different proteins, hence distinguishing two graphs having an identical skeleton. For UC3 (Fig. 7c), MCC score shows a high variance between folds for lower MCC, but the trend can be observed looking at the shaded area representing the standard deviation, which gets shrunk for medium to high CC values.

Concerning the distance from the input to the output nodes, we found that performances correlate negatively in both directions, meaning that F1 decreases as the input–output distance increases. This tendency, shown visually in Figs. 7(d–f), holds across the different use cases and it is slightly accentuated when the model uses only the I/O node features. Note that a distance of 4 in a PPIN is close to the network diameter, so it can be considered "long" in this context. The model performances up to this distance are consistently above the baseline, so we can conclude that the DGN is able to learn long-distance relationships in PPINs.

As regards graph size, we found that performances with respect to the number of nodes (Figs. 7(g–i) follow trends similar with the I/O distance, as larger graphs will have more distant I/O pairs. Additionally, we can observe worse performances with smaller graphs, which are likely to have trivial structure. A similar behavior can be observed with respect to the number of edges (Figs. 7j–l), though here we can notice a much wider

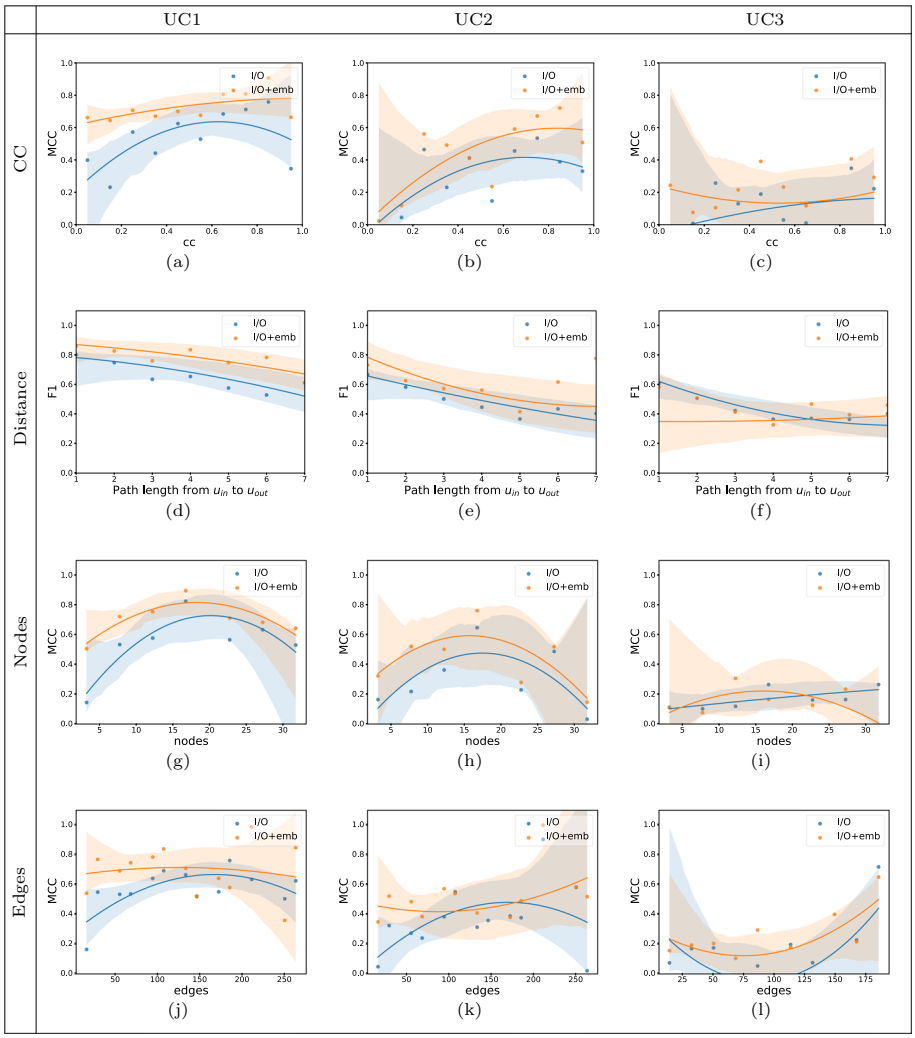


Fig. 7 MCC scores of the proposed DGN model stratified by various features (distance from input to output protein, CC, number of nodes, and number of edges) for the three use cases (UC1, UC2, UC3). Blue points or lines represent performances using only I/O features, while orange ones add protein embeddings as node features. Shaded areas indicate the standard deviation across observations

standard deviation, especially at the extremes of the curves for UC1 and UC2, symptom that the number of edge has not a direct impact on performances.

