Department of AI, Data and Decision Sciences

Master's Degree in Data Science and Management
Chair of Data Science in Action

A Machine Learning Approach for Physiological Role Prediction in Protein Contact Networks: a large-scale analysis on the human proteome

SUPERVISOR

Prof. Alessio Martino

CO-SUPERVISOR

Prof. Mauro Sozio

CANDIDATE

Mattia Cervellini

ID 776501

Academic Year 2024/2025

Declaration of Authorship

I, MATTIA CERVELLINI declare that this thesis titled, *"A Machine Learning Approach for Physiological Role Prediction in Protein Contact Networks: a large-scale analysis on the human proteome"* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master's degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 09/09/2025

Acknowledgements

I want to express my sincere gratitude to my supervisor Professor Alessio Martino for his continuous guidance, support, and encouragement throughout the writing of this thesis. His expertise and patience represented an invaluable contribution which significantly enhanced the quality of the resulting work.

I would also like to thank my family, whose encouragement and understanding have sustained me during the entirety of my academic career.

Finally, a heartfelt thanks goes to my friends, who constantly reminded me that whatever obstacles arise there is always room for beer and laughter.

Abstract

Proteins are arguably one of the most interesting families of macromolecules as they are the base building block for most living organisms, involved in virtually all biological processes. Protein structures can be modeled as three-dimensional graphs based on their residue contact networks, making them well-suited for Graph Machine Learning approaches. This thesis conducts a large-scale analysis on the entire human proteome aimed at classifying proteins based on their physiological roles. The work includes different approaches for pattern recognition defined on graph structures based on feature engineering, kernel methods and Graph Neural Networks. Two main feature engineering techniques are employed: the first one is based on the spectral densities of protein graphs, while the second approach is based on the fundamental concept of algebraic topology known as simplicial complexes. Another approach to protein classification was based on Graph Neural Networks with an extensive exploration of recent message passing techniques and possible network architectures. The aforementioned techniques were evaluated on two main tasks: (i) binary classification of protein structures based on whether they perform enzymatic functions, and (ii) multiclass classification of enzymatic proteins according to their enzyme class. Performances have been validated on the entire human proteome by means of repeated stratified splitting to correctly and robustly assess the performance of the analyzed approaches. The final results highlight the Jaccard-based kernel as best performer for the binary task with a balanced accuracy of 0.90 while in the multiclass scenario the GNN architecture displayed the highest discriminative capabilities with a balanced accuracy of 0.92. These results indicate that the multiclass EC class assignment is more complex compared to binary enzymatic classification with the more flexible GNN structure being able to outperform all other methods without significant overfitting. Such results demonstrate that graph-based representations of protein structure enable competitive functional prediction with classical kernel methods and modern message-passing architectures providing comparable strengths.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Biological Motivation

Proteins are the most indispensable type of macromolecules for cellular and organic physiology. Proteins are in charge of many biological processes, the most relevant being: catalyzing metabolic reactions, transducing and integrating biological signals, transporting ions and metabolites, providing mechanical scaffolding and regulating gene expression [2, 103]. Enzymes, in particular, accelerate chemical transformations by stabilizing transition states and lowering activation barriers, thereby enabling reaction rates compatible with life. Perturbations in protein function underlie diverse pathologies [93], from metabolic errors to cancer, so accurate functional annotation is a prerequisite for biological understanding and therapeutic discovery.

Despite decades of progress in biochemistry and structural biology, comprehensive functional characterization has not kept pace with the rapid accumulation of sequences and structures. Moreover, inferring function from sequence alone is complicated by factors like domain shuffling [53], convergent evolution [97], and the presence of multifunctional [49] proteins. Three-dimensional approaches on the other hand impose strict physicochemical constraints on activity: the geometry of active sites and the organization of co-factors collectively shape specificity and catalytic role. These considerations motivate the appearance of computational intelligence strategies that exploit structural information to propose functional hypotheses and guide experiments.

Within this context, inferring a protein's physiological role from its structure is both timely and relevant. Structure-based learning approaches offer a scalable complement to experimental annotation by leveraging patterns that recur across families and folds —such as common structural motifs and global architectural features— to distinguish enzymatic from non-enzymatic proteins and to assign functional classes. By using structural signals in predictive models, such methods aim to bridge the annotation gap and accelerate biological insight across large proteomes.

1.2 Enzymatic Proteins and the EC Numbering System

Proteins can be divided into two macro-categories: enzymatic and non-enzymatic. As previously mentioned enzymatic proteins (i.e., enzymes) facilitate chemical reactions often decreasing the amount of energy required for the reaction to speed it up. To carry out their function, enzymes bind substrates at key locations called active sites and are very specific, binding only specific substrates for specific reactions [56]. Each enzyme is associated with an Enzyme Commission (EC) Number [102], assigned by the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (IUBMB). The EC numbering system presents the hierarchical structure highlighted in Table 1.1.

Level	Description
EC x	**Main class** (e.g., Oxidoreductases, Transferases, etc.)
EC x.x	**Subclass** – type of compound or group involved
EC x.x.x	**Sub-subclass** – specific type of reaction
EC x.x.x.x	**Serial number** of the enzyme in this sub-subclass

Table 1.1: Hierarchy of the EC Numbering System — Source: Adapted from [96]

The IUBMB defined a total of seven distinct main classes of enzymes which are briefly described in Table 1.2. Classes 1 to 6 were defined in the first *Report of the Commission on Enzymes* in 1961 [45] while class 7 was defined later in 2018 because the pre-existent classes were not capable of representing properly the role of proteins specialized in moving molecules and ions across membranes [95].

EC Class	Name	Function
EC 1	Oxidoreductases	Catalyze oxidation–reduction (redox) reactions by transferring electrons between substrates.
EC 2	Transferases	Transfer functional groups (e.g., methyl, glycosyl) from one molecule (donor) to another (acceptor).
EC 3	Hydrolases	Catalyze hydrolytic cleavage of bonds by addition of water (e.g., ester, glycosidic bonds).
EC 4	Lyases	Cleave bonds by means other than hydrolysis or oxidation, often forming double bonds or rings.
EC 5	Isomerases	Catalyze intramolecular rearrangements, converting a molecule into one of its isomers.
EC 6	Ligases	Join two molecules by forming new bonds, typically coupled to ATP hydrolysis.
EC 7	Translocases	Catalyze the movement of ions or molecules across membranes or their separation within membranes.

Table 1.2: Main EC Classes, Names, and Functions — Source: Adapted from [66]

1.3 Graph-Based Representations of Protein Structures

A powerful way to abstract the three-dimensional organization of proteins is to represent them as graphs. In this formulation, amino acid residues are mapped to nodes, while edges encode spatial proximity or chemical interactions between residues. The most common definition relies on a distance threshold applied to Cα atoms resulting in a Protein Contact Network (PCN) [27]. This abstraction preserves essential information about the protein's fold and intramolecular connectivity enabling systematic computational analysis.

Representing proteins as PCNs provides a natural substrate for exploring Graph Machine Learning (GML) methods, which are specifically designed to exploit relational and topological information of graph structures. Unlike conventional vector-based encodings, graph representations capture both local patterns and global architectural features. These properties make PCNs particularly suitable for function prediction tasks, where subtle structural characteristics often govern specificity and catalytic activity. Furthermore, graph abstractions are highly flexible: additional information such as residue type, chemical descriptors, or edge weights based on inter-residue distances

can be seamlessly incorporated to enrich the representation.

By leveraging PCNs, modern learning algorithms can uncover structural signals fundamental for the analysis of complex topological structures like proteins. In this way, graph-based representations act as a bridge between raw structural data and predictive models, offering a scalable framework for linking protein structure to physiological function.

1.4 Research Objectives and Tasks

This thesis investigates the capabilities of a diverse set of GML techniques, including embedding strategies inspired by the granular computing information processing paradigm [5, 80], graph kernels, and Graph Neural Networks (GNNs), to recognize structural patterns for accurate protein classification. The focal objective is to classify proteins in the human proteome according to their physiological functions, formulated as two complementary tasks: (i) a binary discrimination between enzymatic and non-enzymatic proteins (hereinafter Task A) and (ii) a multiclass assignment of enzymatic proteins to their first-level EC classes (hereinafter Task B). The study explores advanced graph representations based on spectral densities and algebraic topology, subsequently evaluating the efficacy of both classical and deep learning classifiers on said tasks.

A distinctive contribution lies in the direct, uniform benchmarking of spectral- and algebraic-topology-based descriptors against modern GNNs, an empirical comparison that appears not to have been systematically executed under shared experimental conditions in prior literature. The evaluation protocol is intentionally rigorous: stratified hold-out validation with fixed splits across all models, systematic hyperparameter optimization, and class-imbalance-aware performance measurements are employed to reflect real-world data characteristics. This design enables paired comparisons and strengthens the reliability of the conclusions. The analysis is conducted at proteome scale on approximately $50,000$ unique human protein structures and spans 12 distinct combinations of learning algorithms and PCN representation techniques, ensuring a comprehensive and fair assessment of methodological strengths and limitations across topological embeddings and deep learning architectures.

These elements combined establish the work as a baseline reference for future studies, offering reproducible results and methodological guidelines to inform the design and evaluation of graph-based approaches for proteome-scale EC class annotation.

1.5 Thesis Outline

The work is organized according to the following structure: Chapter 2 reports a presentation of the main theoretical background of the explored ML and GML approaches and an analysis of the previous scientific works with similar grounds; Chapter 3 presents the technical details on how the data was retrieved and filtered for the analysis; Chapter 4 includes the theoretical and technical description of all methods exploited in the thesis, from modelling to final performances evaluation; Chapter 5 highlights the empirical results deriving from the experimentation carried out and the main data-driven findings; finally Chapter 6 presents a brief recap of the work and the final conclusions together with considerations regarding strengths, limitations and possibilities for future developments of the work.

Chapter 2

Literature Review

This chapter will be split in two separate parts: Section 2.1 will introduce the foundational literature regarding the ML and GML techniques exploited in the thesis, Section 2.2 on the other hand will present a brief taxonomy of the present research and a compendium of scientific literature regarding GML approaches for physiological role prediction in proteins.

2.1 Theoretical Background

2.1.1 An Introduction to Graphs

A graph can be defined as a pair of sets $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ represents a finite set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the finite set of edges [28]. A graph is said to be undirected if its edges represent symmetric relations between nodes, formally the existence of an edge $(u, v) \in \mathcal{E}$ with $u, v \in \mathcal{V}$ guarantees the existence of $(v, u) \in \mathcal{E}$ which is equivalent to (u, v). Directed graphs on the other hand have edges representing directed relations so the existence of (u, v) does not guarantee the existence of (v, u) and the two edges do not carry the same meaning. Graphs in their simple formulation do not account for *self-loops* (i.e., edges of the type (u, u)) and *multi-edges* (i.e., multiple edges connecting the same two nodes). Most commonly in graph notation $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$.

