universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Graph-Enhanced Gene Embeddings for Biological Process Prediction: Architecture Design and Topological Analysis

verfasst von | submitted by

Zoé Dézsi BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2026

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 821
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Mathematik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Phys. Dr. Jörg Menche

Acknowledgements

Jörg, thank you for your support and encouragement; they meant a lot to me throughout this process.
Felix, thank you for bringing me to the LBI NetMed team and for your support.
Chloé, thank you for always being available for discussion and for your advice.
To the entire LBI NetMed team, thank you for being so inspiring. It has been a great experience sharing knowledge with you.

Emese, we have such a genuine and uplifting friendship, and I always look forward to our conversations.
Oskar, you have been so reassuring to me, and I deeply value your presence and the way you think.
Paul, over the past three years, your friendship has been one of the few constants in my life, and I am so grateful for you.

Alexandra, Zalán, el se tudom képzelni, mi lenne velem nélkületek. Anya, Apa, nekem vannak a világon a legjobb szüleim. Köszönöm, hogy mindig számíthatok rátok.

Abstract

Gene function prediction is fundamental to understanding cellular processes and disease mechanisms, yet annotation through traditional biological experiments remains challenging. While graph neural networks have shown promise using protein-protein interaction networks, integration strategies for complementary functional similarity graphs are underexplored. This thesis compares six embedding architectures to determine optimal dual-graph integration strategies for biological function prediction.

We evaluate three models based on a dual-view graph neural network architecture: alternating concatenation, early concatenation, and late concatenation. Moreover, three single-view baselines across multiple Gene Ontology filtering strategies are used to assess prediction performance.

Key findings establish that processing graphs independently before concatenation outperforms both early fusion and complex alternating strategies. Using only the functional similarity graph derived from Gene Ontology molecular function annotations yields better performance than using the protein-protein interaction network alone, likely because its modular structure more closely reflects underlying biological processes. All models maintain high recall while differing in precision, revealing false positive reduction as the critical limitation. Selective term filtering substantially improves precision by addressing class imbalance.

Zusammenfassung

Die Vorhersage der Genfunktionen ist grundlegend für das Verständis zellulärer Prozesse und Krankenheitsmechanismen, doch die Annotation durch traditionelle biologische Experimente bleibt eine Herausforderung. Obwohl Graphneuronale Netze vielversprechende Ergebnisse bei der Verwendung von Protein-Protein-Interaktionsnetzwerken gezeigt haben, sind Integrationsstrategien für ergänzende Graphen der funktionalen Ähnlichkeit doch wenig erforscht. Diese Arbeit vergleicht sechs Einbettungsarchitekturen, um optimale Strategien zur Integration von Doppelgraphen für die Vorhersage biologischer Funktionen zu ermitteln.

Wir bewerten Modelle basierend auf einer Dual-View-Graph-Neural-Network-Architektur: alternierende Konkatenation, frühe Konkatenation und späte Konkatenation. Darüber hinaus werden drei Einzel-View-Baselines über mehrere Gene Ontology-Filterstrategien hinweg verwendet, um die Vorhersageleistung zu bewerten.

Wichtige Erkenntnisse zeigen, dass die unabhängige Verarbeitung von Graphen vor der Zsammenführung sowohl die frühe Fusion als auch komplexe alternierende Strategien übertrifft. Die alleinige Verwendung des Funktionsähnlichkeitsgraphen, der aus den Gene Ontology-Annotationen der molekularen Funktion abgeleitet wurde, führt zu einer besseren Leistung als die alleinige Verwendung des Protein-Protein-Interaktionsnetzwerks, wahrscheinlich weil seine modulare Struktur die zugrunde liegenden biologischen Prozesse genauer widerspiegelt. Alle Modelle weisen eine hohe Erkennungsrate auf, unterscheiden sich jedoch in der Präzision, was zeigt, die Reduzierung von Falsch-Positiv-Ergebnissen die kritische Einschränkung darstellt. Die selektive Termfilterung verbessert die Präzision erheblich, indem sie das Ungleichgewicht der Klassen berüchsichtigt.

Contents

List of Figures

1 Introduction

1.1 The gene function prediction challenge

A central challenge in biology is the determination of functional properties of gene products, i.e., RNAs and proteins. [24] The purpose of gene function prediction is to annotate unknown genes to the correct functional category in the database, such as the Gene Ontology (GO) knowledge base. [25] The vocabulary system provided by the Gene Ontology is the minimal information necessary to define gene function by using three domains: cellular component (CC), molecular function (MF) and biological process (BP). [26] The accurate annotation of gene functions is essential to understanding life at the molecular level and has great biomedical and pharmaceutical implications. [27] However, despite the rapid growth of the number of sequenced genomes, it is usually difficult to annotate gene functions through traditional biological experiments due to high cost. [25] Moreover, there is a strong annotation bias in the GO annotations where 58% of the annotations are for 16% of the human genes, suggesting that many human genes are poorly annotated or under-represented in Gene Ontology data. [28]

This annotation gap motivates the development of computational methods for gene function prediction that can leverage existing knowledge and data.

Two primary data sources have been used for gene function prediction: protein-protein interaction networks and the GO annotations.

1.2 Limitations of the protein-protein interaction networks and the Gene Ontology annotations

The topology extracted from protein-protein interaction networks has proven to be a highly valuable source of information for predicting gene function with the help of network-based analysis. [25] However, protein-protein interaction networks, when used for gene function prediction, also come with significant limitations.

Current PPI databases capture around $100{,}000 - 300{,}000$ of the human interactome (i.e. the network of protein-protein interactions), while the estimated number of potential interactions is around 650,000. [29] This incompleteness is particularly problematic for poorly studied proteins. Present day human interactome resources show bias toward well-studied proteins, since less studied ones frequently have fewer documented interactions and fewer structural data available. [30] Consequently, function prediction methods relying solely on PPI may fail for genes with limited interaction data.

PPI networks capture direct and/or physical associations but miss many functional relationships. Genes can, for example, participate in the same biological processes or pathways without physically interacting. [31, 32, 33]

These limitations highlight the need for integrating PPI networks with complementary data sources. Gene Ontology annotations provide functional relationships independent of physical interactions however, Gene Ontology annotations have their limitations as well. In addition to the already mentioned strong annotation bias, the evidence codes of the annotations reflect different levels of support and specificity. Experimental evidence codes represent stronger, direct support than inferred or automated annotations, meaning annotation reliability and specificity varies widely. [5]

Moreover, gene prediction methods which rely solely on Gene Ontology annotations have the problem that only the positive annotations of genes are reported and thus,

recorded in the Gene Ontology knowledge base. The lack of negative annotations limits the discriminative ability of these function prediction models. [34]

We see that neither source alone is sufficient, but since they are offering orthogonal information (i.e. PPI networks capture interaction topology and protein complexes, while GO annotations capture broader functional relationships) their integration may overcome individual limitations.

1.3 Dual-view integration of PPI network and GO annotations

In this work, we compare three dual-view autoencoder architectures (alternating concatenation, early concatenation, and late concatenation) based on graph neural networks on the task of gene function prediction. In particular, we construct two graphs as the input to these models: a protein-protein interaction network representing physical binding relationships, and a functional similarity graph derived from molecular function Gene Ontology annotations, where edges connect genes that are functionally similar to each other. By processing both graphs through the graph convolutional layers, the models learn gene embeddings that integrate the complementary structural and functional information. What we are interested in is how the different architectural choices impact the capacity of these embeddings predicting GO biological process terms. This would give us insight into which models learn generalizable functional representations.

The alternating concatenation model is an adaptation of the the scNET framework developed by Sheinin et al. [1]. They introduced a dual-view autoencoder architecture based on graph neural networks using an attention mechanism that alternates between the protein-protein interaction and cell-cell relations (coming from gene expression similarity) graphs to denoise single-cell RNA-seq data and learn context-specific gene embeddings.

In the early concatenation model the two graphs are first processed through separate graph convolutional branches before being concatenated and further processed as one.

The late concatenation model processed the two graphs completely separately, they are only merged together for the final gene representation.

In addition to these three dual-view architectures, we also introduce two models based on graph neural networks that only take either the protein-protein interaction network or the functional similarity graph as input. Comparing these to the three dual-view architectures helps us gain a deeper understanding of which source of information (structural vs. functional) results in better embeddings for predictions.

To put the performances of these models into perspective and to emphasize the importance of graph-based information, a non-graph-based baseline model is used.

2 Background

Network science provides a framework through which different disciplines can come into a seamless interaction with each other. The core tools utilized by network science come from the mathematical field of graph theory. [2] Network biology, a field that has been around for two decades, focuses on understanding cellular functions and diseases across biological systems and scales, by representing a biological system as an interconnected entity rather than a collection of individual components. [3]

In this chapter, we present the biological theory and mathematical tools used throughout this thesis.

2.1 The human interactome

Molecular interaction networks play a crucial role in analysing these biological processes. The interactome (here equivalent to the protein-protein interaction (PPI) network however, in a wider context the interactome includes all molecular interactions, and in that case the protein-protein interaction network is only a component of the it) is one such molecular interaction network, whose nodes represent genes that are linked to each other by physical (binding) interactions. [4] Studying the topology of the protein-protein interaction network has resulted in gaining a better understanding of how the structural position of a protein within the network relates to its biological importance. In particular, proteins that are located in the centre of the network or in topologically influential positions are often enriched for disease-associated genes and known drug targets, highlighting that network architecture itself can reveal fundamental molecular players in pathogenic processes. [12]

The PPI network is a scale-free network which is characterized by a degree distribution that follows a power law. Mathematically, this means that

$$p_k \sim k^{-\gamma},$$

where p_k is the probability that a node has exactly k edges and γ is the degree exponent. This distribution has structural consequences; there is a large number of small degree nodes and there are a few high-degree nodes, which we also call hubs. [2]

2.2 The Gene Ontology knowledge base

The functional molecules produced by the genes are called gene products (proteins and noncoding RNAs) which perform different functions at molecular, cellular, and organismal levels. The Gene Ontology (GO) knowledge base is the most comprehensive representation of gene function across three domains: molecular function (MF), cellular component (CC) and biological process (BP). In the GO knowledge base, classes of gene functions (called GO terms) are defined and are related to each other using a directed acyclic graph (DAG). The key relations include "is a" (subclass of), "part of" (component), and "has part". This gives a hierarchical structure with multiple parentage allowed. The statement that a given gene or gene product possesses a specific functional characteristic represented by the GO term is called a GO annotation. These annotations also include the evidence they are based upon, experimental data being the most reliable. In general, multiple GO annotations are required to completely describe the function of a gene product. [5, 6]

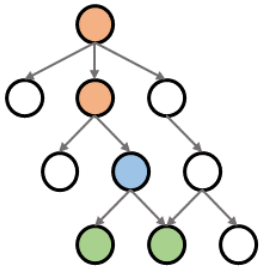

Figure 1: Representation of the Gene Ontology directed acyclic graph. The two orange nodes are the ancestors and the two green nodes are the descendants of the blue node.

2.3 Graph neural networks

Graph neural networks (GNNs) are deep neural networks that operate directly on graph structured data. GNNs emerged as the generalization of convolutional neural networks (CNNs) to non-Euclidean domains. While CNNs require fixed-size, grid-like input (such as images or sequences), GNNs can naturally handle the irregular, relational structure of graphs. [49] This capability makes them particularly suited for biological networks, where genes, proteins, and their interactions form complex graph topologies.

Motivated by a first-order approximation of localized spectral filters on graphs, Kipf and Welling introduced graph convolutional networks (GCNs) [50]. We follow their article closely.

Let $G = (V, E)$ be a graph with $n = |V|$ nodes, adjacency matrix $A \in \mathbb{R}^{n \times n}$, and node feature matrix $X \in \mathbb{R}^{n \times d}$, where d is the feature dimension. A GCN layer then computes new node representations by aggregating features from local neighbourhoods:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

where

- $H^{(l)}$ are the node representations at layer l with $H^{(0)} = X$
- $\tilde{A} = A + I_n$ is the adjacency matrix with added self-loops
- $\tilde{D}$ is a diagonal matrix such that $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
- $W^{(l)}$ is the weight matrix
- σ is the nonlinear activation function (e.g., $\mathrm{ReLU}(.) = \max(0, .)$)

The normalization term $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ serves two purposes. First, adding self-loops with I_n ensures that each node incorporates its own features alongside those of its neighbours. Second, the symmetric normalization controls the scale of feature propagation across nodes of varying degree. This prevents high-degree nodes from dominating the aggregation and stabilizes the learning process. [50]

Multiple GCN layers can be stacked together. An L-layer GCN aggregates information from nodes that are at maximum L steps away from the central node, enabling learning representations that reflect both local neighbourhoods and broader network context. [50] However, stacking too many layers comes with the problem of oversmoothing: node representations become increasingly similar as depth increases, eventually converging to

indistinguishable values [51]. In [52], the authors state that most current GCN models are shallow due to oversmoothing and stacking more layers tend to degrade performance without architecture modifications. The GCN model proposed by Kipf and Welling [50] achieves the best performance with 2 layers [52].

In the GCN structure described above, each neighbour contributes according to degree-based normalization, with all edges treated uniformly in the aggregation [50]. In biological networks, however, interactions can vary widely in functional relevance [53]. This motivates attention mechanisms that learn to weight neighbours adaptively during aggregation.

Graph attention networks (GATs) were introduced by Veličković et al. [54]. Here we describe the method closely following this article.

The input to a single graph attention layer is a set of node features, $\{h_1, ...h_n\}$, $h_i \in \mathbb{R}^d$, where n is the number of nodes, d is the number of features of each node. The output is then a new set of node features $\{h'_i, ..., h'_n\}$, $h'_i \in \mathbb{R}^{d'}$, where d' is not necessarily equal to d.

The attention mechanism computes attention coefficients α_{ij} for each edge (i, j), determining how much attention node i pays to node j when updating its representation. These α_{ij} are computed in three steps:

In the first step, attention scores are computed, using a shared attention mechanism $a : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to \mathbb{R}$:

$$e_{ij} = a(Wh_i, Wh_j)$$

where $W \in \mathbb{R}^{d' \times d}$ is the learned weight matrix describing the shared linear transformation of the node representations. In the article,

$$a(Wh_i, Wh_j) = \text{LeakyReLU}\left(\mathbf{a}^T(Wh_i||Wh_j)\right)$$

where $||$ is the concatenation operation, $\mathbf{a} \in \mathbb{R}^{2d'}$ is a weight vector and

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ cx & \text{else} \end{cases}$$

where c is a nonnegative constant. In the case of $c = 0$, we get the ReLU (Rectified Linear Unit) function.

The second step is the normalization, using the softmax function:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

where $\mathcal{N}_i$ is some neighbourhood of node i in the network. As a way of incorporating the graph structure into the model, the e_{ij} are only computed for $j \in \mathcal{N}_i$. This is a so call masked attention. The normalization helps to make the coefficients more comparable across different nodes. Moreover, since the coefficients for each node sum to one, they also provide a probability distribution of the neighbours.

Lastly, the new node representations are computed:

$$h'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W h_j\right)$$

where σ is a nonlinear function.

GATs use multi-head attention to stabilize learning. K independent attention mechanisms compute different attention patterns, and in the final layer their outputs are averaged:

$$h_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k W^k h_j\right)$$

where α_{ij}^k are normalized attention coefficients computed by the k-th attention mechanism and W^k is the corresponding input linear transformation's weight matrix.

Kipf and Welling introduced graph autoencoders (GAEs) [55], an unsupervised learning framework that learns node representations by reconstructing graph structure. In the article [55], they are defined in the following way: given an undirected, unweighted graph $G = (V, E)$ with $n = |V|$ nodes, let A be the adjacency matrix of G and D its degree matrix. The $n \times d$ node feature matrix is denoted with X.

Then the autoencoder architecture consists of the encoder:

$$Z = \mathrm{GNN}(X, A)$$

where $Z \in \mathbb{R}^{n\times m}$ contains m-dimensional embeddings of the nodes. And the decoder:

$$\tilde{A} = \sigma(ZZ^T)$$

where $\tilde{A}$ is modeled as a Bernoulli random variable, $\tilde{A}_{ij}$ being the probability of an edge between nodes i and j, i.e.

$$\tilde{A}_{ij} = P(A_{ij} = 1 \mid z_i, z_j) = \sigma(z_i^T z_j)$$

The training objective is the minimization of the reconstruction error. Under the assumption that $\tilde{A}$ is a Bernoulli random variable, minimizing the negative log-likelihood yields a binary cross-entropy reconstruction loss:

$$\mathcal{L} = -\sum_{i,j}\left[A_{ij}\,\log(\tilde{A}_{ij}) \;+\; (1 - A_{ij})\,\log(1 - \tilde{A}_{ij})\right].$$

This is suitable for link prediction, for reconstruction of node features other appropriate losses can be used depending on the feature type.

2.4 Multi-view learning

Multi-view learning addresses the challenge of learning from data described by multiple heterogeneous feature sets or domains. The variables associated with each data point can naturally be partitioned into groups, which can be referred to as views. [56] In biological networks, different views might represent different types of relationships (e.g., physical interactions, functional similarities) that describe the same set of genes.

Xu et al. [56] identified two fundamental principles of multi-view learning: the consensus and the complementarity principle. The consensus principle states that different views of the same data should agree on predictions and exploit consistency across different views to improve learning performance. While, according to the complementarity principle, each view may contain information not present in the others.

The simple concatenation of features from multiple views treats them as independent sources of information, thereby overlooking the inter-view relationships and potentially leading to challenges associated with high dimensionality. Effective integration methods

simultaneously consider the consensus and complementarity principles to maximize the utilization of information across the different views. [57]

Biological and medical datasets are often incomplete and noisy, therefore the integration of diverse data sources is needed to achieve a more comprehensive understanding of biological systems. [58] Multi-view learning offers a principled framework for integrating complementary biological evidence while enforcing consistency across views, leading to more robust representations.

2.5 UMAP

UMAP (Uniform Manifold Approximation and Projection) [40] is used with two purposes in this thesis; 2-dimensional UMAP visualizations are used as one of the assessment criteria for choosing a feature selection method, and we construct the functional similarity graph using the k-nearest neighbour graph computed internally by the UMAP algorithm.

UMAP is a dimensionality reduction algorithm grounded in manifold theory and topological data analysis. While it is typically used for visualization or dimensionality reduction, in the first part of the algorithm, a weighted k-nearest neighbour graph is constructed, capturing the local manifold structure of the higher dimensional data. This graph can be used independently from the low-dimensional projection done in the second part of the algorithm. This second part we do not use in our method. In this section, we describe UMAP's graph construction procedure.

We follow the graph construction steps as described in [40]. Let $X = \{x_1, ..., x_N\}$ be the input dataset with a metric $d : X \times X \to \mathbb{R}_{\geq 0}$. For each x_i the set $\{x_{i_1}, ..., x_{i_k}\}$ of the k nearest neighbours of x_i under the metric d is computed, where k is a chosen hyperparameter. Then for each x_i we define

$$\rho_i := \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k,\ d(x_i, x_{i_j}) > 0\}$$

and set σ_i to be the value such that

$$\sum_{j=1}^{k} \exp\left(\frac{-\max\left(0, d(x_i, x_{i_j}) - \rho_i\right)}{\sigma_i}\right) = \log_2(k)$$

holds. Using these, a weighted directed graph $\bar{G} = (V, E, w)$ is defined, where the nodes V of $\bar{G}$ are the set X. The set of directed edges E is defined as $E = \{(x_i, x_{i_j}) \mid 1 \leq j \leq k,\ 1 \leq i \leq N\}$ and the weight function w as

$$w((x_i, x_{i_j})) := \exp\left(\frac{-\max\left(0, d(x_i, x_{i_j}) - \rho_i\right)}{\sigma_i}\right).$$

The selection of ρ_i ensures that x_i connects to at least one other data point with an edge of weight 1, which is in alignment with the local-connectivity constraint that is needed for the theoretical justification of the method.

