KG embeddings and GNNs for protein-protein interaction prediction

António Estêvão[58203], Diogo Venes[58216], Guilherme Gouveia[58176], and Miguel Dinis[58198]

[1] Faculty of Science of the University of Lisbon
[2] Knowledge Graphs

Abstract. Understanding protein-protein interactions (PPIs) is vital for decoding cellular processes and disease mechanisms. In this study, we explore the integration of knowledge graph embeddings (KGE) and graph neural networks (GNN) to predict PPIs using curated data from the STRING v.12.0 database. Our approach constructs a knowledge graph from protein interaction data and applies KG embedding models to encode semantic relationships. We then leverage Graph Convolutional Networks (GCNs) to learn node representations that incorporate both topological and relational features. The combined embeddings are used to train a classifier that predicts potential protein interactions. Our results demonstrate that hybrid representations of KGE and GNN can significantly enhance predictive performance, offering a promising framework for computational biology applications.

Keywords: GNN · Embeddings · t-SNE · GO · AUC · PPIs.

1 Introduction

Accurate identification of PPIs plays a key role in understanding disease mechanisms and discovering therapeutic targets. However, **experimentally validating PPIs is expensive, time-consuming, and limited in scalability.** This motivates the development of computational methods that can predict protein interactions efficiently and at large scale.

Traditionally, PPI prediction has relied on sequence similarity, structural modeling, or co-expression data. More recently, network-based approaches have emerged, leveraging the fact that proteins interact in complex, structured networks that can be analyzed computationally. Among these, knowledge graphs (KGs) offer a particularly rich representation: they capture not only interactions but also diverse relationships and semantic contexts between biological entities.

In this work, we explore a machine learning approach to PPI prediction that integrates **Knowledge Graph Embedding (KGE)** models with **Graph Neural Networks (GNNs)**. KGE methods encode entities and relationships in a continuous vector space, capturing the semantics of the graph structure in a form

suitable for tasks such as link prediction[1]. GNNs, on the other hand, are deep learning models that operate directly on graph structures, aggregating information from a node's local neighborhood to learn expressive representations.

We built our pipeline on top of curated interaction data from the **STRING V.12.0** database, constructing a knowledge graph in which nodes represent proteins and ontologies (GO) and edges represent observed interactions. We evaluated several embedding models and combined them with GNNs to create embeddings for each protein. These embeddings are then used to train a link prediction model that can identify new or missing protein interactions.

This paper is structured as follows:

- **Section 2** describes the dataset and graph construction process;
- **Section 3** describes our methodology and model;
- **Section 4** presents experiments and discusses the findings;
- **Section 5** presents some limitations and future directions;
- **Section 6** is a brief conclusion.

2 Dataset and Knowledge Graph Construction

2.1 STRING v.12.0 Dataset Overview

The STRING v.12.0 database [2] is a curated repository of biological interactions, including protein-protein interactions (PPIs), genetic interactions, and chemical associations. For this project, we focus exclusively on physical protein-protein interactions, which are most relevant for structure-function prediction and biological inference.

To model protein-protein interactions as a knowledge graph suitable for embedding and link prediction tasks, we constructed a heterogeneous graph by integrating protein identifiers, functional annotations, and ontological relations. The data was derived from processed STRING v.12.0 interactions and Gene Ontology (GO) resources, represented across four CSV files.

- ***idx2entityid.csv*** - This file maps internal indices to biological entity identifiers. Each entity is either UniProt protein ID (e.g., Q8NBZ0) or Gene Ontology (GO) term ID (e.g., GO 0047049)
- ***idx_ppis.csv*** - This file encodes known protein-protein interactions using index references. Each row represents a known interaction between two proteins, forming the core of the knowledge graph's edge set for the *interacts with* relation.
- ***idx_positive_annotations.csv*** - This file links proteins to functional GO terms. These annotations describe the biological processes or molecular functions associated with each protein.
- ***idx_go_links.csv*** - This file represents hierarchical or semantic relationships between GO terms, for example, subclass or *is-a* relations in the ontology.