One of the most well-known ways to represent a graph is the so-called *adjacency matrix*: a square matrix of shape $n \times n$ that can be defined as

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

which for simple undirected graphs takes the form a symmetric binary matrix with zero diagonal [28].

Starting from $\mathbf{A}$ it is possible to define the function $D(v_i) = \sum_{j=1}^{n} \mathbf{A}_{i,j}$, laying the foundation for another fundamental graph-related definition: the *degree matrix* $\mathbf{D}$. $\mathbf{D}$ is a diagonal matrix of shape $n \times n$ that can be defined as

$$\mathbf{D}_{i,j} = \begin{cases} D(v_i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Furthermore it is possible to assign an arbitrary number of features to the elements of a graph, effectively creating an *attributed graph* which can be defined as $G_a = (\mathcal{V}, \mathcal{E}, \mathcal{V}_a, \mathcal{E}_a)$ where $\mathcal{V}_a$ and $\mathcal{E}_a$ are two sets of attributes for nodes and edges respectively [8].

Thanks to such properties, graphs are optimally fitted to describe complex topological scenarios in which atomic elements (i.e., nodes) have non trivial binary interactions among each other. A prime example of this representation capabilities are indeed protein structures which can be represented as labelled (attributed) graphs also known as PCNs (further details can be found in Section 4.1).

2.1.2 Graph Embeddings

In order to solve classification problems inside the graph domain directly *graph matching* techniques have been developed with the aim of evaluating the similarity of different graph structures. The most relevant techniques under the graph matching umbrella are: isomorphism tests [67], partial graph matching [105] and inexact graph matching [22], all which are however very limited in the context of graph classification. Especially when it involves large graph structures or a large quantity of graphs, such limitations derive more often than not from prohibitive computational costs (e.g., subgraph isomor-

phism is NP-complete) [15, 23].

A common alternative is to map graphs into an *embedding space* where standard classifiers operate efficiently. An *embedding* procedure transforms complex structured inputs into vectors in a simpler Euclidean metric space. This process effectively creates a compact fingerprint of each input structure enabling scalable learning [40, 72] while on the other hand, an effective embedding space should be able to preserve as much information as possible from the original graph structure [62].

2.1.3 Simplicial Complexes

Simplicial complexes are a concept from algebraic topology which has been widely explored in the domain of graph analysis [65, 77, 85]. They are in essence groups of elements (i.e., simplices) glued together along their faces. A simplex of order k (k-simplex) is effectively a convex hull of $(k+1)$ points: a point is a 0-simplex, a line a 1-simplex, a triangle a 2-simplex and so on. Every non-empty subset of a simplex is a face of said simplex which is itself a simplex of lower order. Following this reasoning a simplicial complex ($\mathcal{S}$) can be defined as a group of simplices (s) having the following two properties:

1. if $s \in \mathcal{S}$, every face in s is also included in $\mathcal{S}$
2. if $s_1, s_2 \in \mathcal{S}$, then $s_1 \cap s_2$ is a face of both s_1 and s_2

Starting from a generic simplicial complex it is possible to define the so-called k-skeleton, representing a simplicial complex whose forming simplices are all at most of order k. It is easy to see how a simple unweighted graph can be seen exactly as a 1-skeleton where graph nodes represent points in the skeleton and each edge is a 1-simplex.

Simplicial complexes are a powerful tool for graph analysis especially in the case of labeled graphs: if each node has its own distinctive label it is possible to create a finite alphabet of all different simplices of node-labels composing the structure which can be exploited to create a *symbolic histogram* of the graph [63]. Simplicial complexes are also suited to model interactions of any order which makes them versatile enough to

be adapted to networks with complex local substructures (e.g., PCNs), more on this in Section 4.2

2.1.4 Kernel Methods

Kernel methods enable learning non-linear patterns by mapping data into a (possibly infinite-dimensional) feature space, where linear models such as Support Vector Machines (SVMs) operate efficiently [16]. The fundamental building block for kernel methods are *kernel functions*, which are used to implicitly embed the data towards a possibly infinite-dimensional Hilbert space $\mathcal{H}$ in which linear classification of data points is more likely [25]. Considering an input space of any kind $\mathcal{X}$, a kernel function can be defined as a continuous function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Furthermore $K(\cdot, \cdot)$ is said to be a positive semi-definite kernel if and only if it respects the following two properties:

$$K(x_i, x_j) = K(x_j, x_i) \qquad \forall x_i, x_j \in \mathcal{X} \tag{2.3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \qquad \forall\, c_i, c_j \in \mathbb{R}, \forall x_i, x_j \in \mathcal{X} \tag{2.4}$$

Considering an instance matrix $\mathbf{X} \in n \times m$ where n is the number of observations and m the number of features per observation, any positive semi-definite kernel applied to observations of $\mathbf{X}$ yields a so-called positive semi-definite Gram Matrix (of shape $n \times n$). This Gram matrix, under Mercer's Theorem [68] (which requires $\mathcal{X}$ to be compact and K to be continuous), admits a feature map ϕ into a (possibly infinite-dimensional) Hilbert space $\mathcal{H}$. This phenomenon justifies the so-called *kernel trick* [87], described by the following equation:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}} \tag{2.5}$$

Equation (2.5) highlights how, instead of performing explicit mapping with the function $\phi(\cdot)$ followed by the dot product among vectors in $\mathcal{H}$, it is possible to use a positive semi-definite kernel function $K(\cdot, \cdot)$ that satisfies Mercer's condition which implicitly performs both operations. A prime example of positive definite kernel is

the Radial Basis Function (RBF) kernel which takes the form of $K(x, y) = e^{-\gamma||x-y||^2}$ which can be intuitively read as an exponentially decaying similarity of squared distance, points very close together act as almost identical while influence falls off rapidly as they move apart depending on the magnitude of γ.

Kernel methods have been widely applied in the domain of graphs since the early 2000s with the appearance of the first graph kernels [55]. Graph kernels can be defined (with a similar logic as the one presented previously) as a positive semi-definite function K in the space of graphs $\mathcal{G}$ for which there is a map $\phi : \mathcal{G} \to \mathcal{H}$ such that $K(G_i, G_j) = \langle \phi(G_i), \phi(G_j) \rangle$ for any $G_i, G_j \in \mathcal{G}$ [73].

2.1.5 Graph Spectrum

Starting from the graph matrix representations presented in Section 2.1.1 it is possible to define two other important matrices for graph analysis. The first one being the *Laplacian Matrix* $\mathbf{L}$ defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{2.6}$$

The matrix $\mathbf{L}$ encodes topological information about the network. Many structural insights can be extracted from its spectrum: from the number of connected components [100], to the algebraic connectivity [35], and the total number of spanning trees [18].

Starting from $\mathbf{L}$ it is possible to define another real symmetric matrix: the *Normalized Laplacian Matrix* $\overline{\mathbf{L}}$ as

$$\overline{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \tag{2.7}$$

$\overline{\mathbf{L}}$ conveys similar structural information as $\mathbf{L}$ but has various additional properties, the most important being that all of its eigenvalues (i.e., its spectrum) lie in the range $[0, 2]$ independently of the characteristics of the underlying graph [17]. Given such property it is possible to consider the spectral decomposition of $\overline{\mathbf{L}}$ as

$$\overline{\mathbf{L}} = \mathbf{P}\mathbf{D}^{\overline{L}}\mathbf{P}^T \tag{2.8}$$

where $\mathbf{D}^{\overline{L}} = \text{diag}\left\{\lambda_1^{\overline{L}}, \ ..., \lambda_n^{\overline{L}}\right\}$ is a diagonal matrix containing the n eigenvalues of $\overline{\mathbf{L}}$ where n is equal to the number of nodes in the original graph. The eigenvalues contained in $\mathbf{D}^{\overline{L}}$ can be interpreted as a sort of signature of the underlying graph, they are however difficult to compare given their variable dimensionality (equal to the number of nodes in the graph), Section 4.4 will present a method to circumvent this issue.

2.1.6 Graph Neural Networks

Deep learning began with architectures designed for data living on regular Euclidean domains. Multilayer perceptrons (MLPs) learn generic non-linear functions on vector inputs; convolutional neural networks (CNNs) leverage locality and weight sharing on grids for images and audio; recurrent neural networks (RNNs) share parameters along sequences. Graph-structured data break these assumptions: there is no canonical ordering of nodes, sizes vary, and neighborhoods are irregular. Graph neural networks (GNNs) arose to bring the same inductive biases (e.g., locality and parameter sharing) to graphs [13, 39, 84]. The earliest GNN formulations extended recursive neural networks to arbitrary graphs and introduced contractive propagation to guarantee convergence [39, 84]. These models already contained the core idea of repeatedly exchanging information along edges and updating node states.

Two complementary message passing (MP) methodologies crystallized how to "convolve" on graphs: (i) spectral view which defines convolution via the graph Laplacian eigenbasis and learn filters in the spectral domain [14,43] and (ii) spatial view which operates directly on neighborhoods with permutation-invariant aggregations [42, 99, 104]. A key discovery on GNNs relates their discriminative power to the Weisfeiler–Leman (WL) graph isomorphism test: message-passing GNNs are more often than not as powerful as the 1-WL test [70].

Figure 2.1 presents the standard structure of a GNN for graph classification: some initial pre-processing is applied to graph structures to prepare them to pass through convolutional layers in which MP is applied. After a variable number of MP rounds the resulting node level representations get pooled into a unique vector corresponding to a graph level representation which gets subsequently fed to a classification head (typically

a MLP structure) which is devoted to making the final graph level predictions.

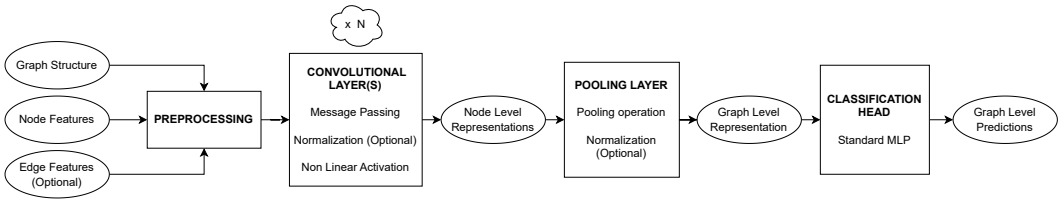


Figure 2.1: Standard GNN structure

2.2 Previous Work on Proteins' Physiological Role Prediction

2.2.1 Problem Framing and Task Taxonomy

In this thesis, physiological role prediction is operationalized as two supervised tasks on protein structures. The first is a binary classification that distinguishes enzymes vs. non-enzymes; the second is a multiclass assignment of the first level EC number. The EC hierarchy and class definitions established by IUBMB (Tables 1.1 - 1.2) are adopted.