As a last step, this directed graph is transferred to an undirected graph. Let A be the weighted adjacency matrix of $\bar{G}$, and define the symmetric matrix

$$B := A + A^T - A \circ A^T,$$

where $\circ$ is the pointwise product. If we interpret A_{ij} as the probability that the directed edge $x_i \to x_j$ exists, then B_{ij} is the probability that the edges $x_i \to x_j$ or $x_j \to x_i$ exist. The UMAP graph is then the undirected weighted graph whose adjacency matrix is given by B.

In [40], one can find the theoretical explanation and justification of this construction using category theory, in particular fuzzy simplicial sets.

UMAP has been used in computational biology, particularly for single-cell RNA-seq analysis. [41] There are several reasons for this:

- UMAP is scalable. It has computational complexity approximately $O(n \log(n))$, making it practical for large biological datasets.
- UMAP preserves both local and global data structure [41], making it suitable not only for visualization but also for constructing similarity graphs.
- UMAP's distance metric can be customized for different data types. For sparse, high-dimensional biological data, metrics like cosine distance could perform better than Euclidean distance.

Moreover, when compared to other ways of constructing graphs, UMAP has some advantages.

Computing k-nearest neighbours with uniform edge weights is simpler, but this method does not account for varying data density. In areas where data is sparse, fixed weights may overemphasize weak similarities and in dense areas they may underweight strong local structure.

Another possibility is to compute similarity scores between the genes based on the distances between the feature vectors (using a chosen metric). However, this results in a complete graph, which requires sparsification, for which there are a variety of methods. UMAP's k-parameter naturally controls graph sparsity.

Diffusion maps [42] also capture manifold structure but are typically more computationally expensive ($O(n^2)$ to $O(n^3)$) and are more sensitive to kernel and scale hyperparameters, requiring more careful tuning, when compared to UMAP.

2.6 Network-based gene function prediction methods

Gene function prediction can be formulated as a multi-label classification problem. Given a set of genes and a set of functional terms (typically from Gene Ontology), the task is to predict a binary association matrix Y where

$$Y_{ij} := \begin{cases} 1 & \text{if gene i has function j} \\ 0 & \text{else.} \end{cases}$$

One of the foundational principles of computational gene function prediction methods is guilt-by-association [44], meaning that genes sharing certain properties are likely to share functions too. The roots of this concept lie in molecular biology; Eisenberg et al. [45] discussed the implication that interacting proteins tend to share common biological functions.

This guilt-by-association method is often applied to biological networks. The genes (nodes) in the networks are connected by edges representing relationships such as physical

interactions, coexpression or sequence similarity. In this representation, the guilt-by-association principle is understood in the following way: genes that are in each other neighbourhoods in the network tend to share functions.

The principle received systematic computational validation from Wolfe et al. [46], who demonstrated its applicability across diverse gene coexpression networks. By analysing multiple expression datasets and comparing coexpression patterns with Gene Ontology annotations, they showed that coexpressed genes are more likely to share GO annotations.

However, guilt-by-association also has its limitations. The principle makes the assumption that network data can adequately capture functional relationships, but protein interaction networks are known to be incomplete and noisy. [47] Another issue with association networks is that they are typically not condition-specific, the quality of the evidence sources of the interactions in these networks can be significantly different. [3] In [48], Gillis and Pavlidis demonstrate that only a very small fraction of edges encode true functional information and that large portions of the network cannot be reliably used for functional prediction.

Early network-based function prediction methods applied guilt-by-association principles directly to PPI networks. Schwikowski et al. [35] proposed a simple majority-vote scheme by ordering the annotated functions of all neighbours of a given protein in the PPI, from the most frequent to the least frequent. Functions that occurred the same number of times were ordered arbitrarily. The first three functions in the order were then declared as predictions for the function of this given protein.

Network propagation approaches [7] such as random walk with restart and heat diffusion follow fixed transition rules and iteratively spread information across network edges starting from a set of nodes called seed nodes resulting in importance scores which can be interpreted as gene representations. Although these are powerful methods for protein function prediction and can be used for predicting functional associations of unannotated gene sets [8], they also have several limitations: the transition rules are pre-defined, they are not learned, the resulting representations are specific to previously chosen seed nodes, and in most cases, when predicting protein functions, a single network is used, it is less common to use multiple graphs.

Another technique that has been used for gene function prediction is matrix factorization. An example of such a method is netNMF-sc (Network-regularized Non-negative Matrix Factorization for single-cell data) [9] which learns a low-dimensional representation of scRNA-seq transcript counts using network constraints through graph Laplacian regularization combined with non-negative matrix factorization. These learned gene embeddings can be then used for function prediction. However, since these methods model gene relationships as linear combinations of latent factors, they are unable to capture non-linear relations.

The emergence of graph embedding methods has enabled more sophisticated network-based methods. DeepWalk [36] uses local information of the network from truncated random walks to learn latent representations of the nodes by treating walks as the equivalent of sentences. Node2vec [37] uses a biased random walk procedure to explore diverse neighbourhoods in the network and by maximizing the likelihood of preserving network neighbourhoods of nodes, low-dimensional feature representations of the nodes are learned. Both of these methods have been commonly used to node classification problems including

protein function prediction and related tasks. [39]

scLINE [10] is a network embedding-based dimensionality reduction approach that provides a representation of high dimensional and sparse single-cell scRNA-seq data. The distance between the joint probability distribution in the lower dimensional space and the empirical distribution of the original space is minimized to ensure that the low dimensional vectors can preserve the similarity of the network. This method integrates multiple biological networks (PPI, co-expression, pathways).

Graph neural networks (GNNs) have emerged as powerful tools for learning representations on graph-structured data, enabling end-to-end learning that integrates network topology with node features. PINNACLE [11] uses graph neural network layers to generate protein representations by integrating the PPI topology and the context derived from gene expression.

The most directly related work to the approach presented in this thesis is scNET [1], which introduced a dual-view graph neural network architecture for integrating single-cell RNA-seq data with PPI networks. scNET employs alternating graph convolutions on two complementary graphs: the PPI network (gene-gene interactions) and a KNN graph derived from cell-cell expression similarity. scNET demonstrated that dual-view integration produces embeddings that better capture functional annotations, improve cell clustering, and enable more accurate pathway analysis compared to methods using either data source alone. The key innovation is the alternating message passing between two graph views, allowing information to flow between physical interaction constraints (PPI) and expression-based similarity (KNN). However, scNET is specifically designed for single-cell analysis.

All these methods have produced successful results, however, they primarily focus on the PPI network alone or in combination with gene expression data and treat functional annotations, if used, merely as auxiliary features rather than explicitly modeling functional similarity as a structured graph that can be jointly learned with physical interactions.

3 Materials and methods

This chapter presents the complete methodological framework of the thesis. We first describe the datasets and their sources, then detail the preprocessing steps and construct the functional similarity network. Furthermore, we present three dual-view, two single-view architectures for gene function prediction as well as a baseline model, followed by the descriptions of the evaluation metrics used to assess model performance.

3.1 Selection of model datasets

In this section, we provide details on the sources of the data used for the construction of the interactome and the Gene Ontology annotation feature matrices.

3.1.1 Construction of the interactome

For the construction of the protein-protein interaction network four sources are being used. Firstly, the two sources containing directly measured protein-protein interactions, are Bioplex [13] (version from May 2021) and HuRI [14] (version from April 2020). Direct association refers to those interactions that occur between molecules which are in direct contact with each other. The Bioplex interactome contains 13,957 proteins and 118,162 interactions, while the HuRI contains 6,047 proteins and 13,441 interactions. When we combine them, we get an interactome with 15,698 proteins (nodes) and 168,759 interactions (edges). Since the total number of protein-coding genes in the human body, according to GENCODE [15], is 19,962, we select additional protein-protein interactions from two other sources; Biogrid [16] (version from November 2021) and IntAct [17] (version from September 2021). These contain direct and physical associations, where physical associations mean interactions between molecules within the same physical complex.

Finally, we have a protein-protein interaction network with 18,068 nodes and 306,914 edges. However, to ensure that the network contains only genes that are in our GO molecular function feature matrix, we remove all genes that are not in the intersection of "Genes in the PPI network" and "Genes with MF annotations" in Figure 2 along with their interactions. Thus, the protein-protein interaction network that we work with contains 16,585 nodes and 283,971 edges across 13 connected components.

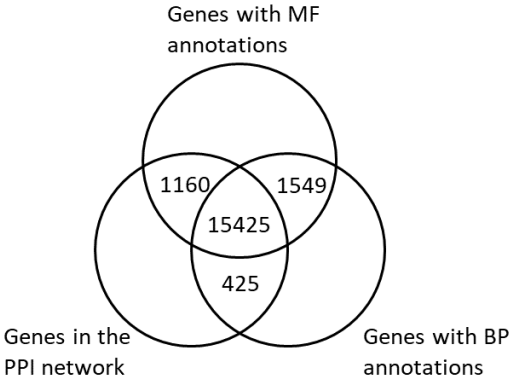

Figure 2: Venn diagram of the annotated genes and the genes in the PPI network.

3.1.2 Construction of the Gene Ontology molecular function feature matrix

We are working with the subgraph of the Gene Ontology graph that includes all the molecular functions with the relation "is a". We have 18,269 genes that are annotated. If a gene is annotated by a specific GO term, it is also considered annotated by all of that term's ancestor terms in the GO hierarchy. The genes that are only annotated by the root term of the branch (GO:0003674) are removed, as this does not provide enough information for further analysis. This leaves 18,219 genes with MF annotations. The matrix with genes as rows and GO terms as columns is referred to as the Gene Ontology molecular function feature matrix, or simply the MF feature matrix. Its entries consist only of zeros and ones, each GO term carrying the same weight.

GO terms that have many descendants in the downwards directed Gene Ontology acyclic graph are assumed to be less informative, since they most likely do not carry much specificity. In order to incorporate this differentiation between the GO terms according to their position in the DAG, we amplify more specific GO terms (i.e. terms with fewer descendants) by multiplying each GO column by

$$-\log \frac{n}{N},$$

where N is the number of all GO terms in our GO MF DAG branch and n is the number of descendants of the given GO term. After weighing, the entries are in the interval $[0, \infty)$. For the feature selection methods used later, it is better to have the entries in the interval $[0, 1]$.

We only want to work with genes that are also contained in the PPI network, for this reason, genes that are not in the PPI network are removed from the MF feature matrix. The resulting MF feature matrix contains 16,585 samples / genes (rows) and 5,061 features / GO MF terms (columns). The fraction of zero entries is 0.9972, thus we have an extremely sparse feature matrix. No rows sum to zero.

3.1.3 Construction of the Gene Ontology biological process feature matrix

The Gene Ontology biological process feature matrix (BP feature matrix for short) is constructed following the same steps as we had for the MF feature matrix. We are working with the biological process subgraph of the GO DAG with the relation "is a". There are 17,776 genes with BP annotations, and just as before, if a gene is annotated by a specific GO term, it is also considered annotated by all of that term's ancestor terms in the GO hierarchy. The genes that are only annotated by the root term of the branch (GO:0008150) are removed. Thus, we have 17,719 genes with BP annotations remaining. We will only have embedding representations for the genes that are in the MF feature matrix, so we additionally remove any genes from the BP feature matrix that are not in the MF feature matrix. The BP feature matrix has the same structure as the MF feature matrix, however as the target for the gene function prediction this matrix is left unweighted.

The final form of the BP feature matrix has 15,425 samples / genes (rows) and 14,221 features / GO BP terms (columns). The fraction of zero entries is 0.9962 and no rows sum to zero.

3.2 Feature selection process for the MF feature matrix

The aim of feature selection is to reduce the dimension of the feature space and filter out noise. We start with rougher methods and gradually move towards more sophisticated

methods. The feature selection is applied to the MF feature matrix, as this matrix will be used for the learned embeddings.

The first step is to normalize the columns as a preparation for feature selection. The aim is to have all the entries in the interval $[0, 1]$. We perform the min-max normalization

$$x_{\text{normalized}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}},$$

where x is the original value, $x_{\min}$ is the minimum value, $x_{\max}$ is the maximum value in the matrix. We choose this method because this is a linear transformation thus the relative distances between values are maintained.

3.2.1 Assessment criteria for feature selection

1. Quantifying GO term specificity

   We want our choice of feature selection method to be guided by biological context. The aim is to examine whether the discarded GO terms systematically differ from the retained ones with respect to their level of specificity in the GO hierarchy. Brenton et al. [18] have established four methods to quantify GO specificity.

   - Number of Ancestors: Measures how "deep" a GO term is in the hierarchy by counting ancestor terms up to the root.

     In general, this measure does not tell much about the specificity of a GO term, since there can be many shallow leaf nodes in the GO graph. For this reason, we do not include this measure in our evaluation of the feature selection methods.

   - Number of Offspring: A normalized count of child terms that increases with specificity, using the following formula:

     $$\text{Offsp}_N(t) = \log(N + 1) - \log(\#\text{Offspring}_t + 1),$$

     where N is the number of offspring of the root node and $\#\text{Offspring}_t$ is the number of offspring nodes of the given GO term t.

   - GO Proportion: Incorporates both the number of ancestors and offspring of the given GO term t as a proportion ranging from 0 (non-specific) to 1 (highly specific), using the following formula:

     $$\text{P}_t = 1 - \frac{\#\text{Offspring}_t}{\#\text{Offspring}_t + \#\text{Ancestors}_t}$$

   - Information Content (IC): This is a data-dependent measure based on the probability of a GO term occurring in the feature matrix, where rarer terms have higher IC and are assumed to be more specific. IC of a given GO term t is calculated the following way:

     $$\text{IC}(t) = -\log_2(p(t)),$$

     where $p(t)$ is the probability of that term t occurring in a data set.

   Following each feature selection method, we analyse these specificity metrics to determine how they were influenced by that method.

2. UMAP and cluster analysis

We employ 2-dimensional UMAP visualizations to assess feature selection quality. After applying a feature selection method, the aim is to see well-separated clusters indicating that GO terms effectively differentiate functional groups. Whereas, too scattered, diffused distributions would suggest poor feature selection. For these exploratory visualizations, we use the same hyperparameters (cosine distance, k=50) as for the later constructed functional similarity graph to maintain consistency, though here we project to 2-dimension rather than extracting the graph. The choice of these parameters is explained under section 3.3.

To cluster the data, we use agglomerative hierarchical clustering with average linkage method (which is sometimes called "unweighted pair-group average method" and abbreviated as "UPGMA"). To describe the algorithm we follow [19]. In each iteration we calculate the distance between each pair of clusters and define a distance matrix, using the following formula:

$$D(R,Q) = \frac{1}{|R||Q|} \sum_{i \in R,\ \ j \in Q} d(i,j),$$

where R and Q are clusters, $|R|$ and $|Q|$ denote the number of elements in each clusters. Moreover, $d(i,j)$ stands for the distance of the two data points i and j. Our chosen distance metric is the cosine distance which is defined as,

$$d(i,j) := 1 - \frac{i \cdot j}{||i||_2 ||j||_2},$$

where $||.||_2$ is the 2-norm of its argument, and $i \cdot j$ is the dot product of i and j. With other words, the cosine distance is $1 -$ cosine similarity. After the distance matrix is computed, three steps are repeated till all points are combined into a single cluster or a stopping criterion is met: find the pair of clusters with minimum distance, merge these clusters, and update the distance matrix. The algorithm produces a dendrogram (tree diagram) that visualizes the hierarchical structure and allows determining the optimal number of clusters by "cutting" the tree at an appropriate height. The stopping criterion in our case is at the highest point in the dendrogram that still yields a given maximum number of clusters that we want.

To find the optimal number of clusters we use the silhouette score, first introduced in [20]. It quantifies how well each data point fits within its assigned cluster compared to other clusters. For a given data point i, which is in cluster A, two quantities are computed:

$$a(i) = \text{average distance of } i \text{ to all other data points in } A,$$

where we use the same distance metric we used for the clustering. And

$$b(i) = \min_{C \neq A} d(i,C),$$

where $d(i,C)$ denotes the average distance of i to all data points in C. Then the silhouette coefficient $s(i)$ for the data point i is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}.$$

The silhouette score ranges from -1 to $+1$. When $s(i)$ is close to $+1$, that means that $a(i)$ is much smaller than $b(i)$, i.e. i is closer to the data points in cluster A than to the data points in other clusters. We can say that the assignment of i to cluster A is well-chosen. When $s(i)$ is around zero, then $a(i)$ and $b(i)$ are approximately equal, it is thus not clear, in which cluster should i be assigned to. When $s(i)$ is close to -1, it implies that $a(i)$ is much larger than $b(i)$, so i should not be in cluster A.

The overall quality $S(K)$ of the clustering method K is then measured by averaging over all $s(i)$ for all data points i. From a range of Ks we select the one for which $S(K)$ is maximal.

In the assessment, the goal is to reduce the optimal number of clusters in our data, while maximizing the silhouette score. Furthermore, we also want to reduce the number of trivial clusters (clusters that only contain one data point).

3.2.2 First feature selection method - Combination method

In the first feature selection method, we want to combine some classical feature selection approaches. For this reason, this method is named as "Combination method".

As one of the feature selection steps we want to use entropy-based feature selection. Therefore, we examine how the entropy distribution across features changes before and after applying other feature selection steps. This helps us understand the range of entropy values we are dealing with and choose an appropriate threshold for the entropy-based step.

The Shannon entropy [67] of the i-th feature is defined in the following way:

$$H(i) = -\sum_j P_j^i \log P_j^i$$

where P^i is the probability distribution vector of the i-th feature.

Some advantages of measuring entropy are:

- It works with any type of distribution.
- It is scale invariant, the meaning of entropy is consistent across features.
- It can handle categorical and numerical variables, so it is generally applicable.
- Since it is based on probability distribution and not on the distance from the mean (like variance), it is a more robust measure of diversity in the presence of outliers.

Looking at Figure 3, we observe that many features exhibit relatively low entropy.

The values of low entropy features can be predicted on average from little information (i.e. these features are more predictable / come with more certainty), thus they are less informative when it comes to distinguishing genes based on these features.

In contrast, high entropy features require more information to predict their values, meaning that they are less predictable / carry greater uncertainty. However, this also means they tend to contain more informative and descriptive content. This is something that is beneficial for our analysis.

Overall, we aim to retain features with moderate to high entropy. After completing all feature selection steps, we expect that most of the filtered-out features are of low entropy.

We apply three feature selection methods in this order: variance thresholding, removal of highly correlated features and entropy-based feature selection.