2.2 Knowledge Graph Construction

With the CSV files we were given, we built a knowledge graph using RDF to be able to model these entities and their relationships.
Then, we created *nodes* for the graph, which are our **proteins** and our **Gene Ontologies**, each identified by their uniprot IDs or GO IDs. Next, we modelled some of the *links/edges* between proteins (**interactions**), between proteins and GOs (**annotations**) and between GOs (**subclass relationships**). Each entity has its own unique URI.
Finally, the KG is serialized and saved in **turtle** format (.ttl), making it ready to be used in tasks further down the line, like embedding generation.

2.3 Train/Test split for Link Prediction

To evaluate a model's ability to generalize to unseen protein-protein interactions (PPIs), **it is essential to avoid data leakage** by ensuring that the same interactions are not used for both training and evaluation. Training on the complete set of known interactions and then evaluating on those same data would lead to overestimated performance. To address this, we partitioned the known PPIs into disjoint training, validation, and test sets. We adopted an **80/10/10 split**, which we deemed appropriate for our task. Initially, we considered applying this split to both the positive (known) and negative (synthetically generated non-interacting) edges. However, after further analysis, we determined that splitting the negative edges separately was unnecessary, as they were independently generated after defining the train, validation, and test partitions for the positive interactions.
The final dataset statistics are as follows:

Statistic	**Count**
Total known interactions	215,278
Training interactions (80%)	172,222
Validation interactions (10%)	21,528
Test interactions (10%)	21,528
Total number of proteins	12,882

Table 1. Final dataset statistics.

3 Methodology

This section describes the pipeline used for predicting the PPIs using KGEs and GNNs. Our approach is entirely data-driven and modular, leveraging the structure of the biological knowledge graph constructed from STRING v.12.0 and Gene Ontology annotations.

3.1 Overview of the Pipeline

Our pipeline for predicting protein-protein interactions (PPIs) integrates knowledge graph representations with graph neural network (GNN) modeling. As already discussed, each node in this graph represents either a protein or a GO term, and edges capture three types of relations: protein-protein interactions, protein-GO annotations, and GO-GO hierarchical links.

To generate feature representations of the graph's entities, we previously computed knowledge graph embeddings, capturing the semantic structure of the graph. These embeddings were then projected into a lower-dimensional space using *t-distributed stochastic neighbor embedding* (**t-SNE**), a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map.

We trained a Graph Neural Network (GNN) to predict potential protein-protein interactions based on a known set of positive interactions. Specifically, a **Graph Convolutional Network (GCN)** was employed to learn **structure-aware node embeddings** by aggregating information from each node's local neighborhood.

In the training process, we represented the protein interaction network as a graph where nodes correspond to proteins, and edges represent **existing (positive) interactions**. These known interactions, provided by the dataset, were encoded in the `edge_index` field of the PyTorch Geometric `Data` object [3]. This field defines the actual **graph topology** and is exclusively used to enable the GCN to perform **message passing** [4]— that is, to propagate and aggregate information between connected nodes. Importantly, only **positive interactions** are included in this graph structure, as they reflect the true known state of the biological network.

To perform link prediction, we constructed a separate set of node pairs, some of which correspond to known positive interactions, while others are **negative samples** — i.e., pairs of proteins for which no interaction has been observed. These candidate links were stored in the `edge_label_index` field, and the corresponding ground-truth labels (1 for interacting pairs, 0 for non-interacting pairs) were stored in `edge_label`.

This separation is crucial: while `edge_index` defines the **true graph used to learn contextual embeddings**, the node pairs in `edge_label_index` serve as **prediction targets**. These target pairs may or may not be present in the original graph, and including them in the structural graph would result in **data leakage**, contaminating the learning process with information that should only be available during evaluation.