Methodologically, a two-stage formulation popularized in early structure-based work [10, 29, 30] is adopted: a binary classification between enzymatic and non-enzymatic proteins (Task A) followed by first level EC number classification on the enzymatic subset (Task B). The thesis mirrors pipelines that model proteins as PCNs whose nodes are residues (or secondary-structure elements) and edges encode spatial proximity, enabling graph-based learning downstream.

2.2.2 Early Alignment-Free ML Approaches

Plenty of studies in literature deal with the classification of proteins according to their physiological role. One very early example is Dobson & Doig (2003) [29], who aimed to distinguish enzymatic from non-enzymatic proteins without using sequence or structural alignments which were very popular at the time (e.g., BLAST [3], FASTA [79], etc.). They described proteins using both simple sequence-derived features (e.g., amino

acid composition, etc.) and structure-derived features (e.g., secondary structure content, largest pocket size, etc.). These feature vectors were classified using SVMs with adaptive feature selection yielding a simplified model that achieved around 80% accuracy. A natural evolution of this study is Dobson & Doig (2005) [30] where the authors expand the classification problem to predict also the first level EC number of the analyzed proteins maintaining similar input features in a One-vs-One (OvO) setting. The performances of this later study are stained by strong class imbalance with accuracy scores ranging from around 80%, for well balanced classes, to around 50% for highly imbalanced situations. Together, these studies established the feasibility of alignment-free, feature-based ML for physiological-role prediction while highlighting two themes that recur in later work: (i) structure-derived descriptors can add value beyond sequence-only baselines; and (ii) evaluation must account for class imbalance and protocol design (see Section 4.9).

2.2.3 Proteins as PCNs for GML

Protein contact networks —also called residue interaction/contact networks— abstract a protein's 3D structure as a graph whose nodes are residues (typically Cα atoms) and whose edges encode spatial proximity; common design choices include an 8 Å cutoff (or a 4–8 Å window to suppress trivial backbone neighbors) and, in some variants, weighted edges based on inverse distances [27].

Early work by Borgwardt et al. (2005) [10] modeled proteins at the level of secondary-structure elements (SSEs), connecting SSEs via sequential and structural edges, and trained SVMs with random-walk graph kernels in a two-stage pipeline (usual enzyme vs. non-enzyme followed by first-level EC class) on balanced EC subsets. Adding structural edges and specific global attributes improved performance over sequential-only or attribute-free baselines [10].

Martino et al. (2017) [64] work with Cα contact networks adding edges when residues are in the range [4-8] Å and summarize them via topology- and spectrum-based descriptors (e.g., normalized-Laplacian spectral densities, centralities, etc.) to feed kernel methods and SVM variants. With this approach the residue–residue contacts define

the learning substrate for the applied ML algorithm. This study is also focused on the first level EC number prediction for a subset of the proteome of *E. coli*.

De Santis et al. (2018) [26] embed heterogeneous PCN and sequence descriptors (spectral density, centralities, protein size, Betti numbers, primary sequence) into a dissimilarity-space and test three strategies: (i) features selected via genetic algorithms with PARC/ν-SVM, (ii) an isometric embedding with standard classifiers, and (iii) a cluster-based one-class model that learns feature weight. They achieve an F1-score of ~ 0.70 with ν-SVM and up to 0.75 with the one-class system on $1,224$ *E. coli* proteins (703 enzymes, 521 non-enzymes).

Various other studies exploited PCNs for a plethora of computational intelligence tasks, from standard classification to more advanced generative tasks [59–61]. Given their versatility and intuitiveness PCNs have been deemed the perfect technique for protein analysis and have been used as basis for all ML experiments presented in Chapter 4.

2.2.4 GNNs for EC number Prediction

Structure-aware GNNs are able to operate directly on PCNs, in this setting, node attributes usually include residue identity and/or physicochemical descriptors (and optionally secondary structure/solvent accessibility), while edges carry geometric features such as inter-residue distances or orientation; a message-passing encoder with graph pooling yields a graph-level representation that feeds one or more classification heads (see Section 2.1.6). Many pipelines implement a two-stage protocol in which a dedicated binary head filters enzymes before EC classification, whereas others fold non-enzymes into a unified multiclass objective by adding a "non-enzyme" label or thresholding EC logits and others operate on two distinct datasets directly one for the enzyme vs. non-enzyme classification and another for EC number prediction (the latter being the chosen method for this thesis as anticipated in Section 2.2.1).

Representative PCN-based approaches include works such as Graph Convolutional Networks (GCN) by Kipf & Welling (2016) [54] that combine residue features with proximity edges to predict EC labels often alongside Gene Ontology terms [38]. More

recent equivariant GNNs inject 3D geometry directly into message passing either by coupling sequence transformers with E(3)-equivariant layers over residue graphs derived from experimental or AlphaFold2 structures [9], or by using geometric vector perceptrons that jointly propagate scalar and directional features on PCNs [50]. Across studies, using PCNs as the learning substrate consistently improves over sequence-only baselines, with the largest gains in remote-homology regimes. When reliable coordinates are available, equivariant architectures tend to outperform plain GNN models on both enzyme detection and top-level EC classification.

Most recent architectures require significant computing capabilities and in order to mediate between the effectiveness of most recent approaches and acceptable model training times this thesis includes only standard GNN models (as in Figure 2.1) whose structures have been optimized for the two tasks of interest. Various MP strategies and different topologies are explored in order to find the best possible architecture with reasonable training times on the available machines (for additional details refer to Section 4.8).

2.2.5 Datasets Choice

Previous studies constructed their dataset starting from either selected balanced subsets of proteins [10], which facilitated modelling but was poorly representative of the real biological distribution of proteins in living organisms, or from subsets of proteomes of simple organisms such as *E. coli* [26,64]. More recent studies regarding Deep Learning architectures built more comprehensive datasets composed hundreds of thousands proteins, with some reaching more than $500,000$ structures [9]. This thesis takes a hybrid approach, the starting dataset represents the entirety of the human proteome composed of approximately $70,000$ distinct structures, said dataset was filtered to select proteins with a single first-level EC number and good enough resolution to be candidates for PCN construction. After the data cleaning procedure the resulting dataset included more than $48,000$ proteins with approximately $21,000$ enzymatic structures, all the details can be found in Chapter 3. Task A exploited the entirety of the dataset while Task B leveraged only the enzymatic proteins. This process allowed the realization of a

large-scale analysis of protein structures with specific focus on the human proteome.

Chapter 3

Data Collection

The initial dataset for the analysis consisted of $69,979$ distinct protein structures representing the entirety of the human proteome. The original files were mass downloaded from Protein Data Bank (PDB) on March the 1^{st} 2025 and subsequently parsed via specialized Python packages such as BioPython [21] and BioPandas [81] in order to extract the following relevant characteristics for each protein:

- Coordinates of all Cα atoms belonging to standard amino acids, essential for the construction of PCNs (In the case of *alternate locations*[1] the average coordinates were considered)
- Residue names related to each Cα atom
- Resolution of the experiment used to determine the positions of atoms in the structure
- First Level EC Number(s)

The resulting protein structures were then filtered, specifically, proteins belonging to the following categories were discarded:

1. Any protein showing an evidently degenerate structure (e.g., one single residue, very far residues, etc.)

[1]For additional details refer to Proteopedia, *Alternate Locations in PDB Files*

2. Any protein which exhibits multifunctional [49] or moonlighting [48] properties (i.e., presenting more than one first level EC Number)

3. Any protein with missing resolution or with a resolution exceeding 3Å. This threshold was chosen because, given the residue interaction range of [4–8]Å, only highly detailed structures are informative for the analysis.

After filtering, the dataset was composed of $48,019$ proteins divided in $26,312$ non enzymatic structures and $21,707$ enzymatic ones. The enzymatic ones were further distributed as presented in Table 3.1, notably classes 5 (Isomerases), 6 (Ligases) and 7 (Translocases) are substantially underrepresented in the dataset compared to other EC Classes and will probably prove to be hard to correctly classify. EC Class 7 in particular is represented by only 28 structures across the entire dataset, this phenomenon is probably due to it being the newest EC class, introduced only in 2018 [95]. Given this extreme scarcity in the dataset Translocases were ignored in the context of Task B.

Given the two layer experimental procedure of this work two distinct datasets were created: (i) a dataset of all $48,019$ structures divided in Non-Enzymatic and Enzymatic (i.e., belonging to any EC Class) for Task A and (ii) a dataset of $21,679$ enzymatic structures (after the removal of Translocases), each with its own first level EC Number for Task B. This two datasets were used in parallel for all of the experimental steps of this work.

EC	Count	Percentage
1	2959	13.63%
2	8878	40.90%
3	7181	33.08%
4	1557	7.17%
5	660	3.04%
6	444	2.05%
7	28	0.13%

Table 3.1: EC Number Distribution for Enzymatic Proteins

Chapter 4

Methodologies

This section will include the technical and theoretical specifications of all methods used in the experimental framework of the thesis.