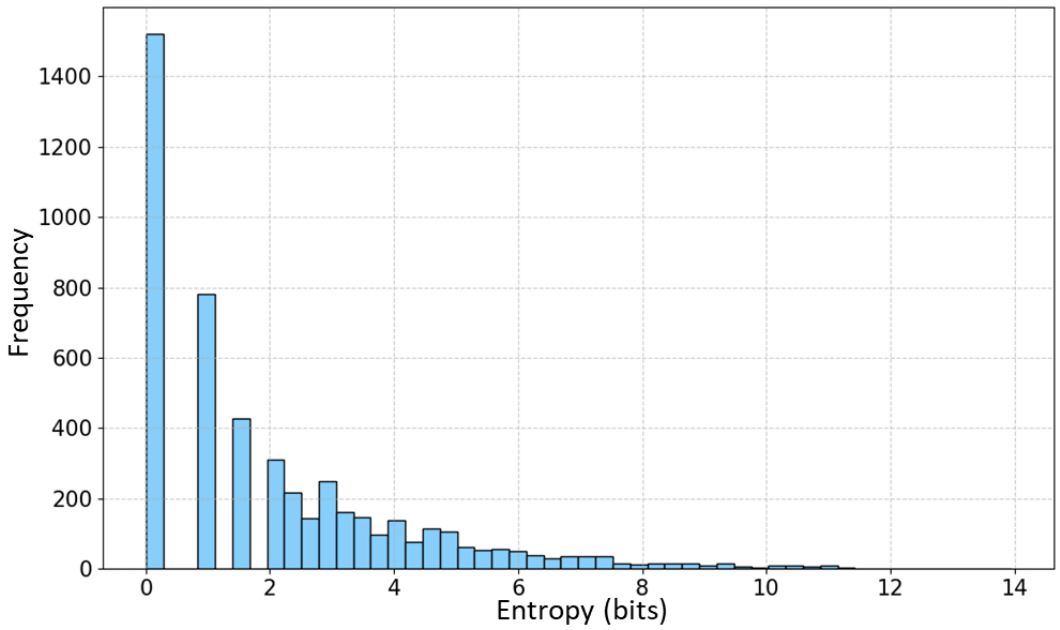

Figure 3: Distribution of entropy across GO terms.

1. Variance thresholding

   We remove all features with low variance. The variance of the i-th GO term is calculated as follows:

   $$\sigma_i^2 = \frac{\sum_j (x_{ji} - \bar{x}_i)^2}{N - 1},$$

   where x_{ji} is the j-th entry in the i-th column of the feature matrix, $\bar{x}_i$ is the average of the entries in the i-th column of the matrix and N is the number of genes that we have.

   To choose an appropriate threshold, we first examine the distribution of the sorted feature variance values. In Figure 4, we see that the vast majority of features have low variance. The threshold of 0.0001 is chosen since it marks the point at which the variance curve starts to increase.

   After removing all features with variance below 0.0001, 3,540 features remain, eliminating approximately 30% of all features. Looking at Figure 5 and comparing it to Figure 3, we observe that this step removes primarily low-entropy features, those that annotate only a small number of genes.

2. Removal of highly correlated features

   As a second step, we compute pairwise correlations between feature columns and remove one feature from each pair whose correlation exceeds a chosen threshold. Because the choice of which feature to remove is random, there is a risk of discarding the more informative feature in a highly correlated pair.

   To calculate the pairwise correlation, we use the Pearson correlation coefficient. The correlation coefficient between the i-th and k-th feature, using the same notations as by the variance thresholding, is defined as follows:

   $$r_{ik} = \frac{\sum_j (x_{ji} - \bar{x}_i)(x_{jk} - \bar{x}_k)}{\sqrt{\sum_j (x_{ji} - \bar{x}_i)^2 \cdot \sum_j (x_{jk} - \bar{x}_k)^2}}.$$

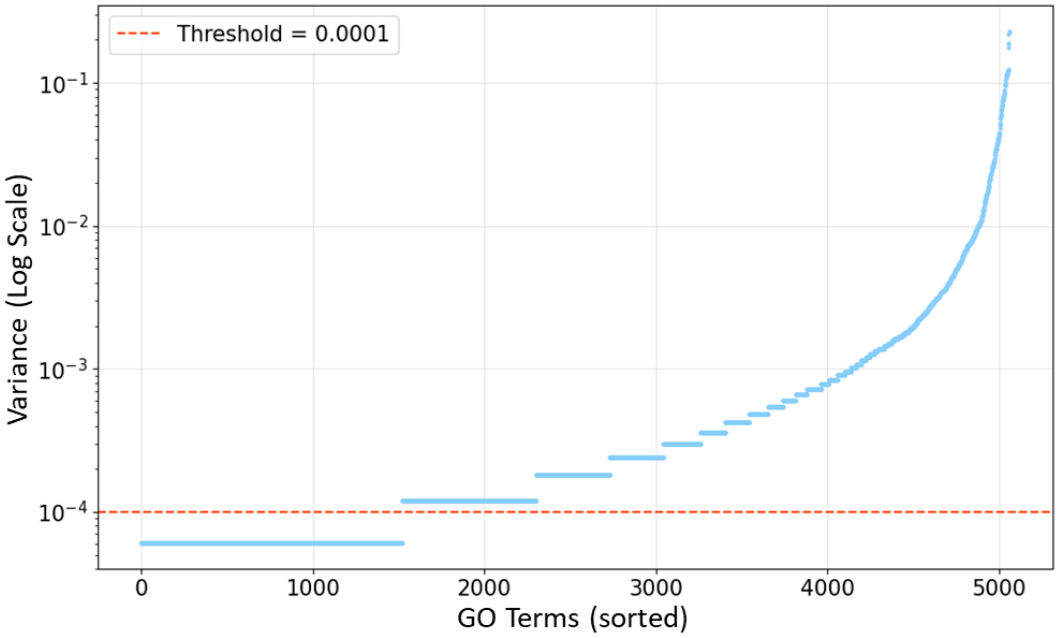

Figure 4: Distribution of sorted feature variances. The horizontal red dashed line indicates the selected variance threshold (0.0001), showing that the large portion of features fall below this cutoff and would be removed during variance-based filtering.

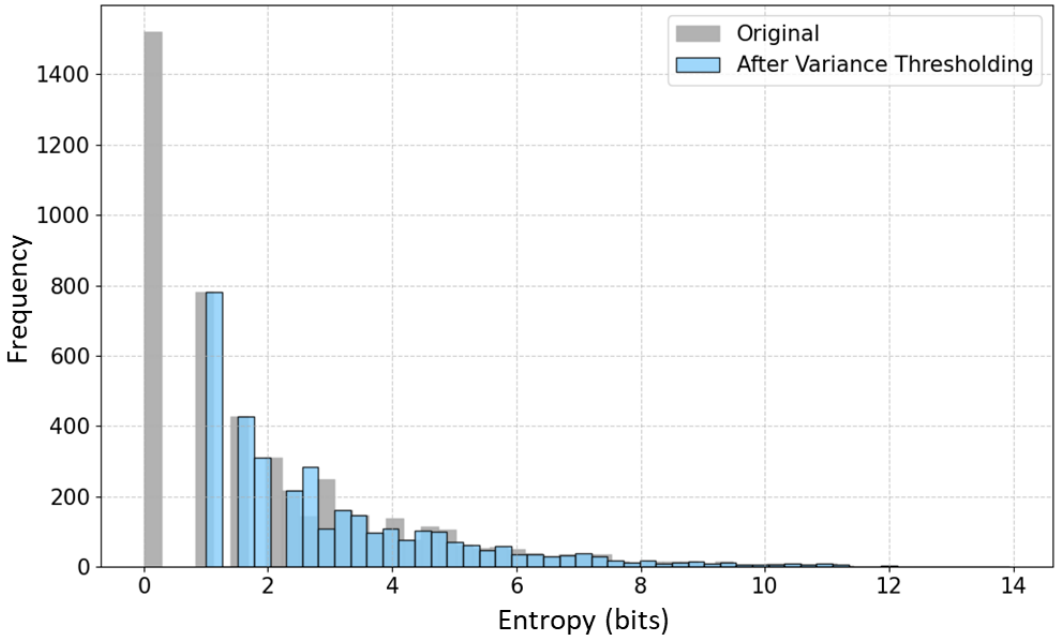

Figure 5: Distribution of entropy across GO terms after variance thresholding.

Figure 6 tells us that the majority of features are uncorrelated, and thus this step should eliminate far fewer features compared to the earlier filtering. Looking at the plot, we choose 0.99 as the threshold for removing highly correlated features. A total of 3,260 features remain, and Figure 7 shows no visible difference compared to Figure 5.

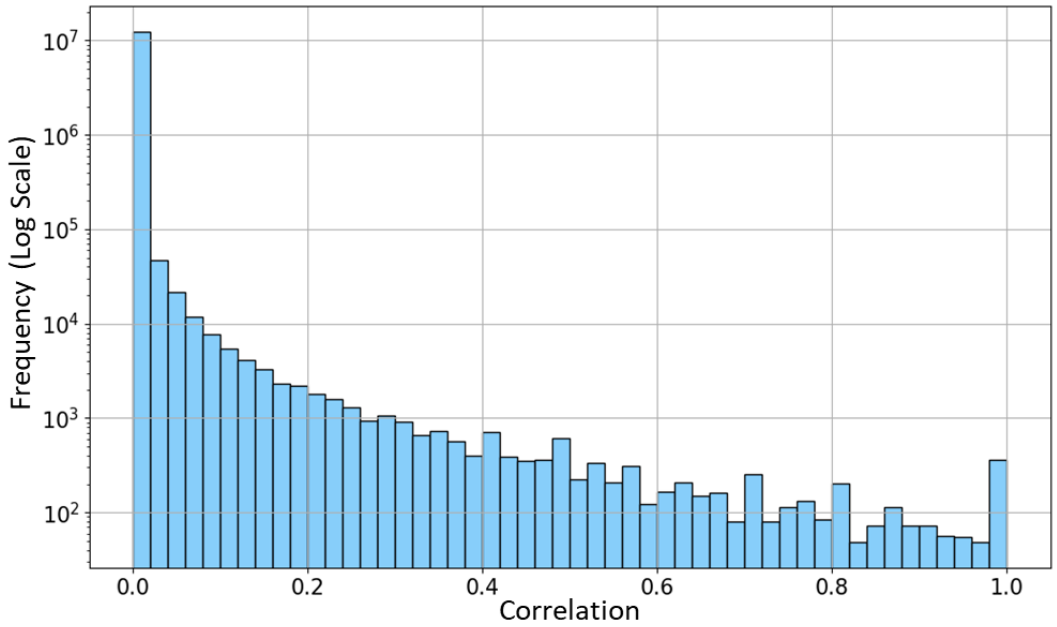

Figure 6: Distribution of absolute pairwise feature correlations after variance thresholding.

3. Entropy-based feature selection

   In the last step, we focus on keeping features with moderate to high entropy. Such features are informative without being completely random (very high entropy) or entirely static (very low entropy). Consequently, we filter out GO terms whose entropy falls below the 10th percentile.

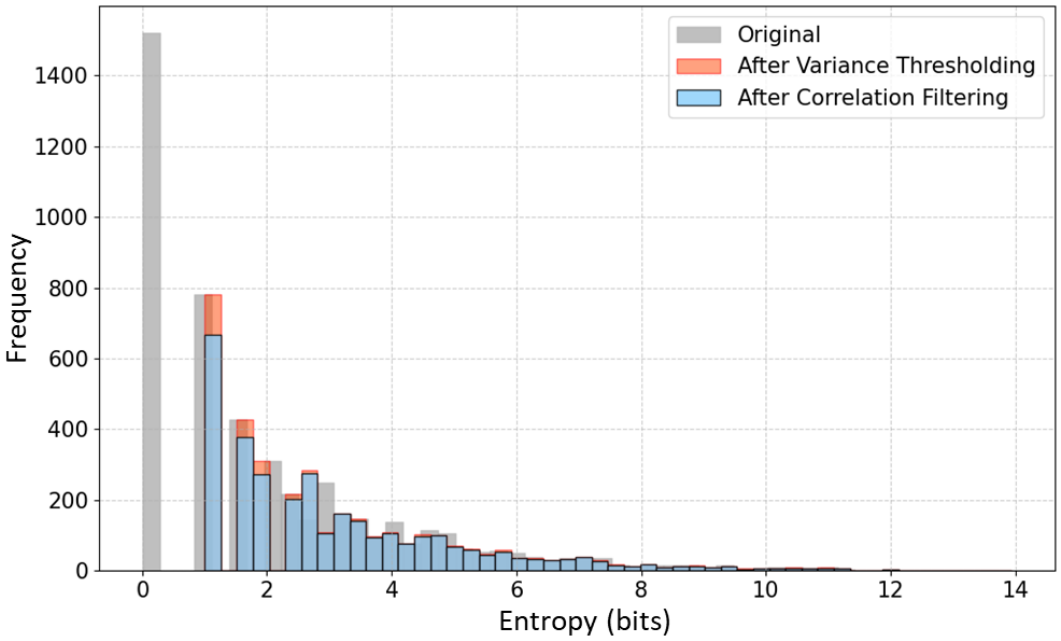

Figure 7: Distribution of entropy across GO terms after the removal of highly correlated features.

Finally, 2,680 features remain. The fraction of zeros in the feature matrix (0.9950) does not decrease significantly.

Next, we assess the performance of this filtering strategy. As shown in Figure 8, most of the removed features exhibit high normalized offspring, high GO proportion, and high information content. These metrics align well with one another and indicate that the majority of the filtered GO terms are highly specific, annotating only a small number of genes and representing biologically rare functions. This poses a potential problem: rare GO terms naturally have low variance because they annotate only a few genes. Filtering them out removes biologically specific information and biases the feature set toward common GO terms.

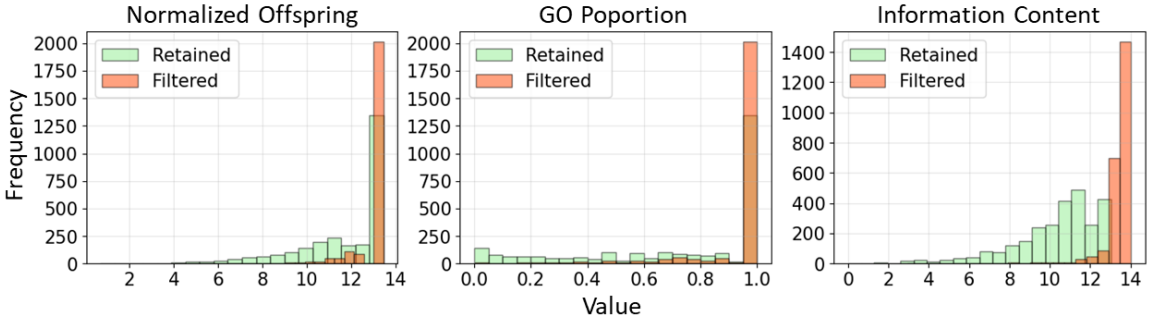

Figure 8: Distributions of the specificity measures of the Combination method.

In Figure 11, we see that the UMAP embedding of the original feature matrix reveals a highly compact arrangement of data points, with limited cluster separation and substantial overlap. In Figure 12, we see that clustering identifies 720 as the optimal number of clusters, though many of these are single-node clusters, as shown in Figure 13. The distribution of cluster sizes follows a scale-free behaviour. In comparison to this, the UMAP after the first feature selection method shows a clearer cluster separation than the one derived from the original feature matrix. The points are more spread out and easier to distinguish, reflecting reduced noise. However, the optimal cluster number increased to 780, with more trivial clusters.

3.2.3 Second feature selection method - Hybrid method 1

We aim to develop a biologically more meaningful feature selection strategy, as the initial method removed all highly specific GO terms. However, GO terms with low specificity are also essential, since they annotate many genes; removing them would produce many zero rows in the feature matrix, which we want to avoid. Therefore, our next approach focuses on retaining both highly specific GO terms and those that annotate a large number of genes.

In Figure 9 we see the frequency distribution showing how many GO terms annotate a given number of genes. Most GO terms annotate a small number of genes (1-10 genes), with the frequency decreasing as the number of annotated genes increases. There are relatively few GO terms that annotate thousands of genes (these would be very general terms near the root in the GO graph). The log-log scale reveals the heavy-tailed nature of this distribution. The mean number of genes annotated by a GO term is 46.5.

The second feature selection method, termed "Hybrid Method 1", combines the two complementary criteria mentioned above. First, we retain features with high Information Content (IC $\geq$ 12). High IC indicates that a GO term is rare and biologically specific, annotating very few genes while representing precise biological processes that may be functionally critical. Figure 8 shows that IC values range from 0 to 14, motivating our

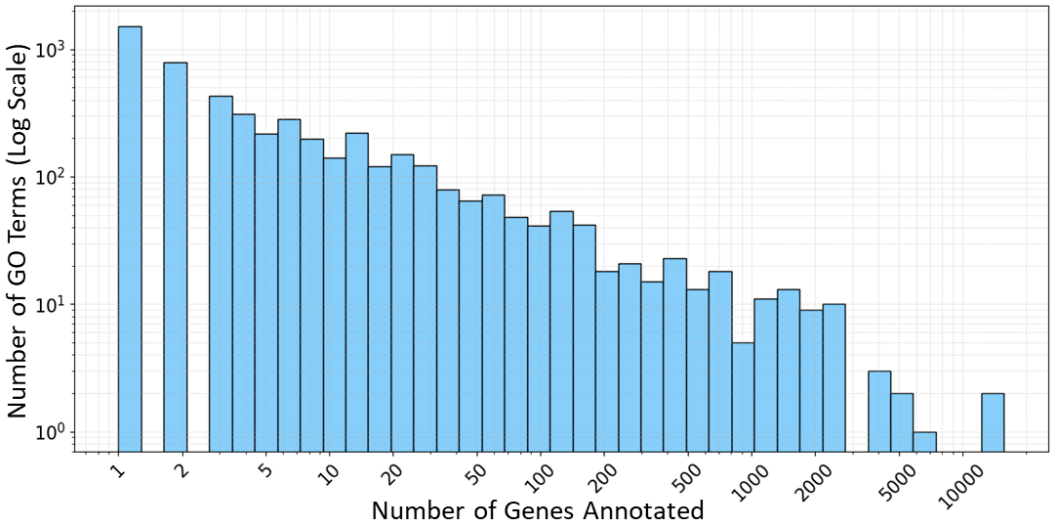

Figure 9: Distribution of gene counts per GO term.

threshold of 12. Second, we retain GO terms that annotate at least 10 genes. This ensures that broadly relevant terms providing essential biological context are preserved, preventing sparse feature vectors where most samples would have zero annotations for overly specific terms alone. Together, these criteria balance specificity and coverage.

This approach retains 4,299 GO terms, achieving a more conservative dimensionality reduction than the first feature selection method.

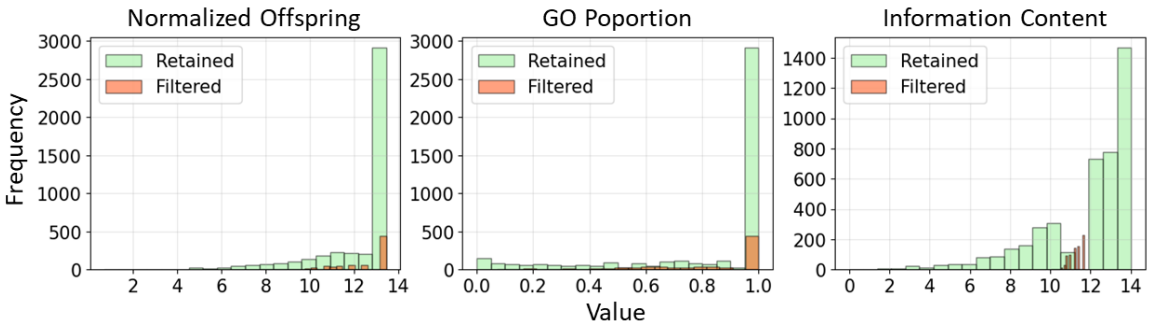

Figure 10: Distributions of the specificity measures of the Hybrid method 1.

As expected, Figure 10 shows that the retained features comprise two distinct groups: highly specific terms with IC ≥ 12 and broadly annotated terms with lower IC values.