Once the GCN has learned node embeddings by aggregating messages across

the graph defined by `edge_index`, it is used to predict the likelihood of interaction between protein pairs. For each candidate pair in `edge_label_index`, the embeddings of the two proteins are concatenated and passed through a simple classifier. This classifier serves as a **decoder**, outputting a score that reflects the predicted probability of interaction. The predicted scores are then compared to the corresponding binary labels in `edge_label` using a suitable loss function, such as **binary cross-entropy**.

This approach allows the model to leverage both the **semantic patterns captured in the global knowledge graph** and the **local structural topology** of the interaction network, enabling more accurate and context-aware link prediction.

3.2 Embedding Generation and Dimensionality Reduction

To capture the semantic structure of the heterogeneous knowledge graph, we used **RDF2Vec** [5,6,7], *"an approach that uses language modeling approaches for unsupervised feature extraction from sequences of words, and adapts them to RDF graphs"*. RDF2Vec generates vector representations for entities by extracting random walks from the knowledge graph and training Word2Vec models over these sequences. This approach is particularly suitable for graphs where rich relational and hierarchical information, such as GO term ontologies and protein-function annotations, needs to be encoded in the embeddings.

In our case, RDF2Vec was applied to the full knowledge graph, which includes protein-protein interactions, protein-GO annotations, and GO-GO semantic links. The resulting embeddings encode both the topological context and relational semantics of each protein and GO term in the graph.

After generating the RDF2Vec embeddings, we applied t-SNE 1 to reduce their dimensionality, primarily to facilitate visualization. This technique proved especially useful for exploring the structure of the protein embedding space, revealing more isolated clusters of proteins as well as denser, more entangled regions. These patterns suggest varying levels of connectivity, with isolated groups likely corresponding to proteins with fewer interactions and the central convoluted areas representing proteins that are more interconnected within the network.

3.3 Graph Neural Network Architecture

To incorporate structural information from the protein-protein interaction (PPI) network, we trained a **Graph Convolutional Network (GCN)** over the interaction graph using the PyTorch Geometric (PyG) framework. The goal of the GCN was to learn protein embeddings that reflect both their local topological neighborhoods and the global semantic features previously captured by RDF2Vec.

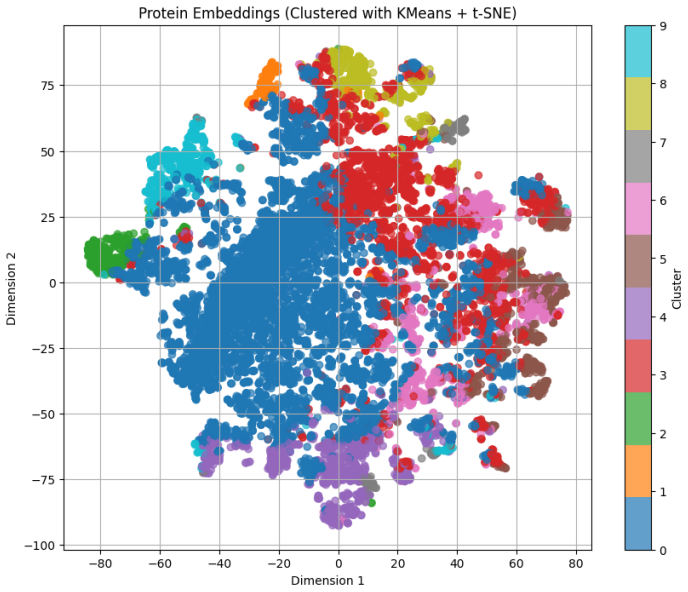

Fig. 1. t-SNE w/ Clustering

The input to the GCN consisted of the **RDF2Vec embeddings** reduced to 100 dimensions for each of the 12,882 proteins. These embeddings were treated as initial node features. The graph topology, defined by protein-protein interactions from the dataset, was used to construct the *edge index* tensor, which encodes the adjacency information for the GCN.