4.1 Proteins as PCN-Graphs

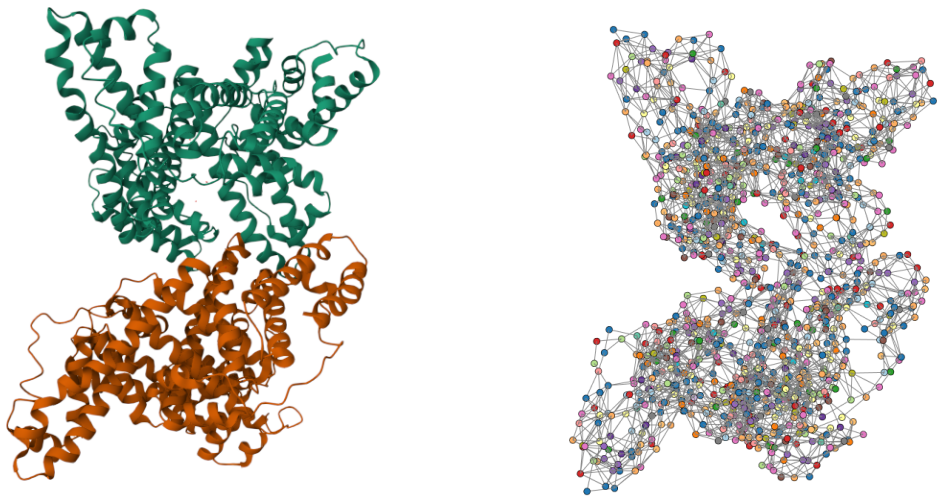
As anticipated in Section 2.2.3 protein structures downloaded from PDB were processed by creating PCNs following the same approach found in [26, 27, 64]. Starting from the 3D coordinates of atoms in the proteins (often retrieved by means of X-Ray Crystallography [7]) only atoms belonging to the 20 standard amino acids were considered (see Table 4.1), specifically their Cα atoms were used as nodes in the resulting PCNs. Cα atoms were connected if the Euclidean distance among them was in the range [4-8]Å, the lower bound was set in order to discard trivial first-neighbor interactions along the chain of a protein while the upper bound is set to 8Å which corresponds approximately to two van der Waals radii of Cα atoms [78]. Outside such range residues are assumed to have no relevant interaction. The final PCNs are graphs whose nodes are labeled with the name of the residue corresponding to each Cα atom. Edges were deliberately kept attribute-free to foster learning directly from the interactions of residues. Following the same rationale the resulting graphs have no notion of the original 3D space, they only retain information about the connectivity structure of the amino acids.

Figure 4.1 presents the structure of human serum albumin (1AO6 on PDB) both in

Amino acid	PDB 3-letter code	Amino acid	PDB 3-letter code
Alanine	ALA	Leucine	LEU
Arginine	ARG	Lysine	LYS
Asparagine	ASN	Methionine	MET
Aspartic acid	ASP	Phenylalanine	PHE
Cysteine	CYS	Proline	PRO
Glutamine	GLN	Serine	SER
Glutamic acid	GLU	Threonine	THR
Glycine	GLY	Tryptophan	TRP
Histidine	HIS	Tyrosine	TYR
Isoleucine	ILE	Valine	VAL

Table 4.1: The 20 Standard Amino Acids and Their Abbreviated Residue Names in PDB – Source: Adapted from [82]

its original form directly from PDB 3D structure visualization (Figure 4.1a) and in its PCN representation (Figure 4.1b) where each node is colored according to the original residue of the relative Cα atom. Nodes of the PCN were placed in the correct 3D location for visualization purposes only, the exact same structure appears as in Figure 4.2 when drawn according to Fruchterman-Reingold force-directed algorithm [36]. This highlights how the performances of the models will depend uniquely on feature engineering techniques and residue interactions.



(a) Protein Model on PDB

(b) PCN from Cα atoms

Figure 4.1: Representations for Human Serum Albumin (1AO6)

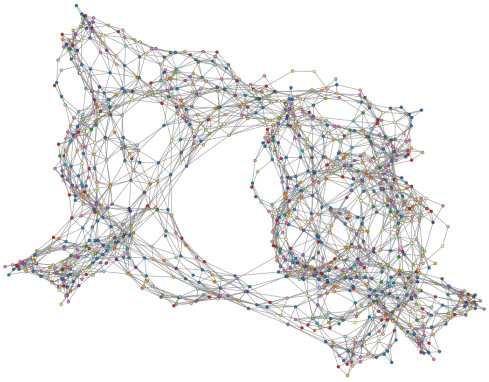


Figure 4.2: Human Serum Albumin Force-Directed Representation

4.2 Graph Embedding via Simplicial Complexes

Starting from PCNs as described in the previous Section together with the notion of Simplicial Complexes anticipated in Section 2.1.3 it is possible to conceptualize a graph embedding technique based on Simplicial Complexes. Simplices of node-labels (i.e., amino acid names) can be used to create a *symbolic histogram* of the protein structure by counting how many times each simplex appears in the protein. Each graph structure can be effectively embedded in the Euclidean Space with a multi-set of its simplices. This process however lacks representation capabilities when applied to out-of-the-box PCNs which are able to describe only pairwise relationships (PCNs are by definition simple, unweighted graphs). Considering 20 different amino acids there could be only $\binom{20+2-1}{2} = 210$ possible combinations of length two which would probably harm the expressive power of the embedding.

In order to circumvent this issue the edges of PCNs were aggregated creating clique hypergraphs. An hypergraph is a graph generalization in which edges (commonly known as hyper-edges) can connect an arbitrary number of nodes at the same time, formally an hypergraph can be defined similarly to a graph as $H = (\mathcal{V}, \mathcal{E}_h)$ in which $\mathcal{V}$ is a finite set of nodes (same as plain graphs) while $\mathcal{E}_h \in \mathcal{V}^n$ is a set of hyper-edges which can connect any number of nodes of $\mathcal{V}$ simultaneously. Hypergraphs have been explored in literature to represent a great variety of complex systems (e.g., co-authorship networks, metabolic networks, brain functional networks, etc.) due to their flexibility and expressive power [32, 41, 52, 69, 75, 101]. A clique hypergraph in particular is an

hypergraph generated starting from a plain graph in which maximal cliques were substituted with hyper-edges of the same order [4, 107]: a simple example of this process would be a triangle in the graph G described by edges $\{(A, B), (B, C), (A, C)\}$ which gets projected in the hypergraph H as a single hyper-edge $\{(A, B, C)\}$. The same procedure is carried out for cliques of all orders. In order to find all maximal cliques to merge into hyper-edges an iterative version of Bron-Kerbosch algorithm was used [12].

The complete dictionary of possible hyper-edges could in theory be enormous. Considering maximal cliques of all sizes up to R and treating each as a combination of the 20 amino acids (order irrelevant, repetitions allowed), the number of possible compositions is

$$\sum_{r=1}^{R} \binom{r + 20 - 1}{r} = \sum_{r=1}^{R} \binom{r + 19}{19} = \binom{R + 20}{20} - 1 = \Theta\left(R^{20}\right) \tag{4.1}$$

where R is the size of the largest maximal clique in the dataset. In practice however the number of distinct hyper-edges in the dataset is much smaller ($\sim 16,000$).

The instance matrix $\mathbf{X}^{(S)}$ resulting from such embedding via simplicial complexes has shape $n \times |d|$ where n is the number of proteins in the dataset and d is the dictionary of distinct simplices in the dataset. Considering $c(H_i, d_j)$ a function that counts how many times the simplex d_j appears in the clique hypergraph H_i the instance matrix $\mathbf{X}^{(S)}$ can be defined as

$$\mathbf{X}^{(S)}_{i,j} = c(\mathcal{H}_i, d_j) \tag{4.2}$$

Given the two-step experimental framework described in Section 2.2.1, two distinct instance matrices were constructed: (i) $\mathbf{X}^{(S)}_A$ including a dictionary d_A of all simplices from both enzymatic and non-enzymatic proteins ($\sim 16,000$ in total) for Task A; and (ii) $\mathbf{X}^{(S)}_B$ including a dictionary d_B of only simplices coming from enzymatic proteins ($\sim 13,000$ in total) for Task B.

4.2.1 INDVAL Scores

The dimensionality of the graph embedding via simplicial complexes presented in the previous section is far from the worst possible case (Equation (4.1)) however considering the fact that the datasets involved in the analysis consist of tens of thousands of distinct proteins it was relevant to investigate whether it was possible to perform some degree of feature selection while maintaining stable classification performances during the modelling phase.

In order to carry out this procedure with a model-agnostic rationale the INDVAL score was considered. The INDVAL score is a sensitivity and specificity integrated evaluation originally proposed to individuate the most characteristic species of a given environment [31]. According to the INDVAL criterion, a species s is a good representative for an environment E if

1. s is present only (or almost only) in environment E — proxy for specificity
2. s is present in all (or almost all) of environments E — proxy for sensitivity

The same exact rationale can be used for individuating signature substructures for a specific type of proteins. In order to adapt the INDVAL scores to the experimental framework of this thesis it is possible to draw a parallel where each simplex of node-labels in the aforementioned dictionary d is considered a species and each class is considered an environment. With this approach it was possible to construct a restricted version of the embedding presented in section 4.2 where only the sub-structures with the best INDVAL scores are considered.

Considering j a specific class and i a specific simplex the unified INDVAL score can be defined starting from the following scores:

$$A_{i,j} = \frac{\text{\# structures in class } j \text{ having simplex } i}{\text{\# patterns having simplex } i} \tag{4.3}$$

$$B_{i,j} = \frac{\text{\# structures in class } j \text{ having simplex } i}{\text{\# patterns belonging to class } j} \tag{4.4}$$

where $A_{i,j}$ is a measure of specificity, maximized when simplex i appears only in class j, and $B_{i,j}$ is a measure of sensitivity, maximized when all structures of class j have the simplex i. By combining $A_{i,j}$ and $B_{i,j}$ the final INDVAL score $I_{i,j}$ reads as:

$$I_{i,j} = A_{i,j} \cdot B_{i,j} \cdot 100 \tag{4.5}$$

Given the fact that $A_{i,j} \in [0, 1]$ and $B_{i,j} \in [0, 1]$ it follows that $I_{i,j} \in [0, 100]$ with the following interpretation: an INDVAL Score $I_{i,j}$ of 100 is assigned to the *perfect* simplex (i.e., substructure) from a classification point of view. In practical terms, simplex i would appear in all observations of class j and never in observations belonging to any other class (i.e., it would exactly mimic the class label).

Given a classification task where the *symbolic histogram* technique was used to embed the input data it is possible to define its INDVAL matrix $\mathbf{I}$ of shape $|C| \times |d|$ where d is the dictionary of all the symbols in the dataset and C is the set of unique classes. Each value in $\mathbf{I}$ corresponds to the INDVAL score of simplex d_j for class C_i

After the creation of $\mathbf{I}$ it is possible to select sub-structures (features) which had at least one of their INDVAL scores above a certain user-defined threshold τ. The choice of τ is not straightforward and strongly depends on the specific dataset as there is no guarantee of the existence of the so-called *perfect* symbol: INDVAL scores could be upper bounded at a value far lower than 100 in a specific dataset. It is however possible to pick τ with a data driven approach thanks to some intuitive heuristics.