In Figure 11, the UMAP visualization shows increased clustering quality compared to the first feature selection method. The projection appears more fragmented, with more separated clusters. The optimal cluster count 620 is a significant reduction from the original 740, see Figure 12. However, in Figure 13, we see that the number of trivial clusters is higher than in the original feature matrix. These results indicate that Hybrid Method 1 improves the data structure but there is still one other feature selection method we want to explore.

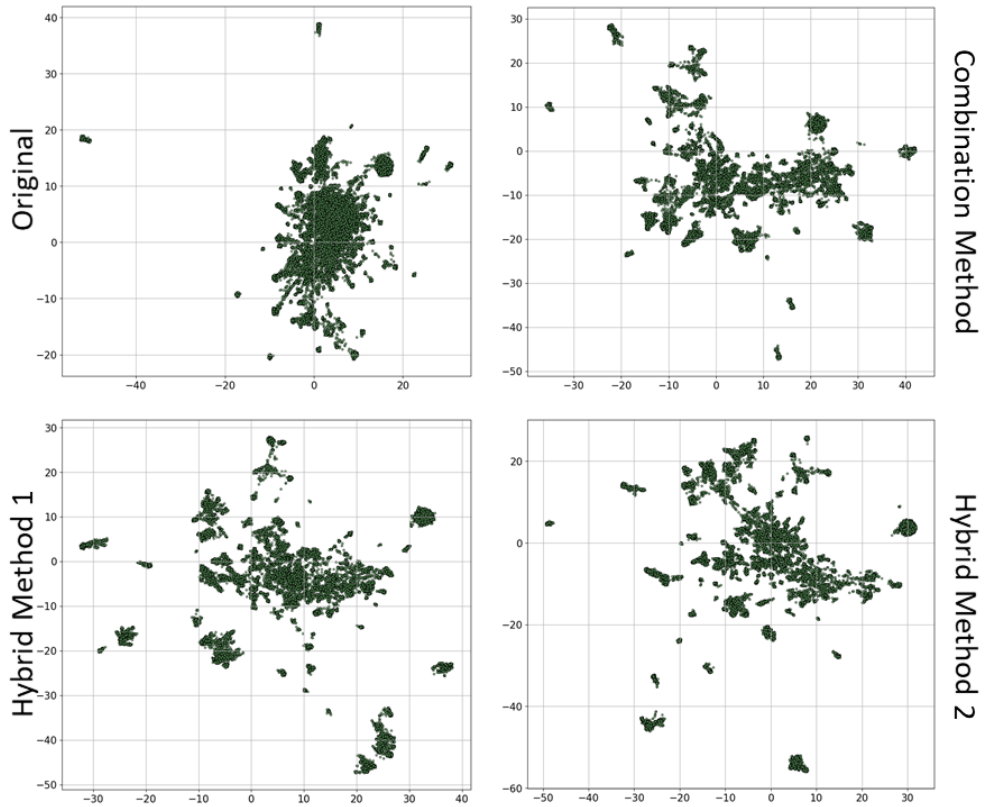

Figure 11: Comparison of the original feature matrix, the feature matrix after the Combination method, the feature matrix after the Hybrid method 1 and the feature matrix after the Hybrid method 2 in terms of their UMAP projection.

3.2.4 Third feature selection method - Hybrid method 2

Since removing low-variance or low entropy features eliminates GO terms with the highest IC values (as demonstrated above), we modify our approach. This method, "Hybrid Method 2", retains:

- GO terms annotating many genes (threshold ≥ 35), and
- GO terms with IC values in the interval $[10, 11)$.

This strategy ensures that: all samples retain non-zero feature representations, classical feature selection principles are satisfied by removing low entropy and low ariance features, and moderately rare, biologically informative terms are preserved.

The resulting feature set contains 1,054 GO terms. This is the most aggressive dimensionality reduction out of the three feature selection methods.

Figure 14 confirms the expected pattern: broadly annotated terms exhibit low IC values and are retained. Unlike Hybrid Method 1, which kept the most specific terms (IC ≥ 12), this approach filters out the highest IC features and instead preserves a subset of moderately specific GO terms.

The UMAP visualizations in Figure 11 demonstrate some improvement. Compared to the original feature matrix, the post-selection projection shows enhanced inter-cluster separation while preserving intra-cluster cohesion. Notably, the clusters remain dense and

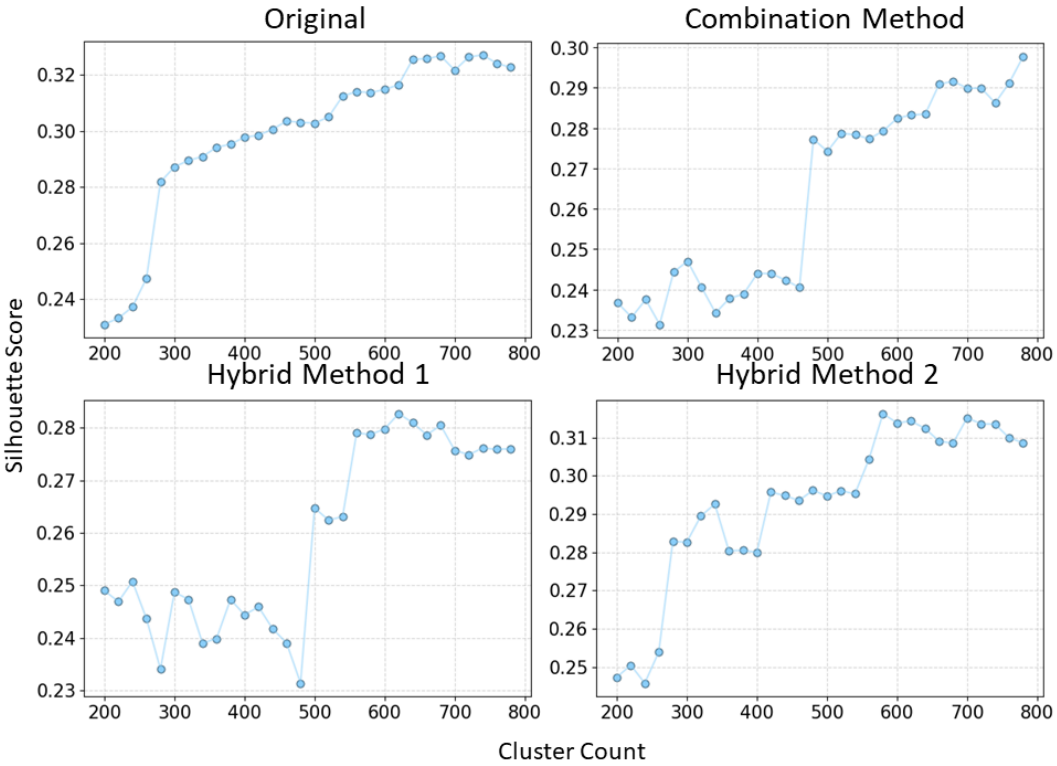

Figure 12: Comparison of the original feature matrix, the feature matrix after the Combination method, the feature matrix after the Hybrid method 1 and the feature matrix after the Hybrid method 2 in terms of the silhouette scores.

well-formed without excessive fragmentation but we see more separation than in the projection after the first and second feature selection method. The optimal number of clusters is 580 which is an even better result than the previous, see Figure 12. We can also see in Figure 13 that the number of trivial clusters is the lowest compared to the other two methods and only a minimal increase from the original.

We select this as our preferred feature selection method because it achieves the best performance according to our assessment criteria while maintaining biological interpretability, significantly reducing the feature space dimension and incorporating principles from classical feature selection approaches.

3.3 Construction of the functional similarity graph

The functional similarity graph is constructed by applying UMAP to the MF feature matrix after the feature selection is done and extracting the resulting k-nearest neighbour graph. This graph serves as one of the two complementary views in the dual-view architecture, capturing the functional relationships between genes. The UMAP is configured with the cosine metric and $k = 50$. Cosine distance has been used in single-cell RNA-seq and other gene expression studies [43] for comparison in sparse, high-dimensional vector spaces, since it focuses on vector direction rather than magnitude.

With 16,585 genes and $k = 50$, each gene connects to approximately 0.3% of the total genes. This provides a connected graph, while preserving local functional structure

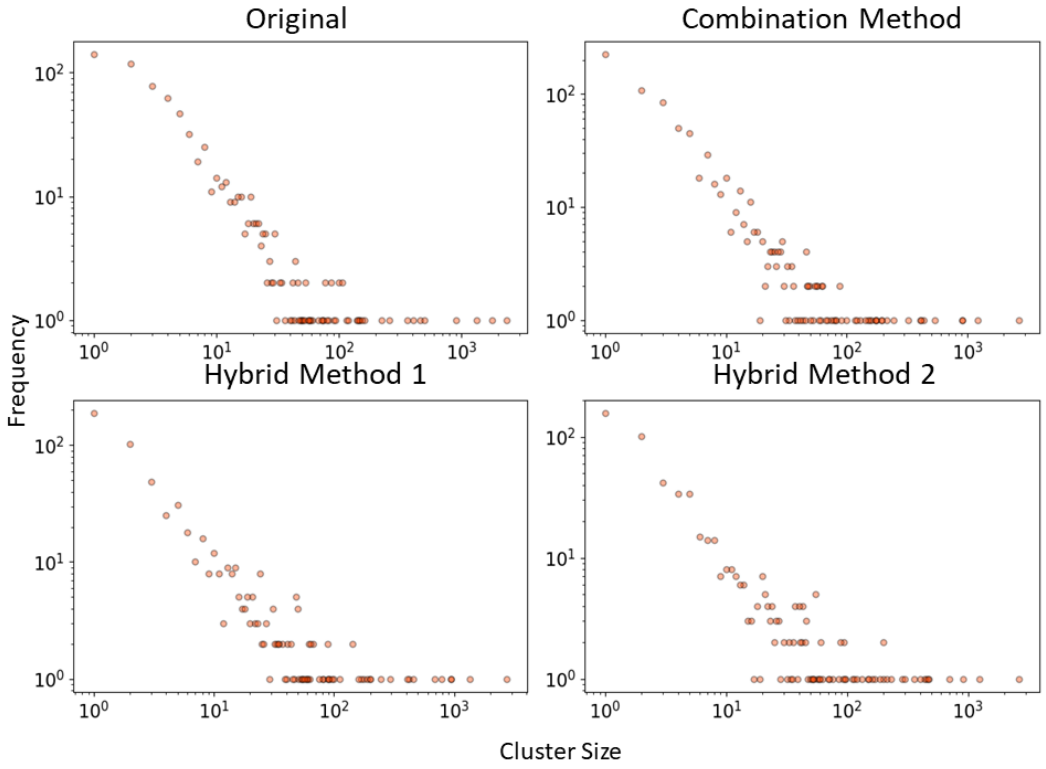

Figure 13: Comparison of the original feature matrix, the feature matrix after the Combination method, the feature matrix after the Hybrid method 1 and the feature matrix after the Hybrid method 2 in terms of their cluster size distribution. "Cluster Size" refers to the number of genes in a given cluster and by "Frequency" we mean the number of clusters that have a given size. Both axes are logarithmic scaled.

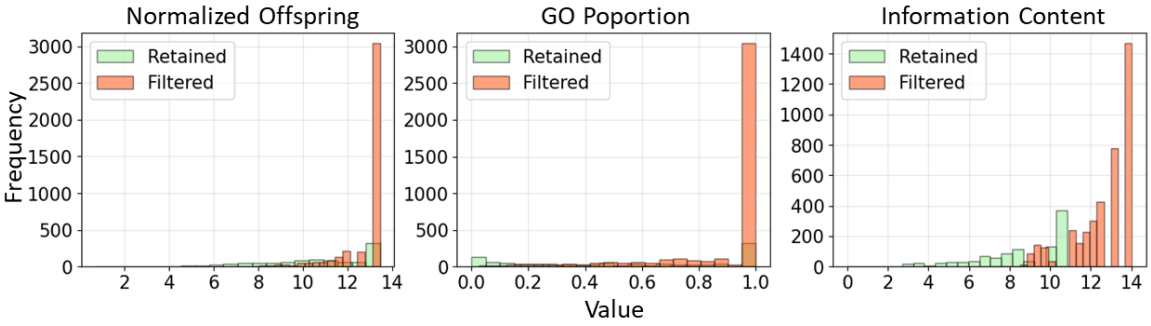

Figure 14: Distributions of the specificity measures of the Hybrid method 2.

through sparsity.

3.4 Feature selection for the BP feature matrix

The BP feature matrix is extremely sparse (0.9962) and imbalanced. The average class imbalance (negative to positive ratio) per GO term is 4,740.7 : 1. If we were to use this matrix for the gene function prediction, the classifier would simply learn that no matter the given GO term, the genes are not annotated by it. Thus, we need to filter the GO terms of this matrix to reduce the imbalance and sparsity. Different methods are applied

to see the impact of filtering on the prediction task.

The most simple way to filter the BP feature matrix is to do frequency filtering, i.e. remove terms that annotate too many genes or annotate only a few genes. The distribution of gene counts per GO term is shown in Figure 15. There are more than a thousand terms that only annotate one gene, and there are a few terms that annotate almost every gene.

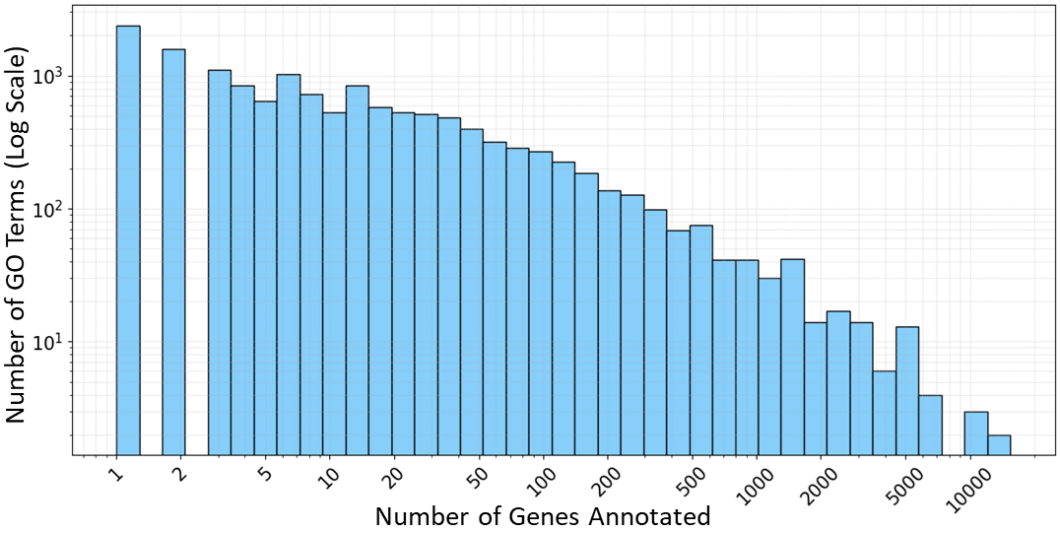

Figure 15: Distribution of gene counts per biological process GO term.

First, terms that annotate less than 200 genes or more than 5,000 are filtered. That leaves 649 terms with a sparsity of 0.953 and an average class imbalance of 37.0 : 1. This filtering strategy is called "Filter 1".

We also want to try a more aggressive version of this simple method, called "Filter 2". As such, we filter the terms that annotate less than 800 or more than 2,000 genes. This results in 122 terms with a sparsity of 0.92 and an average class imbalance of 12.3 : 1.

The frequency filtering does not take the hierarchical relations of the Gene Ontology terms into consideration. For this reason, we also filter based on Information Content (IC). We want terms with low to moderate IC values, so we keep the ones whose IC value falls within the interval $[2, 6)$. This filter, called "Filter 3", leaves 549 terms with a sparsity of 0.951 and an average class imbalance of 32.2 : 1.

We also want to take a combination of these two filtering methods, similarly to how it was done for the MF feature matrix. We keep terms that have an IC value in the interval $[2, 6)$ and annotate at least 800 genes. This results in 157 GO terms with a sparsity of 0.897 and an average class imbalance of 10.6 : 1. This last filtering strategy is called "Filter 4".

These four different variations of the BP feature matrix are then used for the gene function prediction task. To ensure a fair comparison across these four filtering strategies, a common gene set has been identified where each gene is annotated by at least one of the GO terms in each filtered set. This resulted in 14,657 genes available for evaluation.

3.5 Dual-view architectures

We introduce three dual-view encoders. All three combine the protein-protein interaction network's structural information and the functional similarity graph's molecular function Gene Ontology annotation information. Some architectural properties are shared among the encoders. All three encoders have the same inputs:

- the MF feature matrix $X \in \mathbb{R}^{16{,}585 \times 1{,}054}$
- the PPI network with adjacency matrix A_{ppi}, and
- the functional similarity graph with adjacency matrix A_{func}.

The Gene Ontology features are first transformed through a linear projection layer that maps the input dimension 1,054 to a hidden dimension (256 in our implementation):

$$h_0 = \text{ReLU}(W_{\text{proj}}\ X + b_{\text{proj}})$$

where $W_{\text{proj}} \in \mathbb{R}^{256 \times 1{,}054}$ and dropout ($p = 0.1$ in our implementation) is applied for regularization. Dropout [59] is a stochastic regularization technique that randomly removes a fraction of neurons and all its incoming and outgoing connections with probability p during training. This should prevent complex co-adaptations, i.e. increase the robustness of each hidden unit by forcing it to create useful features on its own without relying on other hidden units to correct its mistakes. [59]

The core of the encoder architectures are the two types of graph convolutional network layers; one of them uses the functional similarity graph, the other uses the protein-protein interaction network. The layers have the same stucture:

$$h_l \to \text{GCN}_{\text{graph}}(h_l,\ A_{\text{graph}}) \to \text{BatchNorm} \to \text{ReLU} \to \text{ResCon} \to \text{Dropout} \to h_{l+1}$$

$\text{GCN}_{\text{graph}}$'s structure is as described in section 2.3.

Batch normalization (BatchNorm) [60] is applied after each convolutional layer to address the following problem: the distribution of layer inputs change during training as previous layers' parameters update and thus the layers need to continuously adapt to the new distribution. This change is called "internal covariate shift". By fixing the distribution of each layer's inputs, batch normalization reduces the internal covariate shift and allows for higher learning rates, reduces sensitivity to initialization, and provides a regularization effect. The normalization is performed as:

$$h_l \to \gamma \frac{h_l - \mu_B}{\sigma_B^2 + \varepsilon} + \beta$$

where μ_B and σ_B^2 are the batch mean and variance, ε is a small constant for numerical stability. The simple normalization of each input of a layer may change what the given layer can represent therefore γ and β are introduced as learnable scale and shift parameters. By setting $\gamma = \sqrt{\sigma_B^2}$ and $\beta = \mu_B$, we can recover the original activations. [60]

The inherent problem of oversmoothing, when it comes to graph neural networks, was already mentioned in section 2.3. Unlike traditional convolutional neural networks, where the residual connections (ResCon) address vanishing gradients in very deep networks (50-152 layers) [61], in GCNs even shallow networks of 3-4 layers face oversmoothing [52], so in this case the role of residual connections is to prevent this [62].

$$\mathcal{F}(h_l) \to h_l + \mathcal{F}(h_l)$$

where $\mathcal{F}$ is the residual function, which in our case has the following structure:

$$h_l \rightarrow \text{GCN} \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \rightarrow \mathcal{F}(h_l)$$

The way these graph convolutional layers are combined and how the gene embeddings z are created differ for the three models. However, the very last step of producing the final embeddings is again shared. The gene embeddings z are L2-normalized to unit length:

$$z_{\text{final}} = \frac{z}{||z||_2}.$$

This ensures that similarity comparisons focus on the directional relationships between gene embeddings rather than their magnitude. Moreover, this normalization makes the cosine similarity equivalent to the inner product, which aligns well with the decoder architecture.