Negative Sampling Strategy: To train the model on a binary link prediction task, we needed to generate **negative examples** - protein pairs that are considered not to interact.
Two strategies were adopted:

- **Random Negative Sampling**: Negative edges were generated by randomly pairing proteins that are not connected in the graph.
- **Similarity-Based Negative Sampling**: As an alternative, a more informed approach was developed by computing the **cosine similarity** between protein embeddings. From a candidate pool, the least similar protein pairs (those with minimal cosine similarity) were selected as negative examples, in the hopes that these proteins would be the least likely to interact. This method aims to reduce noise in the training data by selecting negative samples that are less likely to be false negatives (i.e., pairs that might actually interact but are not yet annotated).

Additionally, a third strategy was initially proposed, **Full Similarity-Based Negative Sampling**, where we computed the **cosine-similarity** between all

protein embeddings, as opposed to just the candidates. However, this strategy was far too computationally expensive and was therefore discarded.

GCN Setup: The GCN model consists of the two-layer GCN using *GCNConv* layers: the first layer takes the 100-dimensional RDF2Vec features and outputs a hidden representation. A ReLU activation is applied between layers. The second layer outputs a new embedding for each protein node.

Given pairs of proteins (positive and negative), the model learns to classify whether an edge exists between them. The learned embeddings of each protein pair are concatenated and passed to a downstream classifier.
Importantly, only real interactions were used to train the GCN. The negative samples were used only during the classifier training phase, not in the GCN's graph structure.

3.4 Integration Strategy

We integrate semantic knowledge from the RDF2Vec embeddings directly into the GCN model by using them as **initial node features**. Rather than combining multiple embeddings after training, our approach embeds semantic and structural information in a single forward pass. This allows the GCN to learn interaction-relevant representations that are grounded in both the topology of the protein-protein interaction graph and the rich functional context encoded in the knowledge graph.

These embeddings are passed as input to a two-layer Graph Convolutional Network, which is trained over a graph constructed from experimentally validated protein-protein interactions. The GCN propagates and refines the RDF2Vec features through the interaction network, allowing the model to learn enriched node representations that reflect both local graph structure and global semantic context.

3.5 Methodology Summary

With the GCN trained to generate updated protein embeddings, the link prediction task is formulated as a binary classification problem over protein pairs. For each pair, the embeddings produced by the GCN are concatenated and passed through a simple feedforward neural network that outputs a probability score indicating the likelihood of interaction.

The training set consists of both positive and negative examples. Positive pairs are extracted from experimentally validated PPIs, while negative pairs are sampled from non-interacting protein combinations. Two sampling strategies are supported: random selection and a similarity-based approach that selects protein pairs with minimal cosine similarity in the embedding space. This aims to

create more meaningful negative examples by avoiding false negatives.

The classifier is trained using binary cross-entropy loss, and performance is evaluated using accuracy and ROC AUC metrics. Importantly, while the GCN is trained only on the positive interaction graph, the downstream classifier leverages both positive and negative edges to learn to discriminate interacting from non-interacting protein pairs.

4 Results and Discussion

To determine the optimal hyperparameters for our models, we conducted a grid search over 24 different configurations, varying the hidden dimensions (128 or 256 in the first layer and 16 or 32 in the second) and learning rates (0.001, 0.01, 0.05). This allowed us to systematically explore how model architecture and training dynamics influence performance, ensuring a fair comparison between Rand-GCN and Cand-GCN variants.

We evaluated each configuration using multiple metrics, including validation loss, accuracy, and **AUC** (Area Under the ROC Curve). Among these, AUC was our primary metric, as it provides a comprehensive measure of a model's ability to distinguish between positive and negative protein pairs, regardless of threshold. This is particularly important in imbalanced classification scenarios like protein-protein interaction prediction, where relying on simple accuracy can be misleading. Even with an equal number of positive and negative edges, certain proteins may have significantly more interactions than others, resulting in a realistic yet inherently unbalanced setting. As shown in the table 2, the highest validation AUC (0.8463) was achieved by the Rand-GCN model with a hidden dimension of 128–32 and a learning rate of 0.01. The best-performing Cand-GCN model reached a slightly lower AUC of 0.8325, using a larger hidden dimension (256–32) and a learning rate of 0.05.