Considering the experimental framework of the thesis two distinct INDVAL matrices were defined: $\mathbf{I}_A$ with shape $|C_A| \times |d_A|$ for Task A and $\mathbf{I}_B$ with shape $|C_B| \times |d_B|$ for Task B. Figures 4.3 and 4.4 present the number of features included in each dataset (Task A and B) for every possible τ. It is apparent how there is no *perfect* symbol in any of the two datasets, in particular the maximal INDVAL scores were 43.03 and 73.52 in Task A and Task B respectively. Notably most of the scores are near 0 as highlighted by the abrupt decrease in number of included features as the threshold increases. The two vertical lines in the figures represent the two chosen thresholds $\tau_A = 6$ and $\tau_B = 10$ (for Task A and B respectively). Such thresholds were chosen in order to let the final INDVAL embedding retain $\sim 10\%$ of the original feature set while mostly respecting

the elbow rule heuristic [94]. This sharp feature selection will put to test the effectiveness of INDVAL scores in selecting the most relevant sub-structures for classification problems.

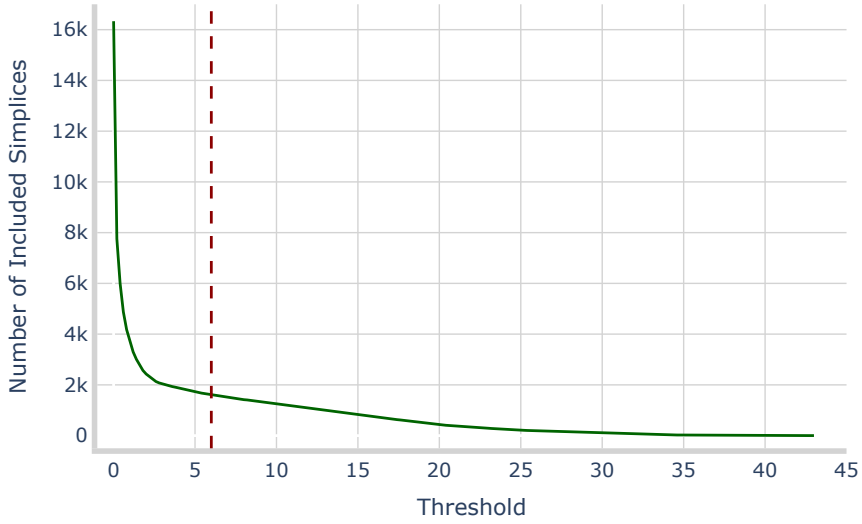


Figure 4.3: Number of Features at Various Threshold Levels for Task A

The INDVAL feature selection procedure generated two instance matrices: $\mathbf{X}_A^{(\text{INDVAL})}$ with $\sim 1,600$ features for Task A and $\mathbf{X}_B^{\text{INDVAL}}$ with $\sim 1,300$ features for Task B.

As a final remark both $\mathbf{X}_A^{\text{INDVAL}}$ and $\mathbf{X}_B^{\text{INDVAL}}$ were obtained by removing features of $\mathbf{X}_A^{(S)}$ and $\mathbf{X}_B^{(S)}$ by comparing entries in $\mathbf{I}_A$ and $\mathbf{I}_B$ with τ_A and τ_B respectively.

4.3 (Hyper)Graph Kernels

As anticipated in Section 2.1.4 graph kernels are among the most widely used techniques for machine learning tasks on graph structures [74]. The methods presented in this section will take as inputs the embedding created by leveraging the clique hypergraph expansion of PCNs combined by the notions on simplicial complexes presented in Section 4.2. The two following (hyper)graph kernel methods are adapted from Martino & Rizzi (2020) [65] and can be constructed directly starting from $\mathbf{X}_A^{(S)}$ and $\mathbf{X}_B^{(S)}$.

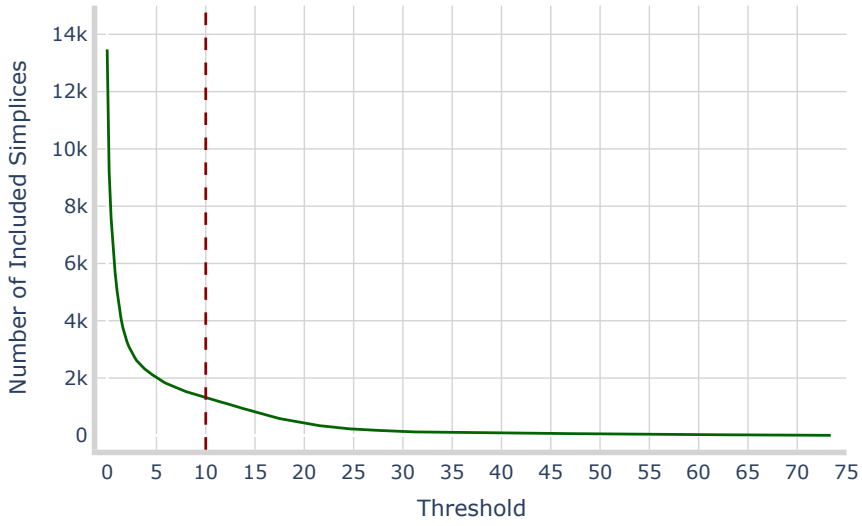


Figure 4.4: Number of Features at Various Threshold Levels for Task B

4.3.1 Histogram Cosine Kernel

The Histogram Cosine Kernel (HCK) between any two PCNs can be computed as the cosine similarity between their respective *symbolic histogram* representations (rows in $\mathbf{X}^{(S)}$). Considering $\mathbf{x}_i = \mathbf{X}^{(S)}_{i,\cdot}$ the HCK between any two PCNs i and j can be defined as

$$K_{\mathrm{HC}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{||\mathbf{x}_i||\ ||\mathbf{x}_j||} \in [0, 1] \tag{4.6}$$

4.3.2 Jaccard Kernel

The Jaccard Kernel (JK) starts from the exact same *symbolic histogram* as the HCK but is computed as the ratio between the intersection and the union of the two multisets. Considering again d as the dictionary of all simplices represented in the histograms and $\mathbf{x}_i = \mathbf{X}^{(S)}_{i,\cdot}$ the JK between any two PCNs i and j can be defined as

$$K_{\mathrm{WJ}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{d} \min\{x_{ik},\, x_{jk}\}}{\sum_{k=1}^{d} \max\{x_{ik},\, x_{jk}\}} \in [0, 1] \tag{4.7}$$

4.4 Spectral Density

As anticipated in Section 2.1.5 the spectrum of the Normalized Laplacian Matrix $\overline{\mathbf{L}}$ of a graph (hereinafter referred to as graph spectrum) can be interpreted as a sort of fingerprint regarding the connectivity properties of the graph itself. The most popular example of such properties being the multiplicity of the eigenvalue 0 corresponding exactly to the number of connected components in the graph.

While the set of eigenvalues of $\overline{\mathbf{L}}$ is highly informative for characterizing global connectivity, its cardinality equals the number of nodes n in the graph, which prevents a direct, size-agnostic use across graphs of different orders which is key in supervised learning pipelines as the one presented in this work.

In order to circumvent this issue it is worth recalling that all eigenvalues of $\overline{\mathbf{L}}$ are real non-negative numbers and specifically lie in the range $[0, 2]$, for these reasons it is possible to approximate the spectral density of a graph thanks to kernel density estimation (KDE) [61] with a Gaussian Kernel [76]. In order to do so, $\overline{\mathbf{L}}$ is treated as random matrix with known spectral density $p(x)$ [47, 64] that can be defined as:

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\left(-\frac{(x-\lambda_i)^2}{2\sigma^2}\right)} \tag{4.8}$$

where σ represents the bandwidth of the kernel which, in order to scale it automatically with graph-sizes, was decided in accordance with Scott's rule [88]. Considering n the number of nodes and $\hat{\sigma}$ the sample standard deviation (relative to eigenvalues), σ was defined as

$$\sigma = \frac{\hat{\sigma}}{n^{1/5}} \tag{4.9}$$

To move from the continuous density to a fixed-length feature vector, $B = 200$ uni-

formly spaced points in $[0, 2]$ were sampled, producing a spectral embedding $\mathbf{s} \in \mathbb{R}^{200}$ for each PCN. This process yields comparable, size-agnostic descriptors that integrate seamlessly into the classification pipeline. Figure 4.5 shows the estimated spectral density for Human Serum Albumin (1AO6); the corresponding embedding is given by the sequence of KDE evaluations at the 200 grid points.

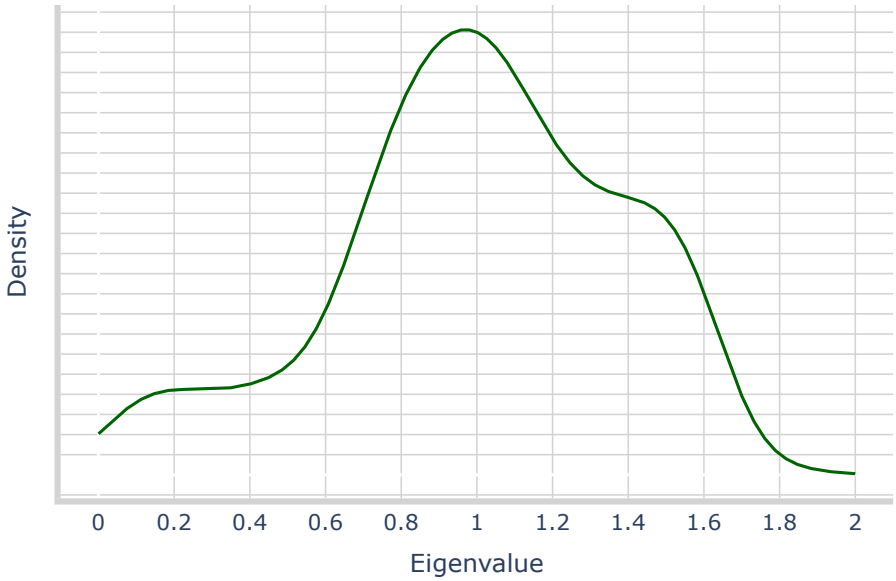


Figure 4.5: Spectral Density KDE Estimation for Human Serum Albumin (1AO6)

Applying the above KDE-based spectral embedding to each PCN yields two instance matrices, $\mathbf{X}_A^{(\mathrm{SPECTRAL})}$ and $\mathbf{X}_B^{(\mathrm{SPECTRAL})}$, corresponding to Task A and Task B, respectively. In both matrices, rows index proteins and columns store the $B = 200$ evaluations of their spectral density on the uniform grid over $[0, 2]$. Hence both datasets have identical feature dimensionality ($B = 200$).

4.5 Summary of Representation Techniques

In summary, the experimental framework of the thesis relies on three complementary representations derived from protein contact networks (Section 4.1).