3.5.1 Alternating concatenation

First, we adapt the scNET [1] dual-view architecture. The model employs an alternating convolution strategy followed by graph attention mechanisms.

There are four graph convolutional network layers arranged in an alternating pattern between the functional similarity graph and the PPI network, starting with the functional similarity graph. This design enables the model to iteratively refine gene representations by alternating between two complementary views of gene relationships.

Following the alternating convolution, the model applies graph attention networks (GAT) to produce the final gene embeddings.

$$\begin{aligned} z_{\text{ppi}} &= \text{GAT}_{\text{ppi}}(h_4,\ E_{\text{ppi}}) \\ z_{\text{func}} &= \text{GAT}_{\text{func}}(h_4,\ E_{\text{func}}) \\ z &= \alpha \cdot z_{\text{ppi}} + (1-\alpha) \cdot z_{\text{func}} \end{aligned}$$

where E_{ppi} and E_{func} are the edge lists of the PPI network and the functional similarity graph, respectively. The edge lists are used to determine which node pairs to compute attention for. α is a learnable parameter initialized to 0.5 and constrained to $[0, 1]$ through a sigmoid activation. The sigmoid function is defined as:

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

This allows the model to automatically weight the importance of each graph structure. The GAT layers are described in section 2.3, they employ multi-head attention with 4 heads with concatenation of head outputs to produce 128-dimensional embeddings (each head producing 32-dimensional outputs).

3.5.2 Early concatenation

In this next model, the protein-protein interaction network and the functional similarity graph are processed through separate convolutional branches before being merged together. Each branch consists of two graph convolutional layers (the first layer does not contain residual connection due to dimensionality mismatch), and then the outputs are concatenated resulting in h_{concat}. The concatenation is followed by a shared linear layer:

$$h_{\text{concat}} \rightarrow \text{Linear} \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \rightarrow \text{Dropout} \rightarrow h_{\text{final}}$$

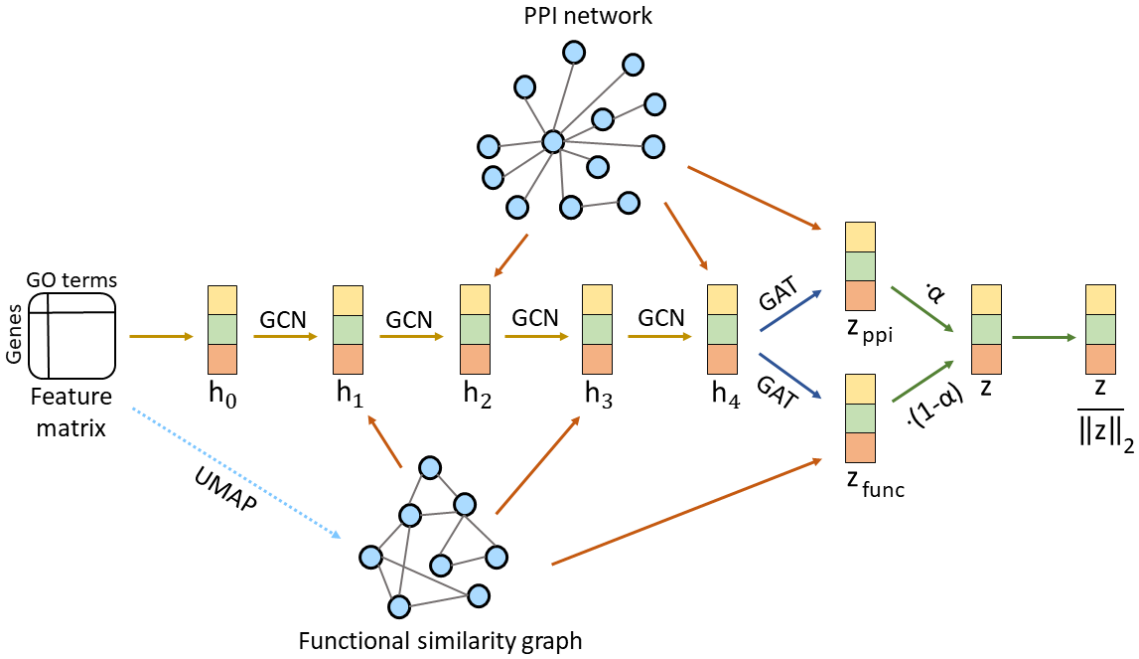

Figure 16: The main steps of the alternating concatenation architecture adapted from scNET [1].

This layer integrates both graph representations before a simple linear projection is applied to h_{final} obtaining the gene embeddings z. By concatenating intermediate representations rather than alternating convolutions, this model allows each graph type to develop specialized features before integration.

3.5.3 Late concatenation

This last dual-view model processes the PPI network and the functional similarity graph through completely independent pipelines until the concatenation that gives the gene embeddings z. Each pipeline consists of two graph convolutional layers. After the second convolutional layer, each branch applies a dedicated graph attention layer to produce separate 64-dimensional embeddings z_{ppi} and z_{func}. These embeddings are concatenated to form the final 128-dimensional representation z.

3.5.4 Multi-layer perceptron decoder

The following multi-layer perceptron decoder is used to reconstruct the original MF feature matrix from the learned embeddings:

$$
\begin{aligned}
z_{\text{final}} &\rightarrow \text{Linear layer with output dim. of 256} \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \rightarrow \text{Dropout} \\
&\rightarrow \text{Linear layer with output dim. of 128} \rightarrow \text{BatchNorm} \rightarrow \text{ReLU} \rightarrow \text{Dropout} \\
&\rightarrow \text{Linear layer with output dim. of 1,054} \rightarrow \text{Sigmoid} \rightarrow X_{\text{recon}}
\end{aligned}
$$

3.5.5 Training procedure

The encoders are trained using a composite loss function that balances three objectives:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{MF}} \cdot \mathcal{L}_{\text{MF}} + \lambda_{\text{PPI}} \cdot \mathcal{L}_{\text{PPI}} + \lambda_{\text{func}} \cdot \mathcal{L}_{\text{func}}$$

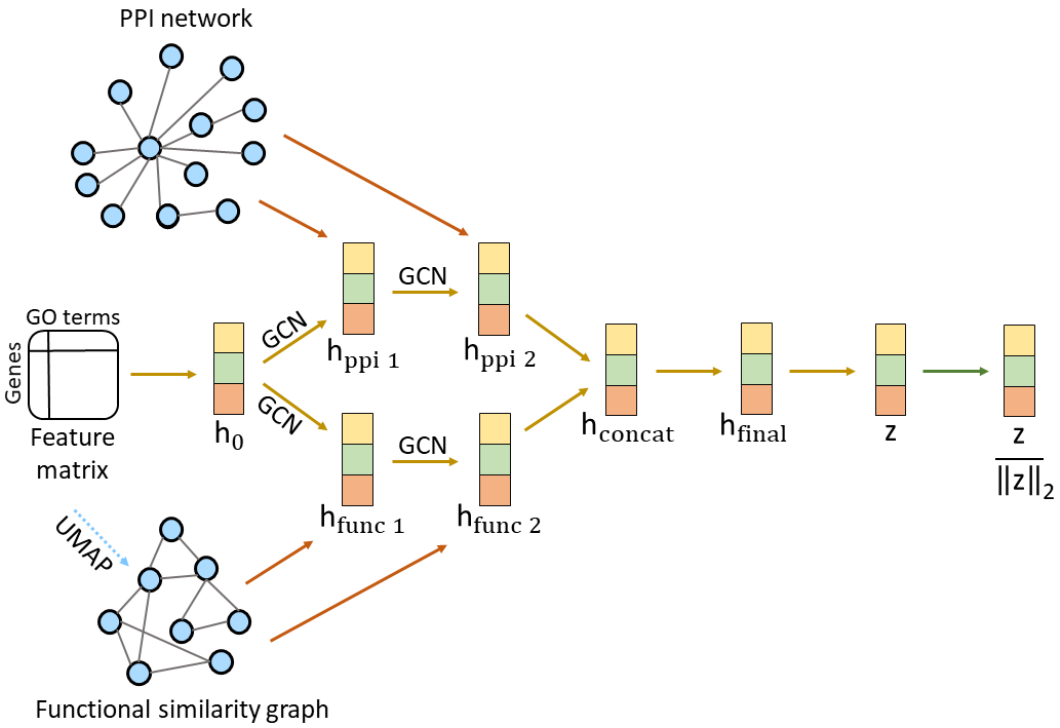

Figure 17: The main steps of the early concatenation architecture.

where

- MF feature matrix loss ($\mathcal{L}_{\text{MF}}$): mean squared error between reconstructed and original MF feature matrix

$$\mathcal{L}_{\text{MF}} = \frac{1}{16{,}585} \|X - X_{\text{recon}}\|_2^2 .$$

- PPI loss ($\mathcal{L}_{\text{PPI}}$): Binary cross-entropy loss for link prediction on the PPI network, using the inner product decoder as described in section 2.3 with the sigmoid function.
- Functional similarity loss ($\mathcal{L}_{\text{func}}$): Binary cross-entropy loss for link prediction on the functional similarity graph, using the same inner product decoder formulation. After the L2-normalization, this inner product is the cosine similarity between gene embeddings, making the decoder interpretable; genes with similar functional roles should have similar embedding directions.

The inner product decoder is trained with balanced positive and negative samples. For each true edge (positive sample), we randomly select one non-existent edge (negative sample). Since our graphs are highly sparse, we sample a subset of non-edges rather than computing loss over all possible node pairs. This reduces computational complexity from $O(16{,}585^2)$ to $O(|E|)$, where $|E|$ is the number of edges in a given graph. The graph reconstruction loss is formulated as:

$$\mathcal{L}_{\text{graph}} = -\left(\sum_{(i,j)\in E^+} \log \sigma(z_i^T z_j) + \sum_{(i,j)\in E^-} \log(1 - \sigma(z_i^T z_j)) \right)$$

where E^+ represents true edges from the graph and E^- is the set of randomly sampled non-existent edges (node pairs with no connection). The first term encourages high

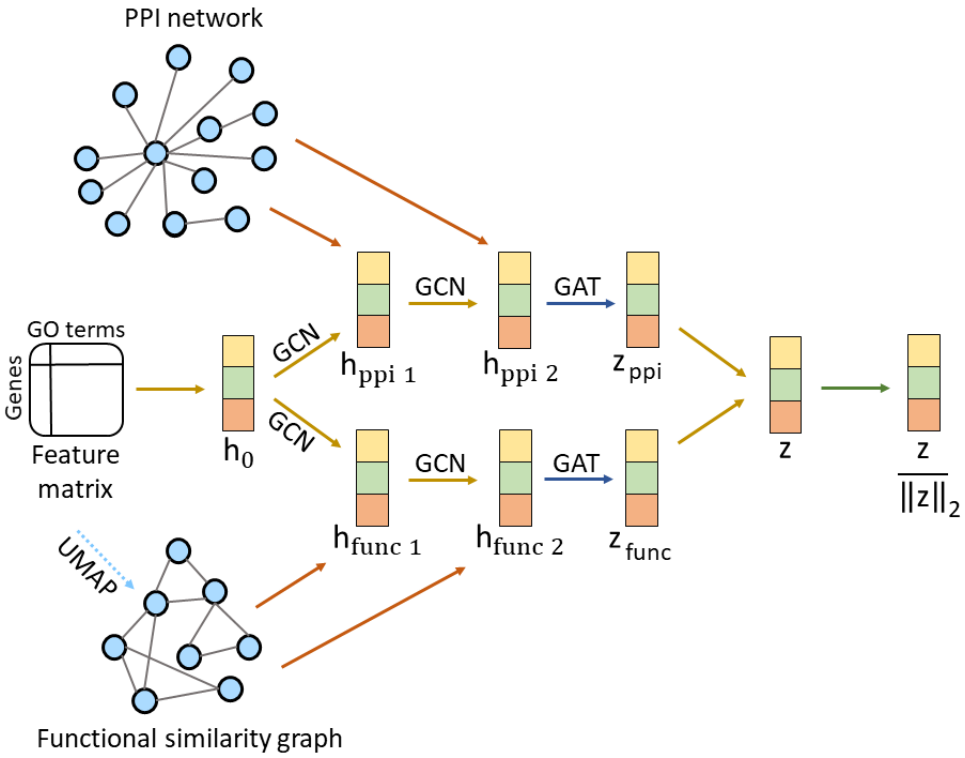

Figure 18: The main steps of the late concatenation architecture.

probabilities for existing edges, while the second term encourages low probabilities for non-existent edges.

To remove potentially spurious functional connections, we implemented a progressive edge pruning strategy for the functional similarity graph. At epochs 20, 40, 60, 80, and 100, we computed edge relevance scores using the inner product of learned embeddings and removed the bottom 10-20% of edges based on these scores. This iterative refinement allows the model to focus on the most informative functional relationships as training progresses.

The training consisted of 100 epochs using the Adam (Adaptive Moment Estimation) optimizer [64]. Adam computes adaptive learning rates for each parameter from estimates of first and second moments of gradients. The optimizer was configured with a learning rate of 10^{-3} and weight decay of 10^{-5}. Weight decay penalizes large parameter values by adding a term proportional to the squared L2-norm of weights to the loss function.

Additionally, PyTorch's ReduceLROnPlateau [63] scheduler with factor = 0.5 and patience = 10 is used. This scheduler monitors the total loss and reduces the learning rate by the specified factor (0.5) when the loss plateaus for more than 10 consecutive epochs. This adaptive learning rate strategy allows the model to take larger steps early in training when far from a minimum, and smaller, more refined steps as it approaches convergence.

Gradient clipping with a maximum norm of 1.0 was applied to prevent the exploding gradient problem. Exploding gradients occur when gradients become excessively large during backpropagation, causing unstable parameter updates that can lead to numerical overflow or divergence. [65] Following the gradient norm clipping method described in

Pascanu et al. [65], the total norm across all parameters is computed:

$$\text{total norm} = \sqrt{\sum_i ||g_i||^2}$$

where g_i represents the gradient of the i-th parameter. If total norm > maximum norm (1.0 in this case), all gradients are scaled by $\frac{\text{maximum norm}}{\text{total norm}}$. This rescaling preserves the direction of the gradient vector.

The models were implemented in PyTorch [63] using the PyTorch Geometric library.

3.5.6 Grid search

The loss function requires balancing three objectives through weighting hyperparameters. λ_{MF}, λ_{PPI}, and λ_{func} are set based on systematic grid search optimization. These weights determine the relative importance of MF feature matrix reconstruction, PPI network structure reconstruction, and functional similarity graph reconstruction in the overall training objective.

The models performances were assessed using a comprehensive set of metrics:

- The primary metric is discrimination ratio. This quantifies how well the model separates functionally related gene pairs from random pairs:

  $$\text{discrimination} = \frac{\mu_{\text{func}}}{|\mu_{\text{random}}| + \varepsilon}$$

  where μ_{func} is the mean cosine similarity between functionally connected genes, μ_{random} is the mean cosine similarity between randomly sampled gene pairs, and $\varepsilon = 10^{-8}$ prevents division by zero. Higher discrimination ratios indicate better capture of biological relationships in the embedding space.

- The reconstruction losses; MF feature matrix loss, PPI loss, functional similarity loss and the total loss.

A composite score is defined for ranking configurations:

$$\text{composite score} = -10 \cdot \text{discrimination} + \mathcal{L}_{\text{total}} + 2 \cdot \mathcal{L}_{\text{MF}}.$$

The maximization of the discrimination ratio has priority (we have a negative coefficient for minimization) while maintaining low reconstruction error is also important. The coefficients reflect the relative importance: discrimination is weighted ten times more than total loss.

The complete grid search evaluated 8 configurations for each dual-view architecture with the following lambdas being tested: $\lambda_{\text{MF}} \in \{1.0,\ 2.0\}$, $\lambda_{\text{PPI}} \in \{0.5,\ 1.0\}$ and $\lambda_{\text{func}} \in \{0.5,\ 1.0\}$. Each configuration was trained for 30 epochs. The epoch count was reduced for efficiency. The best configuration:

- Alternating concatenation: $\lambda_{\text{MF}} = 1.0$, $\lambda_{\text{PPI}} = 0.5$ and $\lambda_{\text{func}} = 0.5$.
- Early concatenation: $\lambda_{\text{MF}} = 2.0$, $\lambda_{\text{PPI}} = 0.5$ and $\lambda_{\text{func}} = 0.5$.
- Late concatenation: $\lambda_{\text{MF}} = 1.0$, $\lambda_{\text{PPI}} = 0.5$ and $\lambda_{\text{func}} = 1.0$.

The detailed grid search results can be seen in Figures 19, 20 and 21.

λ_{MF}	λ_{PPI}	λ_{func}	Run Time in Hours	$\mathcal{L}_{MF}$	$\mathcal{L}_{PPI}$	$\mathcal{L}_{func}$	$\mathcal{L}_{total}$	Discrimination	Composite Score
1.0	0.5	0.5	1.11	0.085	1.399	1.313	1.441	1.91	-17.51
1.0	0.5	1.0	1.11	0.086	1.470	1.413	2.234	1.43	-11.85
1.0	1.0	0.5	1.14	0.077	1.542	1.518	2.378	1.16	-9.05
1.0	1.0	1.0	1.11	0.081	1.404	1.337	2.822	1.73	-14.27
2.0	0.5	0.5	0.98	0.063	1.425	1.363	1.520	1.67	-15.09
2.0	0.5	1.0	1.02	0.085	1.397	1.317	2.186	1.92	-16.83
2.0	1.0	0.5	1.04	0.093	1.537	1.518	2.482	1.17	-8.99
2.0	1.0	1.0	1.07	0.094	1.419	1.349	2.956	1.69	-13.71

Figure 19: Detailed grid search results for the alternating concatenation model.

λ_{MF}	λ_{PPI}	λ_{func}	Run Time in Hours	$\mathcal{L}_{MF}$	$\mathcal{L}_{PPI}$	$\mathcal{L}_{func}$	$\mathcal{L}_{total}$	Discrimination	Composite Score
1.0	0.5	0.5	1.63	0.111	1.330	1.156	1.354	4.12	-39.60
1.0	0.5	1.0	1.76	0.100	1.348	1.208	1.982	2.94	-27.22
1.0	1.0	0.5	1.71	0.106	1.322	1.228	2.042	2.72	-24.95
1.0	1.0	1.0	1.85	0.104	1.344	1.217	2.665	2.82	-25.28
2.0	0.5	0.5	1.90	0.096	1.297	1.138	1.41	4.93	-47.67
2.0	0.5	1.0	1.89	0.098	1.351	1.159	2.031	4.19	-39.62
2.0	1.0	0.5	1.91	0.106	1.384	1.262	2.227	2.37	-21.22
2.0	1.0	1.0	1.82	0.103	1.401	1.288	2.895	2.14	-18.27

Figure 20: Detailed grid search results for the early concatenation model.

3.6 Single-view architectures and the baseline model

To validate the effectiveness of these dual-view architectures and assess the individual contributions of different graph components, we introduce two single-view models and a baseline model for comparison. Each model is designed to isolate specific architectural choices in order to evaluate their impact on embedding quality and performance on gene function prediction.

All of these models have the same embedding dimension of 128 and network depth as the dual-view model. They were trained for 100 epochs, using identical optimization settings (Adam optimizer, learning rate = 10^{-3}, weight decay = 10^{-5} and gradient clipping at maximum norm 1.0), batch normalization and residual connections.