Interestingly, the Rand-GCN models achieve higher AUC scores than the Cand-GCN models in most configurations. This suggests that random negative sampling may make it easier for the model to learn the difference between interacting and non-interacting proteins. In contrast, the similarity-based sampling used in Cand-GCN might include more challenging or ambiguous pairs, which could make training harder and reduce overall performance.

We also observe some outliers with notably lower AUC values (e.g., Cand-GCN with hidden dim 256–16 and learning rate 0.05 achieving only 0.6696). These cases likely reflect unstable training due to overly aggressive learning rates or poor convergence resulting from suboptimal architecture-depth combinations. This reinforces the importance of careful tuning in models with more complex sampling strategies.

Table 2. Grid search results for GNN models on validation AUC.

Model	Hidden Dim 1	Hidden Dim 2	Learning Rate	Val AUC	Epochs
Rand-GCN	128	16	0.001	0.8103	142
Rand-GCN	128	16	0.01	0.8431	69
Rand-GCN	128	16	0.05	0.8375	41
Rand-GCN	128	32	0.001	0.8238	150
Rand-GCN	128	32	0.01	**0.8463**	72
Rand-GCN	128	32	0.05	0.8373	45
Rand-GCN	256	16	0.001	0.8100	133
Rand-GCN	256	16	0.01	0.8371	47
Rand-GCN	256	16	0.05	0.8410	102
Rand-GCN	256	32	0.001	0.8193	110
Rand-GCN	256	32	0.01	0.8424	59
Rand-GCN	256	32	0.05	0.8433	93
Cand-GCN	128	16	0.001	0.7724	122
Cand-GCN	128	16	0.01	0.8192	45
Cand-GCN	128	16	0.05	0.8313	40
Cand-GCN	128	32	0.001	0.7897	125
Cand-GCN	128	32	0.01	0.8292	47
Cand-GCN	128	32	0.05	0.8284	84
Cand-GCN	256	16	0.001	0.7657	93
Cand-GCN	256	16	0.01	0.8230	62
Cand-GCN	256	16	0.05	0.6696	17
Cand-GCN	256	32	0.001	0.7818	97
Cand-GCN	256	32	0.01	0.8183	51
Cand-GCN	256	32	0.05	**0.8325**	38

Figure 2 shows the training and validation loss curves for the best AUC values of each model (Rand-GCN and Cand-GCN). Both models exhibit smooth convergence trends, with the Rand-GCN continuing to improve steadily up to 70 epochs. In contrast, the Cand-GCN appears to have benefited from early stopping, as its validation loss stabilizes around epoch 20–25, indicating a plateau and helping prevent overfitting.

Notably, the Rand-GCN achieves a lower final loss value on both training and validation sets, suggesting more robust generalization. The overall lower and more consistent loss values reinforce the superior performance observed in the Rand-GCN's validation AUC scores.

5 Limitations and Future Work

An interesting direction for this project could be using **GraphSAGE**, a framework for inductive representation learning on large graphs. This approach is particularly useful in settings like ours, where there is a strong class imbalance, as it can improve the generalization of the model to unseen data and potentially