Case study and application scenarios

In this section, we provide an example of how our designed pipeline can be used to retrieve relevant influences among proteins in a biological network. PPINs are often studied in the context of specific diseases or biological processes, focusing on interactions directly linked to known molecular mechanisms. Our methodology, however, is designed to embed general knowledge about the dynamic behavior of these networks and to exploit recurrent interaction patterns to highlight potential regulatory influences that may not yet be explicitly annotated in BP databases.

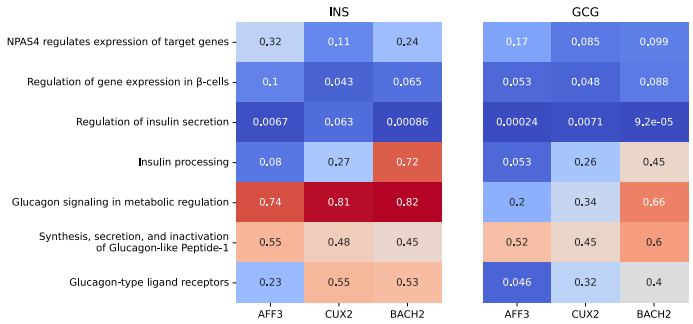


Fig. 8 Predicted sensitivities of insulin (INS) and glucagon (GCG) with respect to the BACH2, AFF3 and CUX2 genes. Values represent the output of the readout sigmoidal head, without thresholding, averaged across the 4 models trained for UC1 without protein embeddings, and normalized by the highest prediction over all the input/output pairs in the BP

To illustrate the pipeline's application, we consider a case study related to type 2 diabetes (T2D). In the study by Son et al. [71], the authors pointed out that BACH2 plays a critical role in T2D-associated β-cell failure, showing that its inhibition can reverse the disease phenotype in experimental models. Their analysis was based on expression data, using single-cell transcriptomics and regulatory network inference to identify BACH2 as a key regulator of the insulin (INS) and glucagon (GCG) proteins, along with the AFF3 and CUX2 genes. We take their analysis as reference knowledge to verify the model's predictions, as it is based on single gene perturbation.

Our approach relies purely on network topology, making no use of expression data or regulatory activity measurements. The goal of this example is to predict INS and GCG dependence on some candidate regulators (BACH2, AFF3, CUX). To do so, we rely solely on the structure of plausible PPINs involving the target proteins, ignoring gene expression derived knowledge.

To construct the analysis, we employ the pipeline detailed in Fig. 4. The PPIN subgraphs were constructed as follows. We queried Reactome [7] for BPs containing the targets, filtering out super-pathways. The motivation behind querying Reactome instead of BioModels is twofold: on the one hand, Reactome contains a wider range of BPs; on the other hand, these BPs are different from the ones used to train the DGN. For each proteins set in the BPs, we retrieved the interaction network from BioGRID, similarly to what has been done for the DyPPIN extraction (see Sect. 2.1). The regulators were not included in any BP, while in the interactome they were just one or two hops away from the BPs proteins. Therefore, we added them in the PPIN subgraphs along with the proteins connecting them. Selected BPs were R-HSA-9768919 (*NPAS4 regulates expression of target genes*), R-HSA-210745 (*Regulation of gene expression in beta cells*), R-HSA-422356 (*Regulation of insulin secretion*), R-HSA-264876 (*Insulin processing*), R-HSA-163359 (*Glucagon signaling in metabolic regulation*), R-HSA-381771 (*Synthesis, secretion, and inactivation of Glucagon-like Peptide-1*), R-HSA-420092 (*Glucagon-type ligand receptors*). We predicted sensitivity for all possible input–output pairs (6531) in the PPINs, using the DGN variant trained without protein embeddings, as they were not available for the three regulators.