3.6.1 PPI-only model

The PPI-only model encodes exclusively the protein-protein interaction network, ignoring functional similarity information. This model consists of four graph convolutional layers, followed by a single graph attention layer. Each layer has the same structure as described above. The model is trained using two objectives; the MF feature matrix reconstruction ($\lambda_{\text{MF}} = 1.0$) and the PPI reconstruction (with $\lambda_{\text{PPI}} = 1.0$) using the inner product decoder.

This model enables us to see if the PPI structural information alone is sufficient for learning meaningful gene representations.

λ_{MF}	λ_{PPI}	λ_{func}	Run Time in Hours	$\mathcal{L}_{MF}$	$\mathcal{L}_{PPI}$	$\mathcal{L}_{func}$	$\mathcal{L}_{total}$	Discrimination	Composite Score
1.0	0.5	0.5	1.69	0.077	1.445	1.364	1.482	1.58	-14.15
1.0	0.5	1.0	1.7	0.085	1.416	1.279	2.072	2.05	-18.23
1.0	1.0	0.5	1.77	0.085	1.460	1.411	2.251	1.41	-11.73
1.0	1.0	1.0	1.83	0.11	1.454	1.385	2.949	1.52	-11.99
2.0	0.5	0.5	1.8	0.093	1.423	1.320	1.558	1.83	-16.56
2.0	0.5	1.0	1.78	0.098	1.430	1.310	2.221	1.88	-16.36
2.0	1.0	0.5	1.78	0.09	1.483	1.443	2.385	1.34	-10.88
2.0	1.0	1.0	1.75	0.098	1.413	1.314	2.923	1.86	-15.45

Figure 21: Detailed grid search results for the late concatenation model.

3.6.2 Functional similarity-only model

The functional similarity-only model operates exclusively on the functional similarity graph, without the protein-protein interaction network. Structurally, it is identical to the PPI-only baseline model but applied to the functional edges. Here, we use the progressive graph pruning with the same scheduler as with our dual-view models. The training objectives include the MF feature matrix reconstruction ($\lambda_{\text{MF}} = 1.0$) and the functional similarity reconstruction ($\lambda_{\text{func}} = 1.0$).

This model tests whether Gene Ontology similarity alone can produce competitive embeddings for gene function prediction.

3.6.3 MF feature-only model

Lastly, this model does not employ graph convolutional layers, it only operates using four linear layers:

$$h_l \to \text{Linear} \to \text{BatchNorm} \to \text{ReLU} \to \text{Dropout} \to h_{l+1}.$$

This architecture treats genes as independent entities, ignoring both PPI and functional relationships gained for the graphs. The model is trained exclusively with the MF feature matrix reconstruction objective ($\lambda_{\text{MF}} = 1.0$), as graph reconstruction losses are not applicable ($\lambda_{\text{PPI}} = \lambda_{\text{func}} = 0.0$).

With this model, we establish the baseline performance that is achievable without introducing any graph structure.

3.7 Evaluation process and metrics

The quality of the learned gene embeddings is evaluated by using them on the task of gene function prediction. This is a supervised multi-label classification problem that measures how well the embeddings capture biologically meaningful relationships.

As the primary classifier model, we use logistic regression due to its simplicity, interpretability, and established use in embedding evaluation tasks [36, 37]. A strong performance indicates that the functional relationships are linearly separable in the embedding space. For multi-label prediction, independent binary classifiers are trained for each functional category using one-vs-rest strategy. Each classifier predicts the probability

that a gene has a certain Gene Ontology term based solely on its embedding representation. L2-regularization (inverse regularization strength C = 1.0), and limited iterations (maximum iteration = 500) are used to ensure convergence while preventing overfitting on the training embeddings. The hyperparameter "class_weight" is set to "balanced". This reweighs the loss function so that the contribution of each class is inversely proportional to its frequency in the training set, thus giving more weight to the minority, in our case positive, class.

The inputs are a gene embedding matrix and a binary GO term label matrix, they are iterated over k folds. Within each fold, a binary logistic regression classifier is trained for each GO term. We use stratified k-fold cross-validation with iterative stratification for multi-label data [66] to ensure that rare GO term annotations were represented in both training and test folds.

Multiple metrics are used to capture different aspects of prediction quality, implemented from [68]:

- Macro-averaged F1 score: this is computed as the unweighted average of F1 scores across all GO terms. This way all GO terms are treated equally regardless of frequency, ensuring that performance on rare terms contribute equally to the overall score. The F1 score is calculated the following way:

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

  where TP is the number of true positives, FP is the number of false negatives, and FN is the number of false positives.

- Micro-averaged F1 score: this is computed by summing true positives, false positives and false negatives across all GO terms before computing the F1 score.

- Precision and recall [69]: precision reveals the fraction of the predicted GO terms that are actually correct and recall is the fraction of annotated GO terms that are recovered. The formula for Precision is:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} \ + \ \mathrm{FP}}$$

  and the formula for recall is:

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} \ + \ \mathrm{FN}}.$$

- Area Under Receiver Operating Characteristic (AUROC) [69]: the ROC curve plots the True Positive Rate against the False Positive Rate. The True Positive Rate is equal to the recall and the False Positive Rate is defined as

$$\text{False Positive Rate} = \frac{\mathrm{FP}}{\mathrm{TN} \ + \ \mathrm{FP}}.$$

  The AUROC is then the area under this ROC curve, this value lies between 0 and 1. A higher AUROC value implies better classification ability.

4 Results

In this chapter, we present the structural differences between the functional similarity graph and the protein-protein interaction network, then the main results, the gene function prediction and the topological analysis of the embedding spaces, are introduced.

4.1 Comparing the structural properties of the functional similarity graph and the PPI network

We are interested in comparing the structural properties of two networks: the functional similarity graph constructed from the MF feature matrix and the protein-protein interaction network.

We use quantitative measures to compare the following properties of the two graphs: degree distribution, clustering coefficient, average shortest path length, and modularity.

1. Degree distribution

   In order to determine how similar the degree distributions of the two graph are, we use the Jensen-Shannon divergence [21]. This is defined in the following way:

   $$JS_\pi(p_1, p_2) = H(\pi_1 p_1 + \pi_2 p_2) - \pi_1 H(p_1) - \pi_2 H(p_2),$$

   where $\pi_1, \pi_2 \geq 0$ are the weights fulfilling $\pi_1 + \pi_2 = 1$, p_1 and p_2 are two probability distributions and H is the Shannon entropy defined as

   $$H(p_1) = -\int p_1 \log p_1.$$

   H is a concave function therefore using Jensen's inequality, we get that $JS_\pi(p_1, p_2)$ is nonnegative and is equal to zero if $p_1 = p_2$. In the article, they derive that $0 \leq JS_\pi(p_1, p_2) \leq 1$. The Jensen-Shannon distance is then the square root of the Jensen-Shannon divergence.

   In our case, we take the weights to be equal i.e. $\pi_1 = \frac{1}{2} = \pi_2$ and compute the Jensen-Shannon distance of the two degree distributions and get the value 0.57. This value is quite high, implying that the two distributions are significantly different. We know that the protein-protein interaction network is scale-free, meaning that the degree distribution has a power-law tail [2]. A consequence of this is that we have many proteins that have a low degree and we have a few proteins with very high degree (hubs). In contrast to this, the functional similarity matrix is in its core a k-nearest neighbour graph, so by construction all nodes have similar degrees (approximately k), which results in a degree distribution that is narrower and more uniform with far fewer hubs.

2. Clustering coefficient

   The clustering coefficient for a node v is calculated according to [22], as

   $$C(v) = \frac{2t(v)}{deg(v) \cdot (deg(v) - 1)}$$

   if $deg(v) \geq 2$, otherwise $C(v) = 0$. Here $t(v)$ denotes the number of pairs of neighbours of v that are neighbours i.e. the number of triangles in the graph that contain v. Then the clustering coefficient of a graph is defined as

   $$C = \frac{1}{|V|} \sum_{v \in V} C(v),$$

where V is the set of all nodes in the graph and $|V|$ denotes the number of nodes in the graph.

The clustering coefficient of the functional similarity graph is 0.53, and of the protein-protein interaction network is 0.08. The high clustering coefficient of the functional similarity graph is not surprising, since it is a k-nearest neighbour graph where neighbours of a point tend to be connected to each other, forming lots of triangles. Whereas the low clustering coefficient of the protein-protein interaction graph is the result of the fact that most genes interact with a few hubs, so triangles are not common.

3. Average shortest path length

   The average shortest path length refers to the average of the shortest possible paths between all possible pairs of nodes in a network. A path is a route that runs along the edges of the graph, its length representing the number of edges the path contains. The shortest path between nodes i and j is the path with fewest number of edges. [2]

   The average shortest path length of the functional similarity graph is 4.4 and of the protein-protein interaction network is 3.1. Even though the functional similarity graph has significantly higher clustering coefficient, its average shortest path length is greater. As a result of the k-nearest neighbour structure, in this graph there are tighter clusters, but the distance between the clusters is longer. In contrast, the hub-like structure of the protein-protein interaction network means that many low-degree nodes connect to a few hubs, leading to shorter distances.

4. Modularity

   In a randomly wired network, connections are distributed uniformly, so no meaningful communities or dense subgraphs are expected to form. This leads to the hypothesis, that such networks lack inherent community structure. By comparing the link density of a suspected community to the density expected in a randomly rewired version of the same network, we can test whether the observed structure is real or just due to chance. Systematic deviations from randomness motivate the definition of modularity, a measure of how well a partition captures true community structure. Modularity enables comparing different partitions, and maximizing it provides a powerful method for community detection. [2]

   Modularity of a certain partition is defined by Barabási [2] the following way:

$$M = \sum_{c=1}^{n_c} \left[\frac{L_c}{L} - \left(\frac{k_c}{2L} \right)^2 \right],$$

   where n_c is the number of communities of the given partition, L_c is the total number of edges, k_c is the total degree of the nodes in the given community and L is the total number of edges in the graph.

   The higher the modularity of a given partition, the better is the corresponding community structure [2]. Since the graphs have a high number of nodes, we use the Louvain community detection algorithm. The Louvain algorithm detects communities by repeatedly improving modularity in two steps. Step one is about local optimization: each node is moved to the neighbouring community that yields the largest modularity gain, producing an initial set of communities. In step two, these communities are compressed into single nodes, creating a smaller graph (with possible

self-loops). These two steps are repeated until no further modularity improvement is possible. [2]

When we calculate the modularity of the partitions that we get as a result of applying the Louvain algorithm to our two graphs, we get 0.89 for the functional similarity graph and 0.39 for the protein-protein interaction network. Looking deeper into the communities, we see that in the functional similarity graph the minimum number of nodes in a community is 22, the median is 256.0, mean is 312.9 and the largest community contains 1934 nodes. This tells us that all communities are reasonably large and they are balanced in size. Communities are dense internally with few inter-community edges. This explains the high modularity score. In contrast, in the protein-protein interaction network the minimum number of nodes in a community is 1, the median is 13.0, mean is 637.9 and the largest community contains 4099 nodes. All this means that there are many trivial (one-node) or small communities and only a few very large communities. This is in alignment with the scale-free behaviour of the graph.

All four methods of comparison show the same picture; the functional similarity graph has mostly larger communities, with a lot of triangles and most nodes have a similar degree, whereas the protein-protein interaction network has a scale-free degree distribution, with a few hubs and a few larger communities. These significant structural differences further motivate the integration of both information.

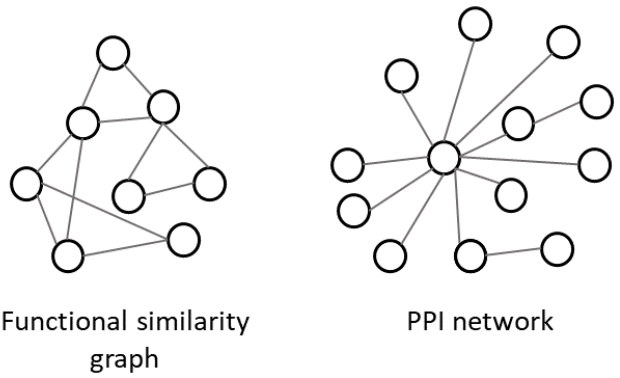

Figure 22: Visualization of the structural differences of the functional similarity graph and the PPI network.

4.2 Gene function prediction

We evaluate the six different embedding architectures for gene function prediction across the four filtering strategies of the biological process feature matrix. We report macro-averaged and micro-averaged F1 scores, precision, recall, and AUROC. As our primary metric we take the macro-averaged F1 scores, the AUROC and the improvement (Δ) relative to the MF feature-only architecture.

Figure 24 presents the comprehensive performance evaluation on macro-averaged F1 scores and AUROC. Several clear patterns emerge from these results.

Late concatenation is the top-performing architecture, achieving the highest marco-averaged F1 scores in all of four filtering strategies. This model reached a maximum

	PPI network	Functional similarity graph
Jensen–Shannon distance	0.57	
Clustering coefficient	0.08	0.53
Average shortest path length	3.1	4.4
Modularity	0.39	0.89
Min. community size	1	22
Median community size	13.0	256.0
Mean community size	637.9	312.9
Max. community size	4099	1934

Figure 23: Summary of the structural differences of the functional similarity graph and the PPI network.

F1 score of 0.355 on Filter 4, representing the best overall performance observed in our experiments.

The alternating concatenation architecture performed comparable to the late concatenation architecture, achieving F1 scores within $0.001 - 0.006$ of the top model across all filters. Through alternating between PPI and functional similarity graph convolutions, this model demonstrated that more sophisticated graph integration strategies can achieve competitive performance however, the added complexity did not yield substantial improvements over the simpler late concatenation approach.

The ranking of architectures remained consistent across all four filtering strategies, suggesting robust differences in architectural capability rather than filter-specific optimization.

The percentage improvements over baseline showed in inverse relationship with absolute performance: Filter 1 and 3, which had lower absolute F1 scores, showed larger relative improvements from the introduction of the graph structure $(23 - 29\%)$, while Filter 2 and 4 showed smaller improvements $(17 - 24\%)$. This pattern is expected, as there is less room for improvement when baseline performance is already higher.

	Filter 1			Filter 2			Filter 3			Filter 4		
Model	F1	AUROC	Δ	F1	AUROC	Δ	F1	AUROC	Δ	F1	AUROC	Δ
Alternating concat.	0.193	0.789	+28.7%	0.309	0.775	+23.1%	0.203	0.786	+27.7%	0.349	0.774	+22%
Early concat.	0.181	0.775	+20.7%	0.295	0.761	+17.5%	0.191	0.772	+20.1%	0.336	0.761	+17.5%
Late concat.	0.194	0.788	+29.3%	0.314	0.775	+25.1%	0.205	0.785	+28.9%	0.355	0.775	+24.1%
PPI-only	0.171	0.763	+14%	0.277	0.75	+10.4%	0.179	0.76	+12.6%	0.317	0.749	+10.8%
Functional sim.-only	0.188	0.772	+25.3%	0.3	0.76	+19.5%	0.197	0.769	+23.9%	0.341	0.761	+19.2%
MF feature-only	0.15	0.693	---	0.251	0.685	---	0.159	0.692	---	0.286	0.686	---

Figure 24: Biological process prediction results. The filter numbering is consistent with the order of the filters introduced in section 3.4. With F1 scores, we refer to the macro-averaged F1 scores.

A critical finding from our experiments is the substantial performance gain provided by incorporating graph structure. The MF feature-only architecture, which uses only the MF feature matrix without any additional graph information, used as a baseline model consistently underperformed all graph-based models by considerable margins. Across all four filtering strategies, graph-based models achieved $10-29\%$ relative improvement in F1 scores over the baseline (Δ column in Figure 24). This improvement was most pronounced in Filters 1 and 3, where graph-based models showed $23-29\%$ gains, and remained substantial even in the more selective Filters 2 and 4 with $17-24\%$ improvement. The consistency of these improvements across filtering strategies provides strong evidence that both the PPI network and the functional similarity graph encode biologically relevant information not captured by the molecular function annotations alone.

Moreover, the AUROC differences between graph-based and the baseline models were similarly substantial, with graph-based models achieving $0.75-0.79$ AUROC compared to 0.69 for the MF feature-only model across filters.

These two metrics indicate that graph structure improves both the ranking quality and the classification performance of gene function prediction.

Another consistent finding is the superior performance of the functional similarity graph over the protein-protein interaction network when used in isolation. Across all four filtering strategies, the functional similarity-only architecture outperformed the PPI-only architecture by $0.017-0.024$ in F1 scores, see Figure 25 for details.

This pattern is particularly interesting given the structural differences between these graphs. As seen in section 4.1, the functional similarity graph exhibits higher modularity and clustering coefficient compared to the PPI network. The higher modularity of the functional similarity graph suggests that it by nature partitions the genes into communities that align well with biological process annotations, which may explain why it has a superior predictive performance compared to the PPI network.

	Functional sim.-only F1	PPI-only F1	Absolute Δ	Relative Δ
Filter 1	0.188	0.171	+0.017	+9.9%
Filter 2	0.3	0.277	+0.023	+8.3%
Filter 3	0.197	0.179	+0.018	+10.1%
Filter 4	0.341	0.317	+0.024	+7.6%

Figure 25: Comparison of the functional similarity-only and the PPI-only model in terms of macro-averaged F1 scores.

The fact that two out of the three dual-view architectures (late and alternating concatenation) outperform single-graph models indicates that both views contribute valuable and non-redundant information for this task.

In the dual-view architectures, the timing of the merging of the two graph representations proved to be an important design choice. The independent processing of each graph before merging the final embeddings without any further processing, consistently outperformed early concatenation, which merges graph representations at an intermediate layer. This

advantage ranged from 0.013 to 0.019 points in F1 scores.

Additionally, the alternating concatenation architecture, which explicitly alternates between the graphs without merging them, demonstrates a strong performance alongside the late concatenation architecture. This suggests that merging representations at an intermediate stage, may force premature integration that loses graph-specific features.

The choice of the filtering strategies had a substantial impact on the absolute performance metrics, though it did not alter the relative rankings of the architectures. Figure 26 summarizes the key characteristics and performance ranges for each filtering strategy.

	# GO terms	Best F1	Worst F1	Range
Filter 1	649	0.194	0.171	0.023
Filter 2	122	0.314	0.277	0.037
Filter 3	549	0.205	0.179	0.026
Filter 4	157	0.355	0.317	0.038

Figure 26: Comparison of the filtering strategies and their impact on the gene function prediction performance.

Filter 2 and 4 impose stricter size constrains on the biological process annotation terms and yield substantially higher F1 scores ($0.277 - 0.355$) compared to the broader Filter 1 and 3 ($0.171 - 0.205$). Filter 4, which combines information content constrains with a minimum coverage threshold, achieved the highest performance across all architectures, with the top model reaching an F1 score of 0.355.

This performance advantage likely stems from the improved class balance, as Filter 4 has an average class imbalance of $10.6 : 1$, the lowest of all filtering methods.

While F1 scores varied considerably across the four filters ($0.171 - 0.355$), AUROC values remained more compressed ($0.749 - 0.789$). Filter 1 and 3 showed slightly higher AUROC values ($0.76 - 0.789$) compared to Filter 2 and 4 ($0.75 - 0.775$). This suggests that the more selective filters produce Gene Ontology terms that are easier to classify (higher F1 scores) but make the ranking more challenging (slightly lower AUROC).

Figure 27 presents the precision and recall values of the six architectures across the four filtering strategies. While recall remained consistently high across all graph-based models ($0.675 - 0.706$), precision varied more substantially ($0.107 - 0.253$). This pattern indicates that all models adopt a recall-prioritized strategy, successfully identifying $68 - 71\%$ of true annotations however, simultaneously producing substantial numbers of false positives.