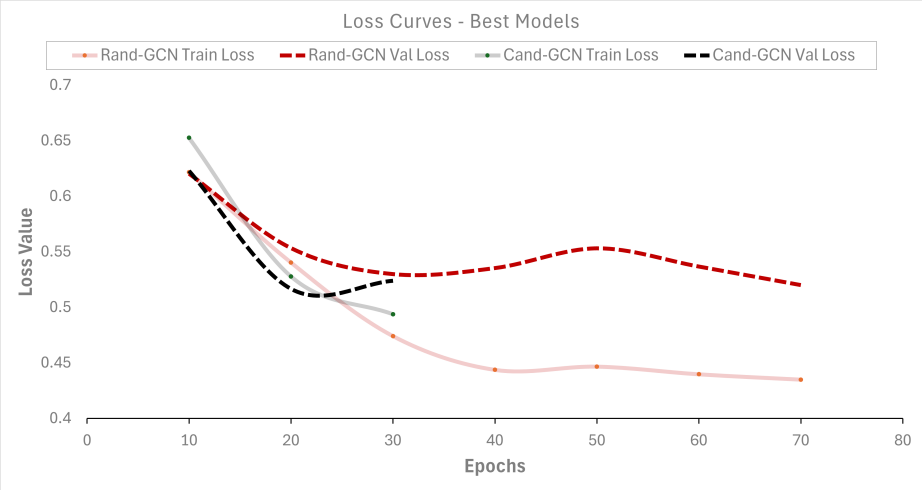

Fig. 2. Loss Curves

mitigate representation bias by aggregating information from node neighborhoods.

In terms of limitations, our candidate negative interaction sampling method (used in **Cand-GCN**) did not perform better than the random method (used in **Rand-GCN**) as we wanted. Therefore, we could try to explore more sophisticated methods of sampling negative interactions, possibly using other biological features of the proteins to get more informed generations. Generally speaking, we could take into account additional biological information as node features. On the other hand, the full-candidate generation method was computationally too expensive to be considered for use in this project, so we could try to explore ways to make it more efficient.

Future work could also include advances in explainability, as our GNN is a black-box model that makes it difficult to interpret why the model makes a specific prediction. This is a common issue with deep learning models and is particularly significant in the biological realm, where understanding the basis for predictions is important.

Finally, further advances could be achieved by collaborating with biologists to validate novel predicted interactions.

6 Conclusion

In this work, we presented a novel pipeline for protein-protein interaction (PPI) prediction that integrates semantic information from knowledge graph embeddings (RDF2Vec) with structural insights from Graph Neural Networks (GCNs). By constructing a biologically meaningful knowledge graph from curated STRING v.12.0 data, we captured both explicit interaction patterns and contextual func-

tional annotations. The RDF2Vec embeddings provided a rich semantic foundation which, when combined with GCN-based structural learning, yielded high-quality protein representations. These representations were used in a downstream link prediction task, achieving promising results with an AUC of up to 0.8463 during validation.
Our approach demonstrates the benefit of combining knowledge graph-based semantics with deep learning on structured biological networks. The modularity of the pipeline also opens up avenues for extension, including the integration of additional biological entities (e.g., diseases or drugs), more advanced embedding techniques, or alternative GNN architectures.
Overall, our results highlight the potential of knowledge-driven graph learning approaches in advancing computational biology and systems-level understanding of cellular functions.

Acknowledgments. We would like to extend a special thank you to our teacher Cátia Pesquita and to the PHD student Laura Balbi for the availability and constant help they gave us during this project.

References

1. Deep Graph Library (DGL): Link Prediction Tutorial. https://www.dgl.ai/dgl_docs/tutorials/blitz/4_link_predict.html, last accessed 2025/05/24
2. STRING Database, https://string-db.org/, last accessed 2025/05/22
3. PyTorch Geometric Documentation. https://pytorch-geometric.readthedocs.io/en/latest/, last accessed 2025/05/24
4. Hussein, O.: Graph Neural Networks Series — Part 4: The GNNs, Message Passing & Over-Smoothing. https://medium.com/the-modern-scientist/graph-neural-networks-series-part-4-the-gnns-message-passing-over-smoothing-e77ffee523cc, last accessed 2025/05/24
5. PyRDF2Vec: RDF2Vec in Python. https://pypi.org/project/pyrdf2vec/, last accessed 2025/05/24
6. PyRDF2Vec Documentation. https://pyrdf2vec.readthedocs.io/en/latest/, last accessed 2025/05/24
7. Ristoski, P., Rosati, J., Paulheim, H.: RDF2Vec: RDF Graph Embeddings for Data Mining. *Semantic Web*, 10(6), 887–899 (2019). https://doi.org/10.3233/SW-180317