In Fig. 8, we report the predicted sensitivity of INS and GCG to the regulators. The scores are obtained by averaging the output of the sigmoidal readout units from the four

models trained over the 4 different folds. BACH2 has the highest score in three out of seven BPs for INS and in 5 out of 7 for GCG, suggesting that it has a relevant role in the regulatory landscape of pancreatic hormones. We remark that this result was obtained without explicitly incorporating prior knowledge about the role of BACH2 in T2D or any expression data, demonstrating that our pipeline can infer relevant interactions based exclusively on the structure of the interaction network.

Although this initial implementation does not incorporate additional BP information, we believe that its generality enables numerous extensions. Co-expression magnitudes could be integrated as edge weights in the PPIN, refining the model's ability to infer regulatory influences. Similarly, BP-specific features could be embedded in the graph representation to enhance sensitivity predictions. Another promising direction is the comparison of predictions across different network variants, such as control and patient-specific interaction networks, to uncover the effects of gene expression alterations.

Beyond disease modeling, our method could support drug target identification. Since it highlights functionally relevant proteins based purely on network topology and dynamics, it can identify potential targets even when they are distant in curated interaction databases. This capability is particularly relevant for diseases where well-characterized molecular mechanisms are lacking or where novel regulatory influences have yet to be mapped. Additionally, the model's ability to infer indirect regulatory effects could help to reveal previously unrecognized intervention points, offering new directions for therapeutic development.

In drug repurposing, our approach could enable fast and systematic screening of potential alternative targets for approved drugs. By leveraging the model's sensitivity predictions, researchers could prioritize proteins that, while not initially associated with a drug's mechanism of action, may still exert a significant regulatory influence in disease-related networks. This could be especially useful for identifying secondary or compensatory BPs that become relevant in drug-resistant conditions, or for expanding the therapeutic scope of existing compounds.

Finally, in a personalized medicine scenario, patient-derived networks could be rapidly constructed and analyzed, enabling individualized assessments of regulatory dynamics. Here, the model's predictions could be integrated as an additional step in BP enrichment analyses performed in differential expression studies, improving the interpretability of functional interactions in PPINs. By incorporating patient-specific network modifications, our approach could provide insights into how individual genetic variations or mutations influence disease mechanisms, potentially guiding tailored therapeutic interventions.

Conclusions

In this work, we have presented a DGN-based framework to infer sensitivity at the PPIN level, by exploiting information obtained at the BP level. This was achieved by first constructing a novel dataset of BPs annotated with sensitivity values obtained through simulation (**DyBP**), and then by explicitly transferring this information to the BioGRID PPIN, ultimately producing the **DyPPIN** dataset. Importantly, the graph nodes in both datasets are annotated with public protein identifiers, which makes them readily usable by existing biochemical pipelines. Furthermore, the information in these datasets can

also be used to provide standard annotations for protein pairs for classic PPI tasks. Both datasets are released for public use.

The **DyPPIN** dataset was used to train a DGN model and predict the sensitivity of pairs of proteins from BioGRID subgraphs by exploiting the graph topology, i.e., by leveraging the way two proteins interact between themselves and among nearby proteins within the PPIN. We robustly assessed the performance of the proposed approach across different use cases and analyzed how the predictive performance vary in relation to the graphs' characteristics. The results demonstrate that the PPIN topology is a major component of the needed signal to infer sensitivity. Furthermore, we showed that model performances can be further enhanced by adding additional sequence information in the form of protein embeddings. This latter point in particular opens up to the application of our methodology to contextualized PPINs, for example those including node-wise gene expression data, to improve performance on specific biological tasks by leveraging specific additional contextual knowledge.

The main issues of our model are the prediction for protein pairs that are far away in the PPIN, and the generalization capabilities on completely unknown PPIN subgraphs. As regards the former, it has been mitigated by employing a DGN variant that is aware of directionality, but efforts should be put in constructing a DGN architecture that better deals with dense and short-diameter graphs. As regards the latter, the generalization capabilities could be improved by extending the training data with a wider range of BPs; this would require the retrieval of pathways from other repositories, like KEGG [6] or Reactome [7], and the development of strategies to compute dynamical properties over their pathways, which cannot be done through ODEs simulation.