First, the simplicial complex embedding (Section 4.2) augments PCNs via clique hypergraphs to capture higher-order, residue-labeled interactions; the resulting symbolic histograms offer direct compatibility with both standard ML algorithms and kernel methods emphasizing label-aware local structures. The main limitations of said method are the combinatorial growth of the dictionary (see Equation (4.1)) and its high sparsity. Two non parametric hypergraph kernel approaches have been explored on the simplicial complex embedding (Section 4.3) as they allow the convenient insertion of non-linear similarity metrics among graph structures. As anticipated in Section 2.1.4 kernels do not produce explicit embeddings of the input data but complete Gram matrices which encode similarity among input objects. Such matrices need specialized attention during the modelling phase as they are not suited for the majority of ML models.

Second, INDVAL-based feature selection (Section 4.2.1) retains substructures that are simultaneously specific to and prevalent within classes, yielding compact dictionaries (i.e., $\sim 10\%$ of the original features considering thresholds τ_A and τ_B) with a model-agnostic rationale. Threshold choice remains however data-dependent and could filter out individually weak but jointly informative patterns.

Third, the spectral density representation (Section 4.4) maps each PCN to a fixed-length vector by estimating the Normalized Laplacian eigenvalue density on $[0, 2]$, providing a size-agnostic summary of global connectivity that is robust to graph size; its weaknesses include loss of residue labels and local motif identity and potential cospectral graphs.

Taken together, the simplicial/INDVAL path prioritizes labeled, higher-order local structure with controlled dimensionality, whereas spectral density contributes a compact, global fingerprint; these views are complementary within the supervised framework of Section 2.2.1.

Table 4.2 presents a schematic summary of the dimensions of each embedding methodology explored.

[2]Dimensionality varies slightly across splits of the data, refer to Section 4.11

	Dimensionality	
Representation	Task A	Task B
Simplicial Complexes	$\sim 16,000^2$	$\sim 13,000^2$
INDVAL	$\sim 1,600^2$	$\sim 1,300^2$
Spectral Density	200	200

Table 4.2: Dimensionality of Embedding Strategies

4.6 Standard Classifiers

The embedding techniques presented in the previous sections have afterwards been compared using a variety of well-known ML algorithms for classification. Early experiments explored most of the families of classification models (e.g., linear models, tree-based models, SVMs, K-NN, logistic regression with splines, etc.). After some early pruning the choice of optimal candidates for the study was: (i) ℓ_1-Linear-SVM for its speed and feature selection capabilities, (ii) Kernel ν-SVM for its ability to work also with pre-computed kernels and great flexibility and (iii) Random Forest for its all-around good performances on a great variety of tasks.

4.6.1 ℓ_1-Linear-SVM

The ℓ_1-norm linear SVM model learns a maximum-margin hyperplane by minimizing the empirical squared hinge loss under regularization [24]. Using an ℓ_1-penalty on the weight vector $\mathbf{w}$ promotes sparsity and leads to embedded feature selection [106]. The ℓ_1 regularizer drives many coefficients exactly to zero, improving interpretability and reducing variance: such properties make ℓ_1-penalty particularly desirable for high-dimensional, sparse representations such as symbolic histograms [33]. The optimization problem remains convex and can be handled efficiently: ℓ_1 solutions however may be less stable under strong feature collinearity than their ℓ_2-regularized counterparts [108]. ℓ_1-norm linear SVMs can also handle multiclass classification problems thanks to the One-vs-Rest (OvR) strategy [98]. Mathematically speaking, ℓ_1-norm linear SVM solves the following optimization problem [33]:

$$\min_{\mathbf{w}} \|\mathbf{w}\|_1 + C \sum_{i=1}^{n} \Big(\max\{0,\ 1 - y_i\, \mathbf{w}^\top \mathbf{x}_i\} \Big)^2 \tag{4.10}$$

where $\mathbf{x_i}$ represents the feature vector for observation i, y_i its associated class label (encoded as $\{-1, 1\}$) and C is the so-called regularization parameter. C governs the balance between the two terms of the objective function and is defined as strictly positive. Its value is inversely related to the strength of the regularization: small values of C enforce stronger penalization on the coefficients, while larger values of C reduce the relative impact of the penalty. Intuitively, as C increases, the contribution of the empirical squared hinge loss becomes predominant with respect to the ℓ_1-norm term, leading the model to prioritize a closer fit to the training data. In this sense, the parameter C effectively controls the trade-off between model complexity and generalization ability, determining whether the resulting hyperplane adheres more tightly to the training samples or emphasizes sparsity in the weight vector.

Notably, there is no upper bound for the hyper-parameter C, which can make the search for the optimal values quite cumbersome depending on the specific dataset.

4.6.2 Kernel ν-SVM

ν-SVM is a variation of the standard C-SVM (whose linear version was presented in the previous section, although with ℓ_1 regularization) first introduced by Scholkopf et al. (2000) [86]. The parameter ν replaces C as the primary regularizer and provides interpretable control over model complexity: it is an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors [20]. By definition, $\nu \in (0, 1]$.

While the linear version of ν-SVM can be expressed in primal form (similar to Equation (4.10)), extending it to non-linear decision boundaries requires kernelization. Since explicitly mapping data into a high-dimensional feature space $\phi(\mathbf{x})$ is generally infeasible, the dual ν-SVM formulation is preferred as it allows the data to appear only in the form of inner products [44]. Replacing feature vectors with kernel evaluations $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ allows the model to operate implicitly in a kernel Hilbert space $\mathcal{H}$ without explicitly computing $\phi(\mathbf{x})$ (see Section 2.1.4). This kernel trick enables efficient non-linear classification while preserving the convexity of the optimization problem.

The dual problem for kernel ν-SVM is formulated as follows [20]

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \tfrac{1}{2}\,\boldsymbol{\alpha}^\top Q \boldsymbol{\alpha} \\ \text{s.t.} \quad & 0 \le \alpha_i \le \tfrac{1}{l}, \quad i = 1, \ldots, l, \\ & e^\top \boldsymbol{\alpha} \ge \nu, \\ & y^\top \boldsymbol{\alpha} = 0, \end{aligned} \tag{4.11}$$

where $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, e is the vector of all ones, l denotes the number of training samples, and α_i are the Lagrange multipliers associated with the constraints. The corresponding decision function for a new input $\mathbf{x}$ is expressed in terms of the support vectors:

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right). \tag{4.12}$$

It is important to note that not all values of $\nu \in (0, 1]$ are viable for every classification problem, Chang & Lin (2001) [19] demonstrated that Equation (4.11) is feasible only if

$$\nu \le \frac{2 \min(\#y_i = 1, \#y_i = -1)}{l} \le 1 \tag{4.13}$$

This phenomenon restricts heavily the possible values for ν especially in highly imbalanced classification scenarios

In summary, Kernel ν-SVM generalizes the linear case to non-linear feature spaces while retaining the interpretability of the ν parameter. Its bounded range often simplifies hyperparameter tuning compared to C-SVM, although the practical choice of ν remains dataset- and kernel-dependent. As with other SVM variants, multiclass classification can be addressed through the OvR strategy [98].

4.6.3 Random Forest

Random Forest (RF) is an ensemble method introduced by Breiman (2001) [11] that extends decision trees through randomized bagging and feature subspacing to reduce variance and improve generalization. Each tree is trained on a bootstrap sample of

the training set (sampling n instances with replacement) and, at every internal node, the split is chosen by maximizing the impurity decrease considering a random subset of features of size m_{try} [11]. This dual source of randomness (resampling rows and subspacing columns) decorrelates individual trees, so that their aggregation via majority vote (for classification tasks) generates a relatively low-variance estimator. Formally, for a forest with B trees $\{h_b\}_{b=1}^{B}$, the RF classifier predicts

$$\hat{y}(x) = \text{mode}\{\, h_b(x) \,\}_{b=1}^{B}, \tag{4.14}$$

and class-posterior estimates are obtained by averaging per-tree posteriors:

$$\widehat{P}(Y = c \mid x) = \frac{1}{B} \sum_{b=1}^{B} \widehat{P}_b(Y = c \mid x). \tag{4.15}$$

Individual trees are typically grown to near purity without pruning using impurity criteria such as the Gini index [37] or Entropy [90].

RF models are naturally able to compute the importance of each input feature quantified in mean decrease in impurity (MDI). In practice RFs are robust to high-dimensional and mixed-type inputs, require few hyper-parameters, and accommodate class imbalance via class weights or stratified sampling. These properties make RF a natural baseline for both the symbolic-histogram and the spectral-density embeddings employed in this work [58, 92].

4.7 Summary of Standard Classification Algorithms

Each of the classification algorithms presented in the previous section was used on all of the embedding methods described in Sections 4.2, 4.2.1 and 4.4. In order to explore the representation capabilities of the kernel methods described in Section 4.3 only the ν-SVM model was explored as it naturally supports pre-computed kernels thanks to the dual formulation of the SVM problem presented in Equation (4.11). Table 4.3 presents a synthetic summary of which classification model was used in combination with each representation strategy.

	ℓ_1-SVM	ν-SVM	RF
Simplicial Complexes Embedding	✓	✓	✓
INDVAL Embedding	✓	✓	✓
Spectral Density Embedding	✓	✓	✓
Histogram Cosine Kernel	✘	✓	✘
Weighted Jaccard Kernel	✘	✓	✘

Table 4.3: Summary of Representation Methods and Related Classification Algorithms

4.8 GNNs

As anticipated in Section 2.1.6 GNN architectures have been demonstrated to be extremely capable for graph classification often reaching state-of-the-art results. Despite GNNs often respecting the conceptual topology of Figure 2.1 there exist a plethora of different MP strategies and a virtually infinite number of different architectures to test, many of which would have prohibitive training times. In order to accommodate for the limited computing power available and to prevent known issues related to MP such as over-smoothing (i.e., most node collapsing to near-identical representations after numerous MP steps [83]), the tested GNN architectures were limited to relatively shallow structures. Specifically all of the tested topologies had at most 5 MP steps, 5 classification head layers and latent dimensions (for each node representation) of at most 256.

GNNs take as input PCN structures directly without the need for further processing or explicit embeddings. Recall that in this specific applicative case each node has one categorical feature (i.e., its amino acid name), such label was represented either via One-Hot-Encoding (OHE) with a binary vector in $\{0, 1\}^{21}$ (20 dimensions for standard amino acids plus 1 for possibly unknown ones) or via dense learnable embeddings with a maximum of 128 elements.