The more selective filtering methods (Filter 2 and 4) achieve substantially higher precision ($0.177 - 0.253$) compared to the broader Filter 1 and 3 ($0.107 - 0.132$). Meanwhile, recall remains constant ($0.675 - 0.706$) across all filters. This finding provides an explanation for the superior performance of Filter 2 and 4: by excluding very rare and very common Gene Ontology terms, these filters create more balanced prediction tasks that enable models to achieve better precision without sacrificing recall.

From a biological perspective, this characteristic may be advantageous: high recall ensures a comprehensive list of candidates for experimental validation, where false positives

	Filter 1		Filter 2		Filter 3		Filter 4	
Model	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Alternating concat.	0.124	0.706	0.204	0.698	0.129	0.704	0.245	0.695
Early concat.	0.116	0.703	0.193	0.688	0.120	0.701	0.234	0.685
Late concat.	0.127	0.703	0.211	0.692	0.132	0.702	0.253	0.688
PPI-only	0.107	0.677	0.177	0.676	0.111	0.675	0.215	0.676
Functional sim.-only	0.123	0.682	0.201	0.681	0.128	0.682	0.244	0.676
MF feature-only	0.106	0.541	0.180	0.535	0.111	0.540	0.23	0.519

Figure 27: Summary of the precision and recall values of the different architectures across the filtering strategies.

can be discarded through additional evidence, while false negatives represent missed functional associations that cannot be recovered. However, the low precision (25% or below for top models) indicates that approximately 75% of positive predictions would require experimental validation which may limit practical utility for high-throughput applications without additional filtering strategies.

The comprehensive evaluation of six embedding architectures across four filtering methods establish the late concatenation architecture with selective filtering (Filter 4) as the optimal configuration for gene function prediction, achieving an F1 score if 0.355, which is a 24% improvement over the MF feature-only baseline approach.

4.3 Embedding space characterization

We analyze the learned embeddings from a topological perspective.

The embeddings are trained on molecular function (MF) features and graph structures, while biological process (BP) terms are used as prediction targets. This separation is essential to avoid data leakage, as predicting the same GO terms used to construct the embeddings would not constitute a valid evaluation of the model's generalization capability. This inherent design constraint is reflected in the topological characteristics.

Looking at Figure 28, we see that the late concatenation model exhibits highly discrete, island-like clusters with clear separation. We can observe approximately 20-30 distinct, well-defined functional modules. There are minimal bridging between the clusters. This structure creates maximally discriminative features for the classifier.

The alternating concatenation model shows a continuous, densely-packed blob-like topology with internal texture but no discrete separation. There are smooth transitions between functional regions rather than hard boundaries, the genes are highly interconnected. This topology preserves rich relational information while maintaining global coherence.

The early concatenation architecture shows an intermediate topology between late and alternating concatenation. Some cluster structure is visible but it is less pronounced that we see with late concatenation. There is moderate separation with some bridging

structures.

The functional similarity-only model displays extreme fragmentation, there are dozens of small, isolated tiny island-like clusters scattered across space. There is a lack of global organization, however this model outperforms the PPI-only architecture, suggesting that despite resulting in an over-partitioned structure, the functional similarity graph captures meaningful relationships.

The PPI-only architecture comes with a single, large, undifferentiated structure. There are almost no visible cluster boundaries or functional organization.

The baseline model demonstrates moderate fragmentation with scattered medium-sized clusters. It has more structure than the PPI-only architecture, but it is less organized than the late concatenation model.

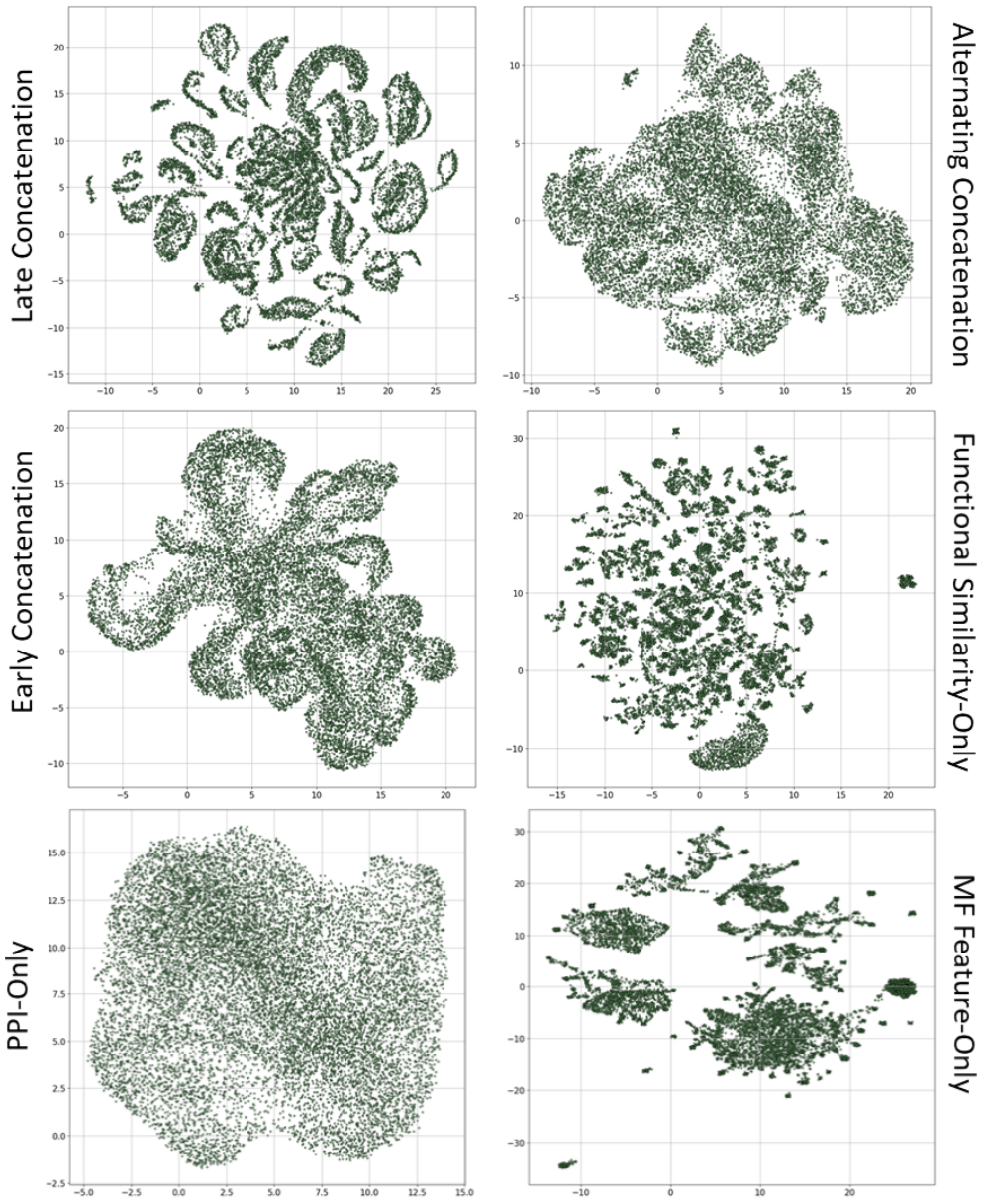

Figure 28: Comparison of the six embedding architectures in terms of their UMAP projection.

The two architectures with the best performance (late and alternating concatenation) represent opposite ends of a spectrum. While the late concatenation model maximizes discrete separation, the alternating concatenation focuses on continuous connectivity. The performances of these models suggest that there are multiple valid approaches to organizing gene embeddings, but they must be executed well. The early concatenation architecture demonstrates that half-measures perform worse.

The single-graph approaches fail in opposite ways. The functional similarity-only model shows good local resolution but poor global coherence and the PPI-only architecture has a good global coverage but poor local resolution.

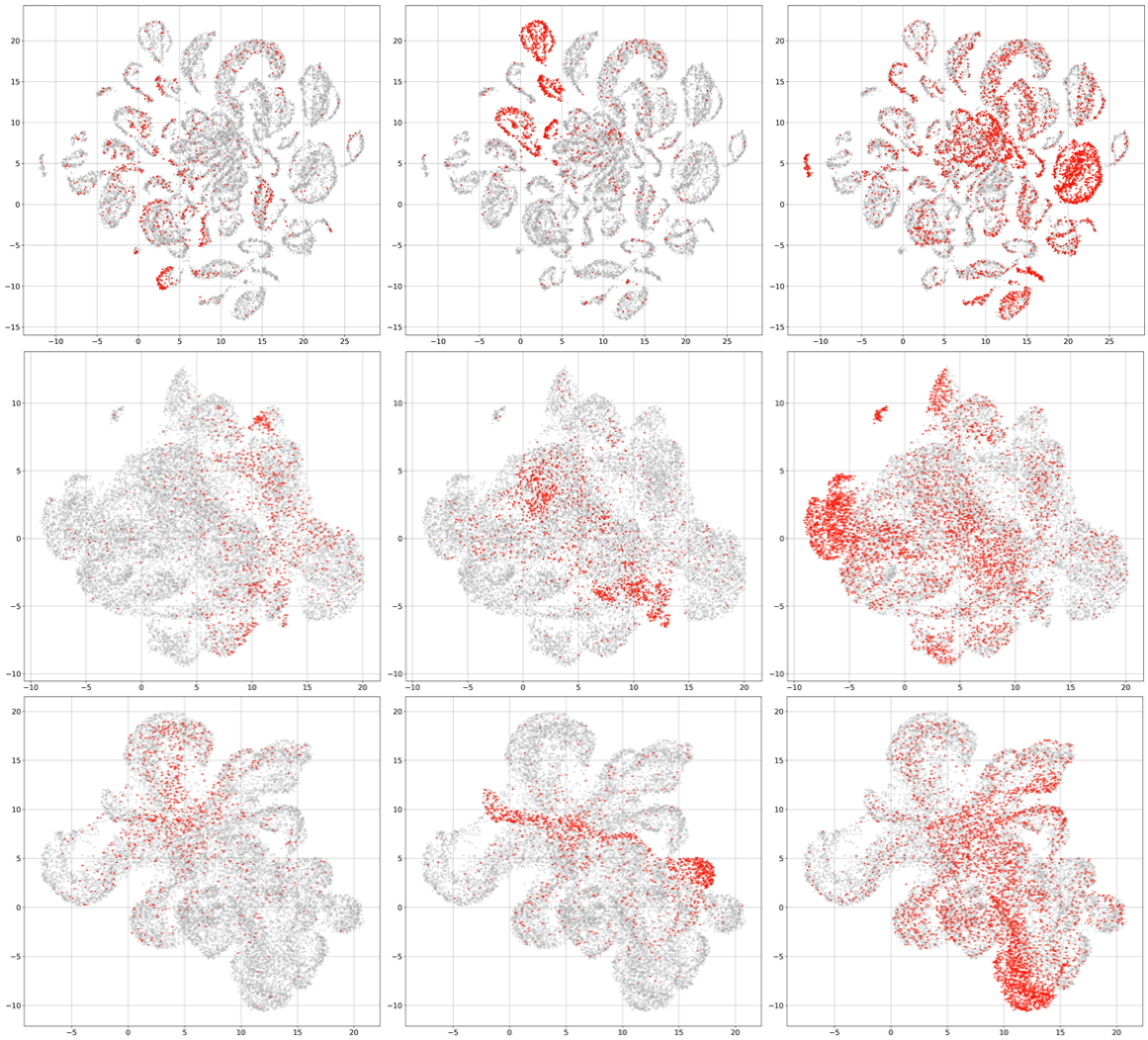

Figure 29: In each UMAP projection, genes annotated with the same biological process GO term are shown in orange. The three GO terms in column-wise order are: GO:0006974 (annotating 802 genes), GO:0006508 (annotating 1201 genes), and GO:0051234 (annotating 3589 genes). The architectures in row-wise order are: late concatenation, alternating concatenation, and early concatenation.

To examine the relationship between embedding topology and prediction performance, we analyze three BP Gene Ontology terms from Filter 4 spanning the annotation frequency spectrum:

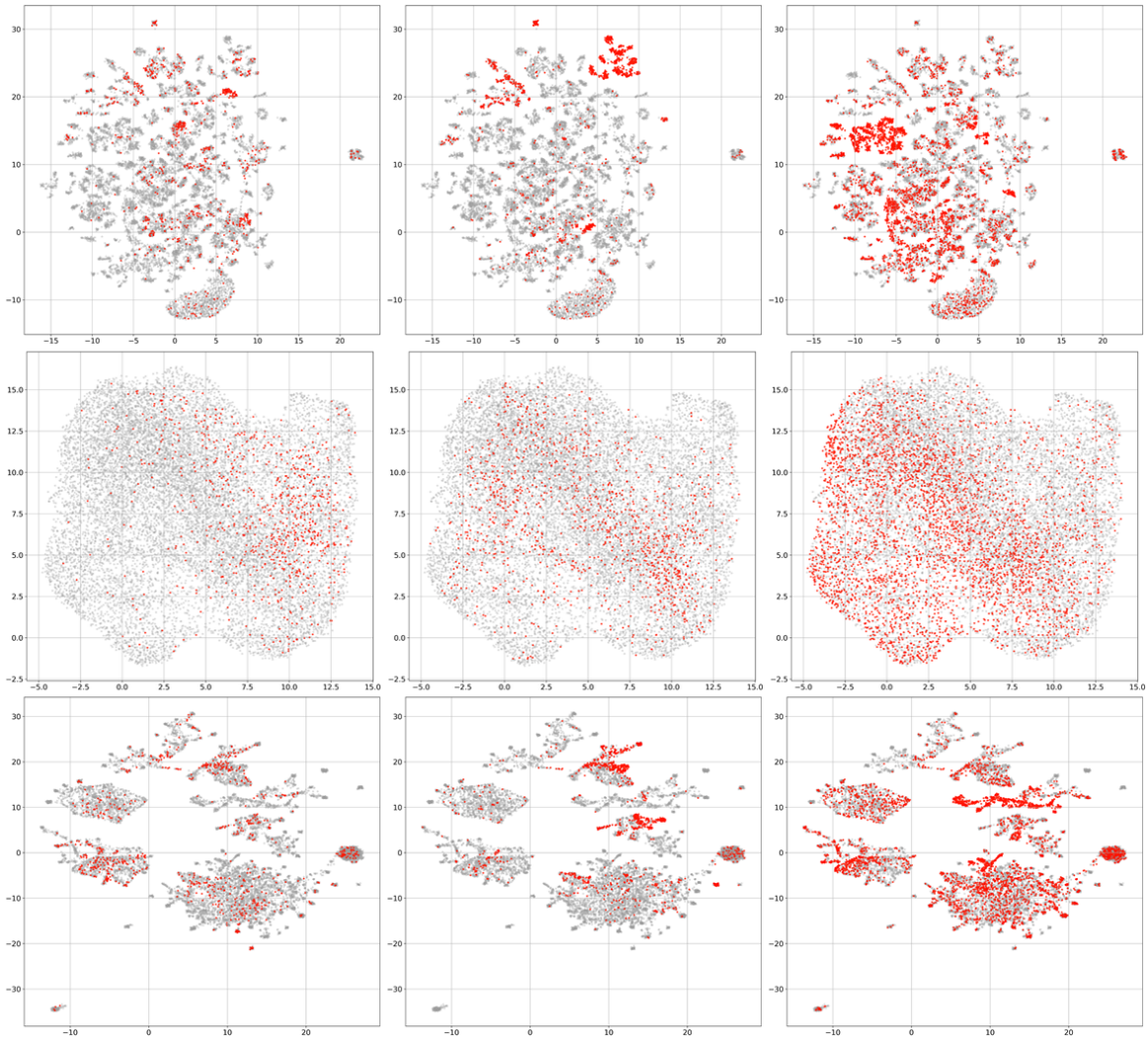

Figure 30: In each UMAP projection, genes annotated with the same biological process GO term are shown in orange. The three GO terms in column-wise order are: GO:0006974 (annotating 802 genes), GO:0006508 (annotating 1201 genes), and GO:0051234 (annotating 3589 genes). The architectures in row-wise order are: functional similarity-only, PPI-only, and MF feature-only.

- a rare BP term: GO:0006974 which stands for "DNA damage response" and annotates 802 genes,
- a BP term with medium frequency: GO:0006508, representing "proteolysis" annotating 1201 genes,
- a common BP term: GO:0051234, meaning "establishment of localization" annotating 3589 genes.

In Figures 29 and 30, we see that the biological process annotation distribution in embedding space varies with both architecture and annotation frequency. In Figure 31, the precision, recall and F1 scores of the six architectures are summarized for these three BP terms.

We can see that that late concatenation architecture achieves the best F1 score for all three GO terms, confirming its superior performance regardless of annotation frequency.

This consistency validates the topological advantage of discrete cluster formation, in contrast to the continuous connectivity focus of the alternating concatenation model.

All models seem to struggle when it comes to the rare BP term, the best F1 score achieved is only 0.246. The precision is very low across all architectures ($0.099 - 0.147$), while the recall is relatively preserved ($0.514 - 0.801$). This demonstrates the recall bias even for rare terms. The UMAP projections show that in the late concatenation architecture annotations concentrate in 2-3 discrete clusters. In the alternating concatenation model the annotations are more dispersed but there is some regional concentration. This architecture achieves the highest recall on this term, which corresponds well with the continuous structure of the embeddings. However, this lack of boundaries is also the reason for the low precision. In the PPI-only model genes annotated with this rare BP term are scattered uniformly without any apparent organization. The annotated genes are highly fragmented across many isolated clusters in the functional similarity-only model.

	GO:0006974 (annotating 802 genes)			GO:000650 (annotating 1201 genes)			GO:0051234 (annotating 3589 genes)		
Model	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Alternating concat.	0.136	0.801	0.233	0.384	0.780	0.515	0.488	0.695	0.573
Early concat.	0.128	0.774	0.219	0.396	0.793	0.528	0.482	0.656	0.555
Late concat.	0.147	0.758	0.246	0.596	0.755	0.666	0.524	0.664	0.586
PPI-only	0.145	0.772	0.244	0.216	0.742	0.335	0.457	0.670	0.543
Functional sim.-only	0.121	0.671	0.206	0.517	0.729	0.605	0.543	0.617	0.578
MF feature-only	0.099	0.514	0.166	0.368	0.575	0.449	0.490	0.380	0.428

Figure 31: Comparing the precision-recall trade-offs for specific GO terms.

With the medium BP term we see a substantial performance improvement, the best F1 score is 0.666. The late concatenation models shows an exceptionally high precision (0.596) compared to the other architectures. There is a wider performance gap between the top two models (late concatenation and functional similarity-only) and the rest, suggesting that this medium term benefits the most from the more discrete cluster structure. It could be that this specific GO term corresponds to a specific MF module. In the UMAP projections, the patterns we see are similar to the ones with the rare BP term.

All architectures demonstrate a moderate performance when it comes to the common BP term, F1 scores ranging from 0.428 to 0.586. The performances of the architectures are closer together, the functional similarity-only and the alternating concatenation models perform nearly as well as the late concatenation. In the UMAP projections, all architectures show widespread distribution, which is to be expected given the high annotation frequency. The annotations in the late concatenation model cover many clusters but not all. In the embedding created by the PPI only model, we see a nearly uniform coverage all across. Annotations are present in the majority of the fragmented clusters of the functional similarity-only embeddings. We can see that when a BP term has such a high frequency, topology matters less.

The topological analysis of the architectures revealed that the late concatenation model's superiority is consistent across the three term frequencies, which validates the more discrete cluster structure. Single-graph approaches show clear topological limitations, suggesting that graph integration is indeed essential.