Having such a powerful tool can be of great help in many real-world applications. For example, in a drug repurposing scenario practitioners often need to screen a large amount, possibly any, protein to identify potential influences that could lead to new therapeutic uses for existing drugs. The proposed method allows for the selection of the set of proteins targeted by a drug and the prediction of other potential targets through the trained DGN. This is a significantly more rapid process than that of running numerical simulations over BP, with the time taken to perform a single input/output pair prediction being approximately 10^{-3} seconds on average. In comparison, the time taken for a simulation is at least four orders of magnitude larger [40]. We recognize that the need to select a set of proteins instead of operating on the whole PPIN could be a limitation in some use cases where there is no clue about the proteins set that need to be investigated. Future work should focus on learning and predicting dynamical properties on entire interactomes, but this opens new challenges because we need a way to model the dynamics of multiple pathways on a single graph.

Several directions can be taken in future studies. Although this study only concerns sensitivity with respect to the perturbation of an input concentration, it could be applied to analyze sensitivity with respect to the perturbation of other parameters (e.g., reaction rates). Along this line, the remark that the presented approach is general and could be applied seamlessly to the prediction of other dynamical properties (e.g., robustness or monotonicity, as done by Fontanesi et al. [40]). Also, it would be interesting to extend the framework to other biological networks or hybrid networks such as drug-target networks. Lastly, the pipeline could be specialized by adding complex node and edge

features to encode post-translational modifications or regulations. This would enable to specify more complex relationships than the BP-PPIN mapping considered in this study.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-025-06140-1.

Supplementary material 1

Acknowledgements
We would like to thank the University of Pisa Data Center for maintaining the necessary hardware resources for this project.

Author contributions
AM and PM designed the study. AD and PM collected the data, performed the simulations, and prepared the datasets. AD performed the machine learning experiments and wrote the accompanying code. All authors analyzed the experimental results. All authors participated in preparing the initial draft and final version of this work.

Funding
This work was supported by PNRR-M4 C2-Investimento 1.3, Partenariato Esteso [PE00000013], "FAIR-Future Artificial Intelligence Research"-Spoke 1 "Human-centered AI"; and PNRR-M4 C2-Investimento 1.5, Ecosistema dell'Innovazione [ECS00000017], "THE-Tuscany Health Ecosystem"-CUP [I53 C22000780001]-Spoke 6 "Precision medicine & personalized healthcare", both funded by the European Commission under the NextGeneration EU programme. Moreover, this work was partly supported by "Hub multidisciplinare e interregionale di ricerca e sperimentazione clinica per il contrasto alle pandemie e all'antibioticoresistenza (PAN-HUB)" funded by the Italian Ministry of Health (POS 2014-2020, project ID: T4-AN-07, CUP: I53 C22001300001).

Availability of data and materials
The DyBP and DyPPIN datasets in readable format are available as open data via the Zenodo repository: https://doi.org/10.5281/zenodo.14535418. The code for training the machine learning models and making predictions is available via GitHub: https://github.com/alessandrodipalma/sensitivity_ppin_dgn. The code repository also contains a command line tool that allows to predict the sensitivity of any BioGRID PPIN subgraph using the trained model checkpoints. The data processed for the training and prediction, along with the DGN model checkpoints to perform the prediction via the supplied code are available at https://doi.org/10.5281/zenodo.14535760.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no Conflict of interest.