After the initial data input and node features embedding graph nodes pass through MP layers. In terms of MP strategies the exploration revolved around some of the most relevant MP paradigms in literature included in the *PyTorch Geometric* (PyG) python package [34]:

1. *Graph Convolution*: first presented in Morris et al. (2019) [71] it implements a first-order, Weisfeiler–Leman–style message passing in which each node up-

dates its representation by combining a transformation of its own features with an aggregated transformation of its neighbors. For a node i with features $\mathbf{x}_i$ and (optional) scalar edge weights $e_{j,i}$, the layer computes

$$\mathbf{x}'_i \;=\; \mathbf{W}_1\,\mathbf{x}_i \;+\; \mathbf{W}_2 \sum_{j\in\mathcal{N}(i)} e_{j,i}\,\mathbf{x}_j, \tag{4.16}$$

where $\mathbf{W}_1, \mathbf{W}_2$ are learnable weight matrices and the neighborhood aggregation is permutation-invariant (e.g., sum/mean/max with sum being the common choice). This operator respects graph isomorphism symmetries and in theory matches the expressive power of the 1-WL test that characterizes many GNN architectures.

2. *Sage Convolution*: first presented in Hamilton et al. (2017) [42] it performs inductive message passing by combining a node's own representation with a permutation-invariant aggregation of its neighbors. For a node i with features $\mathbf{x}_i$ it computes

$$\mathbf{x}'_i \;=\; \mathbf{W}_1\,\mathbf{x}_i \;+\; \mathbf{W}_2 \cdot \mathrm{mean}_{j\in\mathcal{N}(i)}\mathbf{x}_j \tag{4.17}$$

where $\mathbf{W}_1, \mathbf{W}_2$ are learnable weights and the neighborhood aggregation is typically the mean. Optionally, an input projection can be applied before aggregation.

3. *GCN Convolution*: first presented in Kipf et al. (2016) [54] it implements a renormalized, symmetrically normalized neighborhood aggregation. Using an adjacency matrix with self-loops $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and degree matrix $\hat{\mathbf{D}}$ with $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$, the layer computes

$$\mathbf{X}' \;=\; \hat{\mathbf{D}}^{-\frac{1}{2}}\,\hat{\mathbf{A}}\,\hat{\mathbf{D}}^{-\frac{1}{2}}\,\mathbf{X}\,\mathbf{W} \tag{4.18}$$

equivalently, at node level,

$$\mathbf{x}'_i = \mathbf{W}^T \sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{\hat{d}_i\,\hat{d}_j}}\,\mathbf{x}_j \tag{4.19}$$

where $\mathbf{W}$ is learnable and $\hat{d}_i, \hat{d}_j$ correspond to $\hat{D}_{ii}$ and $\hat{D}_{jj}$ respectively. This

propagation enforces permutation invariance and stabilizes training via degree normalization.

4. *GIN Convolution*: first presented by Xu et al. (2019) [104] it realizes an expressive permutation-invariant aggregation by summing neighbor features with a scaled self-feature and then applying a multi-layer perceptron. The scaling coefficient ϵ can be fixed or learned, controlling the relative contribution of the central node. The canonical update is

$$\mathbf{x}_i' = h_\Theta\left((1+\epsilon)\,\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)} \mathbf{x}_j\right), \tag{4.20}$$

where h_Θ denotes an MLP. This sum-based design matches the discriminative power of the 1-WL test under suitable conditions.

5. *GAT Convolution*: first presented by Veličković et al. (2017) [99] it implements data-dependent, permutation-invariant message passing by learning attention weights over edges. For each attention head $k = 1, \ldots, K$, attention logits are computed as

$$e_{ij}^{(k)} \;=\; \mathrm{LeakyReLU}\Big(\mathbf{a}^{(k)\top}\big[\,\mathbf{W}^{(k)}\mathbf{x}_i \,\|\, \mathbf{W}^{(k)}\mathbf{x}_j\,\big]\Big) \tag{4.21}$$

normalized across the incoming neighborhood $\mathcal{N}$ of i,

$$\alpha_{ij}^{(k)} \;=\; \mathrm{softmax}_{j\in\mathcal{N}(i)\cup\{i\}}\big(e_{ij}^{(k)}\big) \tag{4.22}$$

and used to aggregate neighbor features,

$$\mathbf{z}_i^{(k)} \;=\; \sum_{j\in\mathcal{N}(i)\cup\{i\}} \alpha_{ij}^{(k)}\,\mathbf{W}^{(k)}\mathbf{x}_j \tag{4.23}$$

The layer output is obtained by combining the K heads either by concatenation

or averaging,

$$\mathbf{x}_i' = \begin{cases} \Big\|_{k=1}^{K} \mathbf{z}_i^{(k)} & \text{(concatenation)} \\ \frac{1}{K}\sum_{k=1}^{K} \mathbf{z}_i^{(k)} & \text{(average)} \end{cases} \tag{4.24}$$

optionally with dropout on features and attention coefficients. The PyG implementation follows this formulation and typically includes self-loops in the graphs so that the central node performs self-attention during MP.

Notably each GNN topology exploits only one of the aforementioned strategies at a time with no change in the dimensions of node representations after any MP layer besides the first which projects node representations up to the chosen hidden dimensions of the network.

After MP the next important step of a GNN for graph classification is *graph pooling* which aggregates the node representations into a single vector corresponding to a *graph latent representation*.

In the context of this work pooling was carried out either via permutation invariant aggregations (max/mean/sum) or via Attentional Aggregation. Attentional Aggregation is a pooling method presented by Li et al. (2019) [57] which performs soft attention over nodes, computing a convex combination of transformed features with data-dependent weights. Given elements $\{\mathbf{x}_n\}_{n=1}^{N_i}$ belonging to group i, the aggregation is

$$\mathbf{r}_i = \sum_{n=1}^{N_i} \mathrm{softmax}\big(h_{\mathrm{gate}}(\mathbf{x}_n)\big) \cdot h_{\Theta}(\mathbf{x}_n) \tag{4.25}$$

where $h_{\mathrm{gate}} : \mathbb{R}^F \to \mathbb{R}$ (F being the dimensionality of the node representations) produces un-normalized attention scores that are then normalized across the elements in group i, and h_{Θ} is a learnable transformation (typically a MLP) applied to each element before weighting. The softmax ensures permutation invariance and normalizes the sum to one within each group. In this particular application the entirety of the nodes in the graph was considered a single group to be aggregated

The last step after pooling is the actual classification step carried out thanks to a typical MLP structure.

Table 4.4 shows a summary of the all of the parameters that were used to define

candidate GNN structures, notably parameters presenting a † after their name are conditional parameters that were defined only if other parameters presented specific values (e.g., # GAT Heads was defined only for topologies using GAT Convolution). The presented topologies were explored efficiently thanks to the bayesian optimization strategy presented in Section 4.10.

Parameter Name	Type	Values	Notes
Data Embedding Type	Cat	OHE, dense	Type of Embedding for Amino Acids
Embedding Dimensions †	Int	[8, 128]	Size of Dense Embeddings
Activation Type	Cat	ReLU, $\mathrm{LeakyReLU}$	Non Linear Activation Function for the GNN
Negative Slope †	Float	[0.01, 0.3]	Slope for Negative Inputs in $\mathrm{LeakyReLU}$
MP Strategy	Cat	Graph, Sage, GCN, GIN, GAT	
# MP Layers	Int	[1,5]	Upper Bound Reduced to 3 for GAT MP [2]
Normalization Type	Cat	Graph, GraphSize, Batch, Layer, None	Type of Normalization in MP Layers
Hidden Dimensions	Int	[64, 256]	Upper Bound Reduced to 128 for GAT MP [2]
Pooling Type	Cat	Mean, Sum, Max, Attention	Type of Graph Pooling
After Pooling Norm.	Cat	Batch, Layer, None	Normalization Used After Pooling
# MLP Layers	Int	[0, 5]	
MLP Dropout	float	[0, 0.5]	Dropout for the Classification Head
# GAT Heads †	Int	$\{2, 4, 8\}$	Defined only for GAT MP
# GIN MLP Layers †	Int	[1,3]	Defined only for GIN MP

Table 4.4: Summary of Parameters for Candidate GNN Topologies

[2]Upper-bounds for parameters were modified to handle computational needs of GAT Convolution

4.9 Performance Metrics

A core aspect of any ML application are performance metrics which are essential to understand the real capabilities of a specific model and consequently to compare different approaches.

Considering the fact that the experimental framework of this work is based on classification tasks the most complete and informative performance metric is by definition the *confusion matrix* (CM). The CM of a binary classifier can be defined as in Table 4.5

	Pred. Positive	**Pred. Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 4.5: Binary Confusion Matrix

whose entries are:

TP: Observations correctly predicted as positive cases (True Positives)

FN: Positive cases incorrectly predicted as negative (False Negatives)

FP: Negative cases incorrectly predicted as positive (False Positives)

TN: Observations correctly predicted as negative cases (True Negatives)

In a multiclass setting the CM can be easily generalized by creating a matrix with dimensions equal to the number of distinct classes in which the diagonal presents correctly classified observations and other values represent misclassified observations. The most relevant limit of the CM is that it is difficult to numerically evaluate a model from its CM directly.

Starting from the entries of a CM it is possible to define most of the standard performance metrics used for classification tasks, the most relevant ones being:

Accuracy: defined as $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \in [0,1]$. It represents the fraction of observations correctly classified by the model

Precision: defined as $\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \in [0,1]$. It represents the fraction of actually positive instances among the predicted positive instances. (Also known as *Positive Predictive Value*)

Recall: defined as $\frac{\text{TP}}{\text{TP+FN}} \in [0, 1]$. It represents the fraction of actually positive values correctly predicted by the model. (Also known as *Sensitivity*)

F1-Score: defined as $2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}} \in [0, 1]$. It is the harmonic mean of precision and recall and is often capable of conveying more information about model performance despite being harder to interpret.

Bal. Accuracy: considering C the number of classes in the dataset Balanced Accuracy is defined as $\frac{1}{C}\sum_{i=1}^{C}\frac{\text{TP}_i}{\text{TP}_i+\text{FN}_i} \in [0, 1]$. It represents the macro average of Recall across all classes in the dataset. It avoids inflated performances in the case of imbalanced datasets which makes it often preferable to standard Accuracy. Balanced Accuracy can be adjusted for chance and is reformulated as

$$\frac{\frac{1}{C}\sum_{i=1}^{C}\frac{\text{TP}_i}{\text{TP}_i+\text{FN}_i} - \frac{1}{C}}{1-\frac{1}{C}} \in [1/(1-C), 1] \tag{4.26}$$

The presented performance measures can be easily extended to multiclass scenarios by averaging their binary version over all available classes. Adjusted Balanced Accuracy (ABA), specifically the formulation of Equation (4.26), is naturally suited for multiclass scenarios and also avoids performance inflation on imbalanced datasets. For these reasons it was used as main performance metric throughout this work, specifically validation ABA was used as objective function value for the selection of the best set of hyper-parameters for all tested models (more details in Section 4.10).