5 Conclusion

This thesis investigated how different strategies for integrating protein-protein interaction networks and functional similarity graphs affect gene function prediction performance. Through comprehensive evaluation of six embedding architectures across four biological process Gene Ontology term filtering methods, we established several key findings that advance our understanding of graph-based function prediction.

Processing PPI and functional similarity graphs through independent branches before merging final embeddings outperformed both the more complex alternating concatenation approach and simpler early concatenation strategy. This finding suggests that allowing each graph to develop specialized representations captures complementary information more effectively. The consistent superiority of the late concatenation architecture across all four filtering strategies demonstrates that this advantage is a robust property of the architecture rather than an artefact of specific GO term selection.

Even the weakest graph-based model (the PPI-only) substantially outperformed the MF feature-only baseline, validating the fundamental premise that biological functions emerge from the context of molecular networks rather than individual gene properties.

Our UMAP-derived functional similarity graph consistently outperformed the PPI network when used in isolation. This suggests that annotation patterns can provide stronger signals for function prediction than physical protein interactions alone.

All graph-based models maintained high recall while differing substantially in precision. Some Gene Ontology filtering methods that exclude rare and very common terms improved precision, explaining why these filters contribute to better overall performance.

The UMAP projections of the architectures demonstrate that the different integration strategies produce significantly different topologies from the same input data. We saw that neither graph alone can produce effective topology, their certain combination (in late and alternating concatenation architectures) creates organized and predictive feature spaces. However, it is important to note that the topologies organize around molecular function annotations and graph structure, requiring the classifier to learn complex MF-to-BP mappings, not just simple pattern matching.

This work examined only two graph types: the protein-protein interaction network and the UMAP-derived functional similarity graph. Biological systems involve numerous other relationships and expanding to additional graph types could further improve performance.

By the gene function prediction task, the classifier used default decision threshold when converting predicted probabilities to binary predictions. Threshold optimization could improve precision-recall trade-off for specific applications.

The predictive performance of the architectures is influenced by the frequency of the given BP term. By developing different integration strategies for rare and common terms, precision could be improved for the rare GO terms.

In conclusion, we have demonstrated that graph neural network architectures combining protein-protein interaction networks with a functional similarity graph substantially improve gene function prediction compared to feature-only or single-graph approaches. This thesis provides both practical tools for immediate application and a contribution to the continuous methodological innovations in computational gene function prediction.

Bibliography

[1] R. Sheinin, R. Sharan, and A. Madi. scNET: learning context-specific gene and cell embeddings by integrating single-cell gene expression data with protein–protein interactions. *Nat Methods*, 22:708–716, 2025.

[2] A.-L. Barabási. Network Science. *Cambridge University Press*, 2016.

[3] M. Zitnik, M. M. Li, A. Wells, K. Glass, D. M. Gysi, A. Krishnan, T. M. Murali, P. Radivojac, S. Roy, A. Baudot, S. Bozdag, D. Z. Chen, L. Cowen, K. Devkota, A. Gitter, S. J. C. Gosline, P. Gu, P. H. Guzzi, H. Huang, M. Jiang, Z. N. Kesimoglu, M. Koyuturk, J. Ma, A. R. Pico, N. Pržulj, T. M. Przytycka, B. J. Raphael, A. Ritz, R. Sharan, Y. Shen, M. Singh, D. K. Slonim, H. Tong, X. H. Yang, B.-J. Yoon, H. Yu, and T. Milenković. Current and future directions in network biology. *Bioinformatics Advances*, 4(1):vbae099, 2024.

[4] A.-L. Barabási, N. Gulbahce, and J. Loscalzo. Network medicine: a network-based approach to human disease. *Nat Rev Genet*, 12(1):56–68, Jan. 2011.

[5] The Gene Ontology Consortium, S. A. Aleksander, J. Balhoff, S. Carbon, J. M. Cherry, H. J. Drabkin, D. Ebert, M. Feuermann, P. Gaudet, N. L. Harris, D. P. Hill, R. Lee, H. Mi, S. Moxon, C. J. Mungall, A. Muruganugan, T. Mushayahama, P. W. Sternberg, P. D. Thomas, K. Van Auken, J. Ramsey, D. A. Siegele, R. L. Chisholm, P. Fey, M. C. Aspromonte, M. V. Nugnes, F. Quaglia, S. Tosatto, M. Giglio, S. Nadendla, G. Antonazzo, H. Attrill, G. dos Santos, S. Marygold, V. Strelets, C. J. Tabone, J. Thurmond, P. Zhou, S. H. Ahmed, P. Asanitthong, D. L. Buitrago, M. N. Erdol, M. C. Gage, M. A. Kadhum, K. Y. C. Li, M. Long, A. Michalak, A. Pesala, A. Pritazahra, R. Su, K. E. Thurlow, R. C. Lovering, C. Logie, S. Oliferenko, J. Blake, K. Christie, L. Corbani, M. E. Dolan, H. J. Drabkin, D. P. Hill, L. Ni, D. Sitnikov, C. Smith, A. Cuzick, J. Seager, L. Cooper, J. Elser, P. Jaiswal, P. Gupta, P. Jaiswal, S. Naithani, M. Lera-Ramirez, K. Rutherford, V. Wood, J. L. De Pons, M. R. Dwinell, G. T. Hayman, M. L. Kaldunski, A. E. Kwitek, S. J. F. Laulederkind, M. A. Tutaj, M. Vedi, S.-J. Wang, P. D'Eustachio, L. Aimo, K. Axelsen, A. Bridge, N. Hyka-Nouspikel, A. Morgat, S. A. Aleksander, J. M. Cherry, S. R. Engel, K. Karra, S. R. Miyasato, R. S. Nash, M. S. Skrzypek, S. Weng, E. D. Wong, E. Bakker, T. Z. Berardini, L. Reiser, A. Auchincloss, K. Axelsen, G. Argoud-Puy, M.-C. Blatter, E. Boutet, L. Breuza, A. Bridge, C. Casals-Casas, E. Coudert, A. Estreicher, M. L. Famiglietti, M. Feuermann, A. Gos, N. Gruaz-Gumowski, C. Hulo, N. Hyka-Nouspikel, F. Jungo, P. Le Mercier, D. Lieberherr, P. Masson, A. Morgat, I. Pedruzzi, L. Pourcel, S. Poux, C. Rivoire, S. Sundaram, A. Bateman, E. Bowler-Barnett, H. Bye-A-Jee, P. Denny, A. Ignatchenko, R. Ishtiaq, A. Lock, Y. Lussi, M. Magrane, M. J. Martin, S. Orchard, P. Raposo, E. Speretta, N. Tyagi, K. Warner, R. Zaru, A. D. Diehl, R. Lee, J. Chan, S. Diamantakis, D. Raciti, M. Zarowiecki, M. Fisher, C. James-Zorn, V. Ponferrada, A. Zorn, S. Ramachandran, L. Ruzicka, and M. Westerfield. The Gene Ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031, May 2023.

[6] The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47(D1):D330–D338, Jan. 2019.

[7] L. Cowen, T. Ideker, B. J. Raphael, and R. Sharan. Network propagation: a universal amplifier of genetic associations. *Nat Rev Genet*, 18(9):551–562, Sept. 2017.

[8] P. I. Wang, S. Hwang, R. P. Kincaid, A. Sriram, T. McDermott, C. V. Greenblatt, A. Emili, P. Agarwal, E. J. Weinstein, and R. Marcotte. RIDDLE: reflective diffusion and local extension reveal functional associations for unannotated gene sets via proximity in a gene network. *Genome Biol*, 13(12):R125, 2012.

[9] R. Elyanow, B. Dumitrascu, B. E. Engelhardt, and B. J. Raphael. netNMF-sc: leveraging gene-gene interactions for imputation and dimensionality reduction in single-cell expression analysis. *Genome Res*, 30(2):195–204, Feb. 2020.

[10] H. Li, X. Xiao, X. Wu, L. Ye, and G. Ji. scLINE: A multi-network integration framework based on network embedding for representation of single-cell RNA-seq data. *Journal of Biomedical Informatics*, 122:103899, 2021.

[11] M. M. Li, Y. Huang, M. Sumathipala, M. Q. Liang, E. Valdeolivas, A. N. Ananthakrishnan, H. Liao, V. Margulis, L. T. Tran, D. Reker, and M. Zitnik. Contextual AI models for single-cell protein biology. *Nat Methods*, 21:1546–1557, 2024.

[12] E. Fadhal, E. C. Mwambene, and J. Gamieldien. Modelling human protein interaction networks as metric spaces has potential in disease research and drug target discovery. *BMC Syst Biol*, 8:68, 2014.

[13] E. L. Huttlin, R. J. Bruckner, J. A. Paulo, J. R. Cannon, L. Ting, K. Baltier, G. Colby, F. Gebreab, M. P. Gygi, H. Parzen, J. Szpyt, S. Tam, G. Zarraga, L. Pontano-Vaites, S. Swarup, A. E. White, D. K. Schweppe, R. Rad, B. K. Erickson, R. A. Obar, K. G. Guruharsha, K. Li, S. Artavanis-Tsakonas, S. P. Gygi, and J. W. Harper. Architecture of the human interactome defines protein communities and disease networks. *Nature*, 545(7655):505–509, May 2017.

[14] K. Luck, D. K. Kim, L. Lambourne, K. Spirohn, B. E. Begg, W. Bian, R. Brignall, T. Cafarelli, F. J. Campos-Laborie, B. Charloteaux, D. Choi, A. G. Coté, M. Daley, S. Deimling, A. Desbuleux, A. Dricot, M. Gebbia, M. F. Hardy, N. Kishore, J. J. Knapp, I. A. Kovács, I. Lemmens, M. W. Mee, J. C. Mellor, C. Pollis, C. Pons, A. D. Richardson, S. Schlabach, B. Teeking, A. Yadav, M. Babor, D. Balcha, O. Basha, C. Bowman-Colin, S. F. Chin, S. G. Choi, C. Colabella, G. Coppin, C. D'Amata, D. De Ridder, S. De Rouck, M. Duran-Frigola, H. Ennajdaoui, F. Goebels, L. Goehring, A. Gopal, G. Haddad, E. Hatchi, M. Helmy, Y. Jacob, Y. Kassa, S. Landini, R. Li, N. van Lieshout, A. MacWilliams, D. Markey, J. N. Paulson, S. Rangarajan, J. Rasla, A. Rayhan, T. Rolland, A. San-Miguel, Y. Shen, D. Sheykhkarimli, G. M. Sheynkman, E. Simonovsky, M. Taşan, A. Tejeda, V. Tropepe, J. C. Twizere, Y. Wang, R. J. Weatheritt, J. Weile, Y. Xia, X. Yang, E. Yeger-Lotem, Q. Zhong, P. Aloy, G. D. Bader, J. De Las Rivas, S. Gaudet, T. Hao, J. Rak, J. Tavernier, D. E. Hill, M. Vidal, F. P. Roth, and M. A. Calderwood. A reference map of the human binary protein interactome. *Nature*, 580(7803):402–408, Apr. 2020.

[15] GENCODE. GENCODE Release 36 Statistics. *GENCODE*, 2020. Available at: https://www.gencodegenes.org/human/stats_36.html. Accessed 2025 Nov 19.

[16] R. Oughtred, C. Stark, B. J. Breitkreutz, J. Rust, L. Boucher, C. Chang, N. Kolas, L. O'Donnell, G. Leung, R. McAdam, F. Zhang, S. Dolma, A. Willems, J. Coulombe-Huntington, A. Chatr-Aryamontri, K. Dolinski, and M. Tyers. The BioGRID interaction database: 2019 update. *Nucleic Acids Res*, 47(D1):D529–D541, Jan. 2019.

[17] N. del Toro, A. Shrivastava, E. Ragueneau, B. Meldal, C. Combe, E. Barrera, L. Perfetto, K. How, P. Ratan, G. Shirodkar, O. Lu, B. Mészáros, X. Watkins, S. Pundir, L. Licata, M. Iannuccelli, M. Pellegrini, M. J. Martin, S. Panni, M. Duesbury, S. D. Vallet, J. Rappsilber, S. Ricard-Blum, G. Cesareni, L. Salwinski, S. Orchard, P. Porras, K. Panneerselvam, and H. Hermjakob. The IntAct database: efficient access to fine-grained molecular interaction data. *Nucleic Acids Research*, 50(D1):D648–D653, Jan. 2022.

[18] B. Louie, S. Bergen, R. Higdon, and E. Kolker. Quantifying protein function specificity in the gene ontology. *Stand Genomic Sci*, 2(2):238–244, Mar. 2010.

[19] L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. *Wiley*, 1990.

[20] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[21] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Trans Inf Theory*, 37(1):145–151, 1991.

[22] J. J. Aguilar-Alarcón, J. C. Hernández-Gómez, and J. Romero-Valencia. The Clustering Coefficient for Graph Products. *Axioms*, 12(10):968, 2023.

[23] H. Jeong, S. Mason, A.-L. Barabási, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001.

[24] Y. Zhao, J. Wang, J. Chen, X. Zhang, M. Guo, and G. Yu. A Literature Review of Gene Function Prediction by Modeling Gene Ontology. *Front Genet*, 11:400, Apr. 2020.

[25] Q. Chen, Y. Li, K. Tan, Y. Qiao, S. Pan, T. Jiang, and Y.-P. P. Chen. *Network-based methods for gene function prediction. Briefings in Functional Genomics*, 20(4):249–257, July 2021.

[26] S. Y. Rhee and M. Mutwil. Towards revealing the functions of all genes in plants. *Trends in Plant Science*, 19(4):212–221, 2014.

[27] P. Radivojac, W. Clark, T. Oron, et al. A large-scale evaluation of computational protein function prediction. *Nat Methods*, 10(3):221–227, 2013.

[28] A. Tomczak, J. M. Mortensen, R. Winnenburg, C. Liu, D. T. Alessi, V. Swamy, F. Vallania, S. Lofgren, W. Haynes, N. H. Shah, M. A. Musen, and P. Khatri. Interpretation of biological experiments changes with evolution of the Gene Ontology and its annotations. *Sci Rep*, 8(1):5115, Mar. 2018.

[29] M. P. Stumpf, et al. Estimating the size of the human interactome. *Proceedings of the National Academy of Sciences*, 105(19):6959–6964, 2008.

[30] K. Kosoglu, Z. Aydin, N. Tuncbag, A. Gursoy, and O. Keskin. Structural coverage of the human interactome. *Briefings in Bioinformatics*, 25(1):bbad496, Jan. 2024.

[31] J. De Las Rivas and C. Fontanillo. Protein–protein interaction networks: unraveling the wiring of molecular machines within the cell. *Briefings in Functional Genomics*, 11(6):489–496, Nov. 2012.

[32] G. Wu, X. Feng, and L. Stein. A human functional protein interaction network and its application to cancer data analysis. *Genome Biol*, 11(5):R53, May 2010.

[33] L. Papik, E. Ochodkova, E. Kriegova, et al. Positive impact of structural asymmetries in PPI networks on protein complex detection. *Appl Netw Sci*, 10:38, 2025.

[34] Y. Zhao, J. Wang, J. Chen, X. Zhang, M. Guo, and G. Yu. A Literature Review of Gene Function Prediction by Modeling Gene Ontology. *Front Genet*, 11:400, Apr. 2020.

[35] B. Schwikowski, P. Uetz, and S. Fields. A network of protein-protein interactions in yeast. *Nat Biotechnol*, 18(12):1257–1261, Dec. 2000.

[36] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '14), pages 701–710, 2014.

[37] A. Grover and J. Leskovec. node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 855–864, Aug. 2016.

[38] H. Cho, B. Berger, and J. Peng. Compact integration of multi-network topology for functional analysis of genes. *Cell Systems*, 3(6):540–548, 2016.

[39] X. Yue, Z. Wang, J. Huang, S. Parthasarathy, S. Moosavinasab, Y. Huang, S. M. Lin, W. Zhang, P. Zhang, and H. Sun. Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics*, 36(4):1241–1251, Feb. 2020.

[40] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.

[41] E. Becht, L. McInnes, J. Healy, et al. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat Biotechnol*, 37:38–44, 2019.

[42] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[43] N. Ceglia, Z. Sethna, S. S. Freeman, et al. Identification of transcriptional programs using dense vector representations defined by mutual information with GeneVector. *Nat Commun*, 14:4400, 2023.

[44] S. Oliver. Guilt-by-association goes global. *Nature*, 403:601–602, 2000.

[45] D. Eisenberg, E. Marcotte, I. Xenarios, et al. Protein function in the post-genomic era. *Nature*, 405:823–826, 2000.

[46] C. J. Wolfe, I. S. Kohane, and A. J. Butte. Systematic survey reveals general applicability of "guilt-by-association" within gene coexpression networks. *BMC Bioinformatics*, 6:227, Sept. 2005.

[47] R. Sharan, I. Ulitsky, and R. Shamir. Network-based prediction of protein function. *Mol Syst Biol*, 3:88, Mar. 2007.

[48] J. Gillis and P. Pavlidis. "Guilt by association" is the exception rather than the rule in gene networks.*PLoS Comput Biol*, 8(3):e1002444, Mar. 2012.

[49] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[50] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. arXiv preprint arXiv:1609.02907, 2016.

[51] Z. Chen, Y. Yan, Q. Wang, and H. Chen. GLADC: Global Linear Attention and Dual Constraint for Mitigating Over-Smoothing in Graph Neural Networks. *Algorithms*, 18(12):739, 2025.

[52] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and Deep Graph Convolutional Networks. In Proceedings of the 37th International Conference on Machine Learning, volume 119, pages 1725–1735, 2020.

[53] A. K. Rider, T. Milenković, G. H. Siwo, et al. Networks' characteristics are important for systems biology. *Network Science*, 2(2):139–161, 2014.

[54] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. In International Conference on Learning Representations (ICLR), 2018.

[55] T. N. Kipf and M. Welling. Variational Graph Auto-Encoders. arXiv preprint arXiv:1611.07308, 2016.

[56] C. Xu, D. Tao, and C. Xu. A Survey on Multi-View Learning. arXiv preprint arXiv:1304.5634, 2013.

[57] Z. Yu, Z. Dong, C. Yu, K. Yang, Z. Fan, and C. L. P. Chen. A review on multi-view learning. *Front Comput Sci*, 19(7):197334, 2025.

[58] M. Zitnik, F. Nguyen, B. Wang, J. Leskovec, A. Goldenberg, and M. M. Hoffman. Machine Learning for Integrating Data in Biology and Medicine: Principles, Practice, and Opportunities. *Inf Fusion*, 50:71–91, Oct. 2019.

[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[60] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning, pages 448–456, 2015.

[61] K. He, X. Zhang, S. Ren, and S. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[62] M. Scholkemper, M. T. Schaub, A. Jadbabaie, and J. Pesch. Residual connections and normalization can provably prevent oversmoothing in GNNs. arXiv preprint arXiv:2406.02997, 2024.

[63] A. Paszke, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv preprint arXiv:1912.01703, Dec. 2019.

[64] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), 2015.

[65] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In Proceedings of the 30th International Conference on Machine Learning, pages 1310–1318, 2013.

[66] K. Sechidis, G. Tsoumakas, and I. Vlahavas. On the Stratification of Multi-Label Data. In D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, editors, Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2011. Lecture Notes in Computer Science, volume 6913, pages 145–158. Springer, Berlin, Heidelberg, 2011.

[67] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[68] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine Learning in Python. *JMLR*, 12:2825–2830, 2011.

[69] T. Fawcett. An Introduction to ROC Analysis. Pattern Recognition Letters, 27(8):861–874, 2006.