Received: 15 January 2025 Accepted: 10 April 2025
Published online: 08 May 2025

References

1. Bajpai AK, Davuluri S, Tiwary K, Narayanan S, Oguru S, Basavaraju K, Dayalan D, Thirumurugan K, Acharya KK. Systematic comparison of the protein-protein interaction databases from a user's perspective. J Biomed Inform. 2020;103:103380. https://doi.org/10.1016/j.jbi.2020.103380.
2. Szklarczyk D, Kirsch R, Koutrouli M, Nastou K, Mehryary F, Hachilif R, Gable AL, Fang T, Doncheva NT, Pyysalo S, Bork P, Jensen LJ, Mering C. The STRING database in 2023: protein-protein association networks and functional enrichment analyses for any sequenced genome of interest. Nucl Acids Res. 2023;51:638–46. https://doi.org/10.1002/pro.3978.
3. Oughtred R, Rust J, Chang C, Breitkreutz B-J, Stark C, Willems A, Boucher L, Leung G, Kolas N, Zhang F, Dolma S, Coulombe-Huntington J, Chatr-Aryamontri A, Dolinski K, Tyers M. The BioGRID database: a comprehensive biomedical resource of curated protein, genetic, and chemical interactions. Protein Sci. 2021;30:187–200. https://doi.org/10.1002/pro.3978.
4. Orchard S, Ammari M, Aranda B, Breuza L, Briganti L, Broackes-Carter F, Campbell NH, Chavali G, Chen C, del-Toro N, Duesbury M, Dumousseau M, Galeota E, Hinz U, Iannuccelli M, Jagannathan S, Jimenez R, Khadake J, Lagreid A, Licata L, Lovering RC, Meldal B, Melidoni AN, Milagros M, Peluso D, Perfetto L, Porras P, Raghunath A, Ricard-Blum S, Roechert B, Stutz A, Tognolli M, Roey K, Cesareni G, Hermjakob H,. The MIntAct project-IntAct as a common curation platform for 11 molecular interaction databases. Nucl Acids Res. 2014;42:358–63. https://doi.org/10.1093/nar/gkt1115.

5. The UniProt Consortium: UniProt. The universal protein knowledgebase in 2023. Nucl Acids Res. 2023;51:523–31. https://doi.org/10.1093/nar/gkac1052.
6. Kanehisa M, Sato Y, Kawashima M, Furumichi M, Tanabe M. KEGG as a reference resource for gene and protein annotation. Nucl Acids Res. 2016;44:457–62. https://doi.org/10.1093/nar/gkv1070.
7. Milacic M, Beavers D, Conley P, Gong C, Gillespie M, Griss J, Haw R, Jassal B, Matthews L, May B, Petryszak R, Ragueneau E, Rothfels K, Sevilla C, Shamovsky V, Stephan R, Tiwari K, Varusai T, Weiser J, Wright A, Wu G, Stein L, Hermjakob H, D'Eustachio P. The reactome pathway knowledgebase 2024. Nucl Acids Res. 2024;52:672–8. https://doi.org/10.1093/nar/gkad1025.
8. Malik-Sheriff RS, Glont M, Nguyen TVN, Tiwari K, Roberts MG, Xavier A, Vu MT, Men J, Maire M, Kananathan S, Fairbanks EL, Meyer JP, Arankalle C, Varusai TM, Knight-Schrijver V, Li L, Dueñas-Roca C, Dass G, Keating SM, Park YM, Buso N, Rodriguez N, Hucka M, Hermjakob H. BioModels-15 years of sharing computational models in life science. Nucl Acids Res. 2020;48:407–15. https://doi.org/10.1093/nar/gkz1055.
9. Gillespie DT. Exact stochastic simulation of coupled chemical reactions. J Phys Chem. 1977;81(25):2340–61. https://doi.org/10.1021/j100540a008.
10. Bacciu D, Errica F, Micheli A, Podda M. A gentle introduction to deep learning for graphs. Neural Netw. 2020;129:203–21. https://doi.org/10.1016/j.neunet.2020.06.006.
11. Soleymani F, Paquet E, Viktor H, Michalowski W, Spinello D. Protein-protein interaction prediction with deep learning: a comprehensive review. Comput Struct Biotechnol J. 2022;20:5316–41. https://doi.org/10.1016/j.csbj.2022.08.070.
12. Zaki N, Singh H, Mohamed EA. Identifying protein complexes in protein-protein interaction data using graph convolutional network. IEEE Access. 2021;9:123717–26. https://doi.org/10.1109/ACCESS.2021.3110845.
13. Zhang Y, Xiang J, Tang L, Li J, Lu Q, Tian G, He B-S, Yang J. Identifying breast cancer-related genes based on a novel computational framework involving KEGG pathways and PPI network modularity. Front Genet. 2021;12: 596794. https://doi.org/10.3389/fgene.2021.596794.
14. Zhao T, Hu Y, Valsdottir LR, Zang T, Peng J. Identifying drug-target interactions based on graph convolutional network and deep neural network. Brief Bioinform. 2021;22(2):2141–50. https://doi.org/10.1093/bib/bbaa044.
15. Ding Z, Kihara D. Computational methods for predicting protein-protein interactions using various protein features. Curr Protoc Protein Sci. 2018;93(1):62. https://doi.org/10.1002/cpps.62.
16. Baranwal M, Magner A, Saldinger J, Turali-Emre ES, Elvati P, Kozarekar S, VanEpps JS, Kotov NA, Violi A, Hero AO. Struct2graph: a graph attention network for structure based predictions of protein-protein interactions. BMC Bioinform. 2022;23(1):370. https://doi.org/10.1186/s12859-022-04910-9.
17. Elnaggar A, Heinzinger M, Dallago C, Rihawi G, Wang Y, Jones L, Gibbs T, Feher T, Angerer C, Steinegger M, Bhowmik D, Rost B. ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing. arXiv. https://doi.org/10.48550/arXiv.2007.06225.
18. Kewalramani N, Emili A, Crovella M. State-of-the-art computational methods to predict protein-protein interactions with high accuracy and coverage. Proteomics. 2023;23(21):2200292. https://doi.org/10.1002/pmic.202200292.
19. Zewen Xiao Xiao Z, Yue Deng Deng Y. Graph embedding-based novel protein interaction prediction via higher-order graph convolutional network. PloS One. 2020;15: e0238915.
20. Palukuri MV, Patil RS, Marcotte EM. Molecular complex detection in protein interaction networks through reinforcement learning. BMC Bioinform. 2023;24(1):306. https://doi.org/10.1186/s12859-023-05425-7.
21. Zhang D, Kabuka M. Multimodal deep representation learning for protein interaction identification and protein family classification. BMC Bioinform. 2019;20:531. https://doi.org/10.1186/s12859-019-3084-y.
22. Zhang Q, Li J, Xie H, Xue H, Wang Y. A network-based pathway-expanding approach for pathway analysis. BMC Bioinform. 2016;17(17):536. https://doi.org/10.1186/s12859-016-1333-x.
23. Le Novère N. Quantitative and logic modelling of molecular and gene networks. Nat Rev Genet. 2015;16(3):146–58. https://doi.org/10.1038/nrg3885.
24. Cinaglia P. Network alignment and motif discovery in dynamic networks. Netw Model Anal Health Inform Bioinform. 2022;11(1):38. https://doi.org/10.1007/s13721-022-00383-1.
25. Zhang Y, Lin H, Yang Z, Wang J. Construction of dynamic probabilistic protein interaction networks for protein complex identification. BMC Bioinform. 2016;17(1):186. https://doi.org/10.1186/s12859-016-1054-1.
26. Jing X, Zhou Y, Shi M. Dynamic graph neural network learning for temporal omics data prediction. IEEE Access. 2022;10:116241–52. https://doi.org/10.1109/ACCESS.2022.3218027.
27. Costello Z, Martin HG. A machine learning approach to predict metabolic pathway dynamics from time-series multiomics data. NPJ Syst Biol Appl. 2018;4(1):19.
28. Chow K, Sarkar A, Elhesha R, Cinaglia P, Ay A, Kahveci T.: ANCA: Alignment-Based Network Construction Algorithm. IEEE/ACM Transactions on Computational Biology and Bioinformatics 2021;18(2): 512–524 https://doi.org/10.1109/TCBB.2019.2923620 . Conference Name: IEEE/ACM Transactions on Computational Biology and Bioinformatics. Accessed 2025-03-15.
29. Cinaglia P. PyMulSim: a method for computing node similarities between multilayer networks via graph isomorphism networks. BMC Bioinform. 2024;25(1):211. https://doi.org/10.1186/s12859-024-05830-6.
30. Jeong H, Tombor B, Albert R, Oltvai ZN, Barabási A-L. The large-scale organization of metabolic networks. Nature. 2000;407(6804):651–4. https://doi.org/10.1038/35036627. MAG ID: 2144885342.
31. Kitano H. Systems biology: a brief overview. Science. 2002;295(5560):1662–4. https://doi.org/10.1126/science.1069492.
32. Santolini M, Barabási A-L. Predicting perturbation patterns from the topology of biological networks. Proc Natl Acad Sci. 2018;115(27):6375–83. https://doi.org/10.1073/pnas.1720589115.
33. Barabási A-L, Gulbahce N, Loscalzo J. Network medicine: a network-based approach to human disease. Natu Rev Genet. 2011. https://doi.org/10.1038/nrg2918.
34. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle J, Kitano H, Arkin AP, Bornstein B, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Gilles ED, Ginkel M, Gor V, Goryanin I, Hedley WJ, Hodgman TC, Hofmeyr J-HSJH, Hofmeyr Hunter P, Juty N, Kasberger JL, Kremling A, Ursula Kummer Kummer U, Novère Ursula Kummer Ursula Kummer Le, N,

Loew LM, Lucio D, Mendes P, Minch E, Mjolsness E, Nakayama Y, Nelson MR, Nielsen PMF, Sakurada T, Sakurada TTS, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Koichi Takahashi Takahashi K, Takahashi K, Tomita M, Wagner J, Wang J,. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics. 2023;19(4):524–31. https://doi.org/10.1093/bioinformatics/btg015.
35. Clerx M, Cooling MT, Cooper J, Garny A, Moyle K, Nickerson DP, Nielsen PMF, Sorby H. CellML 2.0. J Integr Bioinform. 2020. https://doi.org/10.1515/jib-2020-0021.
36. Beal J, Cox RS, Grünberg R, McLaughlin J, Nguyen T, Bartley B, Bissell M, Choi K, Clancy K, Macklin C, Madsen C, Misirli G, Oberortner E, Pocock M, Roehner N, Samineni M, Zhang M, Zhang Z, Zundel Z, Gennari JH, Myers C, Sauro H, Wipat A. Synthetic biology open language (SBOL) version 210. J Integr Bioinform. 2016;13(3):30–132. https://doi.org/10.1515/jib-2016-291.
37. Bove P, Micheli A, Milazzo P, Podda M (2020) Prediction of dynamical properties of biochemical pathways with graph neural networks:. In: Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies, pp. 32–43. SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0008964700320043.
38. Podda M, Bacciu D, Micheli A, Milazzo P (2020) Biochemical pathway robustness prediction with graph neural networks. In: Proceedings of the 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN),121–126.
39. Podda M, Bove P, Micheli A, Milazzo P (2021) Classification of biochemical pathway robustness with neural networks for graphs. In: Ye, X., Soares, F., De Maria, E., Gómez Vilda, P., Cabitza, F., Fred, A., Gamboa, H. (eds.) Biomedical Engineering Systems and Technologies. Communications in Computer and Information Science, pp. 215–239. Springer. https://doi.org/10.1007/978-3-030-72379-8_11.
40. Fontanesi M, Micheli A, Milazzo P, Podda M. Exploiting the structure of biochemical pathways to investigate dynamical properties with neural networks for graphs. Bioinformatics. 2023;39(11):678. https://doi.org/10.1093/bioinformatics/btad678.
41. Peterson JL. Petri nets. ACM Comput Surv (CSUR). 1977;9:223–52. https://doi.org/10.1145/356698.356702.
42. Nasti L, Gori R, Milazzo P (2018) Formalizing a notion of concentration robustness for biochemical networks. In: Mazzara, M., Ober, I., Salaün, G. (eds.) Software Technologies: Applications and Foundations. Lecture Notes in Computer Science, pp. 81–97. Springer. https://doi.org/10.1007/978-3-030-04771-9_8.
43. Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, Saisana M, Tarantola S(2008) Global Sensitivity Analysis: the Primer. John Wiley & Sons, ???
44. Morris MD. Factorial sampling plans for preliminary computational experiments. Qual Control Appl Stat. 1992;37(6):307–10.
45. Campolongo F, Saltelli A, Cariboni J. From screening to quantitative sensitivity analysis. a unified approach. Comput Phys Commun. 2011;182:978–88. https://doi.org/10.1016/j.cpc.2010.12.039.
46. Burhenne S, Jacob D, Henze GP. Sampling based on sobol' sequences for monte carlo techniques applied to building simulations. Build Simul. 2011;12:1816–23.
47. Micheli A. Neural network for graphs: a contextual constructive approach. IEEE Trans Neural Netw. 2009;20(3):498–511. https://doi.org/10.1109/TNN.2008.2010350.
48. Ye Z, Kumar YJ, Sing GO, Song F, Wang J. A comprehensive survey of graph neural networks for knowledge graphs. IEEE Access. 2022;10:75729–41. https://doi.org/10.1109/ACCESS.2022.3191784.
49. Welsh C, Xu J, Smith L, König M, Choi K, Sauro HM (2022) libRoadRunner 2.0: A High-Performance SBML Simulation and Analysis Library. arXiv. https://doi.org/10.48550/arXiv.2203.01175.
50. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Philos Trans R Soc A: Math Phys Eng Sci. 2016;374(2065):20150202. https://doi.org/10.1098/rsta.2015.0202.
51. Morris C, Ritzert M, Fey M, Hamilton WL, Lenssen JE, Rattan G, Grohe M 2019 Weisfeiler and leman go neural: higher-order graph neural networks. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'19/IAAI'19/EAAI'19 2019. https://doi.org/10.1609/aaai.v33i01.33014602.
52. Rossi E, Charpentier B, Giovanni FD, Frasca F, Günnemann S, Bronstein MM 2024 Edge directionality improves learning on heterophilic graphs. In: Proceedings of the Second Learning on Graphs Conference, pp. 25–12527. PMLR.
53. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;151:1929.
54. Park Y, Marcotte EM. Flaws in evaluation schemes for pair-input computational predictions. Nat Method. 2012;9:1134–6. https://doi.org/10.1038/nmeth.2259.
55. Lannelongue L, Inouye M. Pitfalls of machine learning models for protein-protein interaction networks. Bioinformatics. 2024;40:012. https://doi.org/10.1093/bioinformatics/btae012.
56. Zhu J, Zhou Y, Ioannidis VN, Qian S, Ai W, Song X, Koutra D (2024) Pitfalls in link prediction with graph neural networks: Understanding the impact of target-link inclusion & better practices. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, pp. 994–1002. https://doi.org/10.1145/3616855.3635786.
57. Fawcett T. An introduction to roc analysis. Pattern Recognit Lett. 2006;27(8):861–74. https://doi.org/10.1016/j.patrec.2005.10.010.
58. Chicco D, Jurman G. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. BMC Genomics. 2020. https://doi.org/10.1186/s12864-019-6413-7.
59. Chicco D, Tötsch N, Jurman G. The matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. BioData Min. 2021;14:13. https://doi.org/10.1186/s13040-021-00244-z.
60. Zaheer M, Kottur S, Ravanbakhsh S, Poczos B, Salakhutdinov RR, Smola AJ (2017) Deep sets. In: Advances in Neural Information Processing Systems 30.
61. Errica F, Bacciu D, Micheli A (2021) Graph mixture density networks. In: Proceedings of the 38th International Conference on Machine Learning, pp. 3025–3035. PMLR. ISSN: 2640-3498.

62. Kipf TN, Welling M (2017) Semi-Supervised Classification with Graph Convolutional Networks. arXiv. https://doi.org/10.48550/arXiv.1609.02907.
63. Xu K, Hu W, Leskovec J, Jegelka S (2019) How Powerful are Graph Neural Networks? arXiv. https://doi.org/10.48550/arXiv.1810.00826.
64. Kingma DP, Ba J (2017) Adam: A Method for Stochastic Optimization. arXiv. https://doi.org/10.48550/arXiv.1412.6980.
65. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv. https://doi.org/10.48550/arXiv.1912.01703.
66. Falcon W (2019) The PyTorch Lightning team: PyTorch Lightning. https://doi.org/10.5281/zenodo.3828935.
67. Fey M, Lenssen JE (2019) Fast Graph Representation Learning with PyTorch Geometric. arXiv. https://doi.org/10.48550/arXiv.1903.02428.
68. Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez JE, Stoica I (2018) Tune: A Research Platform for Distributed Model Selection and Training. arXiv. https://doi.org/10.48550/arXiv.1807.05118.
69. Biewald L (2020) Experiment Tracking with Weights and Biases. Software available from wandb.com 2020. https://www.wandb.com/.
70. Fagiolo G (2007) Clustering in complex directed networks 76(2):26107 https://doi.org/10.1103/PhysRevE.76.026107.
71. Son J, Ding H, Farb TB, Efanov AM, Sun J, Gore JL, Syed SK, Lei Z, Wang Q, Accili D, Califano A. BACH2 inhibition reverses β cell failure in type 2 diabetes models. J Clin Investig. 2021. https://doi.org/10.1172/JCI153876.

Publisher's Note