4.10 Hyperparameter Optimization Strategy

Hyper-parameter selection in this work was formalized as the global optimization (minimization was used for presentation purposes in this section) of a black-box objective $f : \mathcal{X} \to \mathbb{R}$ over a mixed (continuous, integer, categorical) search space $\mathcal{X}$. The optimal set of parameters $x^\star$ can be defined as:

$$x^\star \in \arg\min_{x\in\mathcal{X}} f(x) \tag{4.27}$$

where each evaluation of f corresponds to training and validating a model under a specific hyper-parameter configuration. The adopted approach belongs to the *Bayesian Optimization* (BO) paradigm, a class of sequential design strategies that fit a probabilistic surrogate $\mathcal{M}$ to the history $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ (with $y_i = f(x_i)$) and select the next candidate by maximizing an *acquisition function* $\alpha(x; \mathcal{D}_n, \mathcal{M})$ balancing exploration and exploitation [89]. Classical BO instantiates $\mathcal{M}$ as a *Gaussian Process* (GP) and often uses the *Expected Improvement* (EI) acquisition [51]. Under a GP posterior with mean $\mu(x)$, standard deviation $\sigma(x)$ and letting $y^\star = \min_{i \le n} y_i$ (in the minimization case), EI takes the form

$$\mathrm{EI}(x) = (y^\star - \mu(x))\Phi\left(z(x)\right) + \sigma(x)\phi\left(z(x)\right) \tag{4.28}$$

where $z(x) = \frac{y^\star - \mu(x)}{\sigma(x)}$, Φ is the standard normal *Cumulative Distribution Function* (CDF) and ϕ the related *Probability Density Function* (PDF) [91]. Intuitively standard BO tries to model $p(y|x)$ from the history of past trials.

In this work BO was implemented via multivariate *Tree-Structured Parzen Estimators* (TPE): a sequential model-based optimizer that replaces the usual $p(y|x)$ modelling with a more convenient density estimation of $p(x|y)$, yielding a sampling rule closely related to EI while being naturally compatible with mixed and conditional search spaces [1, 6]. Its multivariate variant further models joint dependencies among hyper-parameters, improving search efficiency when strong interactions are present [46].

4.10.1 TPE and Bayesian Optimization

Standard BO selects x_{n+1} by maximizing EI using an explicit surrogate of $p(y|x)$ (e.g. the GP) [89]. TPE inverts this perspective and directly models the conditional densities of configurations given performance. Fix a quantile level $\gamma \in (0, 1)$ and the corresponding performance threshold $y^\star$ such that $P(y \le y^\star) = \gamma$. Partition the observations into the *good* and *bad* sets as follows:

$$\mathcal{D}^{\mathrm{good}} = \{x_i : y_i \le y^*\}, \qquad \mathcal{D}^{\mathrm{bad}} = \{x_i : y_i > y^*\}$$

TPE then builds two kernel density estimates (KDEs),

$$\ell(x) \approx p(x \mid y \leq y^*) = \frac{1}{|\mathcal{D}^{\text{good}}|} \sum_{x_i \in \mathcal{D}^{\text{good}}} K_h(x - x_i), \tag{4.29}$$

$$g(x) \approx p(x \mid y > y^*) = \frac{1}{|\mathcal{D}^{\text{bad}}|} \sum_{x_i \in \mathcal{D}^{\text{bad}}} K_h(x - x_i), \tag{4.30}$$

with kernels $K_h(\cdot)$ adapted to each parameter type (Gaussian for continuous, Aitchison–Aitken or categorical histograms for discrete) [6]. Using Bayes' rule, the EI objective:

$$\text{EI}(x) = \int_{-\infty}^{y^\star} (y^\star - y)\, p(y \mid x) dy \tag{4.31}$$

can be rewritten (details in [6]) as:

$$\text{EI}(x) \propto \left(\gamma + (1-\gamma)\,\frac{g(x)}{\ell(x)}\right)^{-1} \tag{4.32}$$

Maximizing Equation (4.32) is equivalent to minimizing $g(x)/\ell(x)$. Consequently, TPE proposes candidates by sampling from $\ell(x)$ and selecting those with the smallest ratio $g(x)/\ell(x)$.

An advantage of TPE is its compatibility with *tree-structured* search spaces, where some hyper-parameters are active only when certain categorical choices are made (as in Table 4.4). The densities in Equations (4.29) and (4.30) are built along the active branches, avoiding the need to impute values for inactive variables.

The original TPE assumes independence across dimensions by factorizing the KDE as a product of univariate kernels, this can underperform when the hyper-parameters exhibit strong *interactions* among each other. The multivariate TPE variant replaces the factorized model with a joint KDE over the active subspace of parameters described as:

$$\ell(x) \;=\; \frac{1}{n_{\text{g}}} \sum_{i=1}^{n_{\text{g}}} \mathcal{K}_H(x - x_i), \qquad \mathcal{K}_H(u) = |H|^{-1/2}\, K\big(H^{-1/2}u\big) \tag{4.33}$$

(and analogously for $g(x)$), where H is a bandwidth (covariance) matrix and K a multivariate base kernel. Joint KDEs capture cross-parameter correlations directly, improving the fidelity of the ratio in (4.32) and thus the quality of the acquisition.

In summary the optimization pipeline of this work consisted of a TPE-based BO loop (as in Eqs. 4.31–4.32) exploiting a multivariate TPE process implemented in the `Optuna` Python package [1]. This choice provides (i) intelligent exploration of the possibly very large parameter spaces limiting the number of necessary trials, (ii) native support for mixed and conditional hyper-parameters and (iii) improved performances in the presence of hyperparameter interactions thanks to joint KDE modelling.

4.11 Data Resampling and Splitting Strategy

In both experimental settings—Task A (enzyme vs. non-enzyme) and Task B (first-level EC-class classification)—the corresponding dataset was partitioned into five, mutually exclusive, stratified folds to preserve label proportions within each split. A five-run protocol was then executed per task and per model. In each run, a different fold acted as the hold-out test set, one of the remaining four folds served as the validation set for hyperparameter selection and the remaining three folds constituted the training set. Consequently, each instance in the dataset appeared exactly once in the test set and up to four times in the union of training and validation sets, yielding a nominal $60\%/20\%/20\%$ train/validation/test partition per run.

Hyperparameter optimization, as described in Section 4.10, was performed independently for every run using only the training folds and the corresponding validation fold of each run. The selected configuration was then assessed on the run's test fold, which remained untouched during model selection. This design eliminates information leakage from test data into model tuning and ensures that performance estimates reflect true generalization.

The same five fold indices were fixed across all representation strategies and learning algorithms to enable paired, like-for-like comparisons. The procedure provides five independent test scores per model and per task; considering the mean and standard deviation over these scores offers a more robust and variance-aware summary of performance than a single split, while allowing a fair evaluation of both the representation strategies and the models at the best of their capabilities.

Note that in the case of the representation strategies involving simplicial complexes,

a distinct dictionary of simplices d was generated for each data split. The dictionary was defined as the union of simplices appearing in the training and validation folds and the resulting symbolic histograms were calculated according to such dictionary also for the test fold.

Chapter 5

Results

This section reports the performances of the tested methods together with the main outcomes of the analysis. The two experimental tasks, Task A and Task B, are presented separately in Sections 5.1 and 5.2, respectively.

All tables in this section will present performance metrics averaged over the 5 distinct data splits (see Section 4.11) in the form avg ± std unless differently specified.

5.1 Task A

Recall that Task A consists of a binary classification problem aimed at distinguishing enzymatic from non-enzymatic proteins. The dataset for this task includes the entirety of the curated human proteome, comprising 48,019 protein structures divided into 26,312 non-enzymatic and 21,707 enzymatic ones.

5.1.1 Spectral Density Embedding

Table 5.1 shows the performances for all models working with the spectral density embedding of PCNs in Task A. The RF model has the best training performances however it exhibits serious overfitting making ν-SVM the best performer in both Validation and Testing.

The ℓ_1-Lin-SVM classifier exhibits substantially weaker performance compared to the other models. This behavior can be largely attributed to the characteristics of the

Model	Metric	Train	Validation	Test
ν-SVM	ABA	0.942 ± 0.021	0.749 ± 0.004	0.745 ± 0.006
	ACC	0.970 ± 0.010	0.876 ± 0.002	0.874 ± 0.003
	F1	0.967 ± 0.011	0.863 ± 0.002	0.861 ± 0.003
ℓ_1-Lin-SVM	ABA	0.356 ± 0.005	0.351 ± 0.010	0.356 ± 0.009
	ACC	0.678 ± 0.003	0.675 ± 0.005	0.677 ± 0.004
	F1	0.658 ± 0.003	0.655 ± 0.004	0.657 ± 0.004
RF	ABA	0.975 ± 0.008	0.715 ± 0.007	0.721 ± 0.009
	ACC	0.986 ± 0.004	0.857 ± 0.004	0.859 ± 0.004
	F1	0.985 ± 0.005	0.845 ± 0.003	0.848 ± 0.005

Table 5.1: Performances on Spectral Embedding for all Models - Task A

spectral density embedding. Since all graph spectra are supported in the range $[0, 2]$, the KDE procedure samples exactly the same 200 evaluation points for every protein. This design ensures comparability across spectra but also introduces strong limitations. A prime example would be the Gaussian kernel smoothing enforcing continuity so, as a consequence, most estimated densities share very similar global shapes.

Under these conditions, KDE evaluations at two close points $x_1, x_2 \in [0, 2]$ will yield nearly identical values. Hence, adjacent columns of the instance matrix are expected to be highly linearly correlated. Figure 5.1 confirms this phenomenon by showing the correlation structure of $\mathbf{X}_A^{(\mathrm{SPECTRAL})}$: not only consecutive columns, but also columns near the spectral boundaries display high pairwise correlation.

Because ℓ_1-Lin-SVM is by definition a linear classifier, the presence of strong linear collinearity in the input features degrades its ability to identify sparse and discriminative subsets of variables. This explains the markedly inferior performance observed in comparison to the non-linear ν-SVM and the tree-based RF model.

5.1.2 Simplicial Complexes Embedding

Table 5.2 highlights the performances for all models working with the Simplicial Complexes symbolic histogram embedding of PCNs in Task A. ν-SVM appears again as the most powerful model among the tested ones however in this setting all classifiers perform quite similarly with ν-SVM taking the lead by a very small margin. ℓ_1-Lin-SVM works especially well in this setting with a very large number of features as its implicit

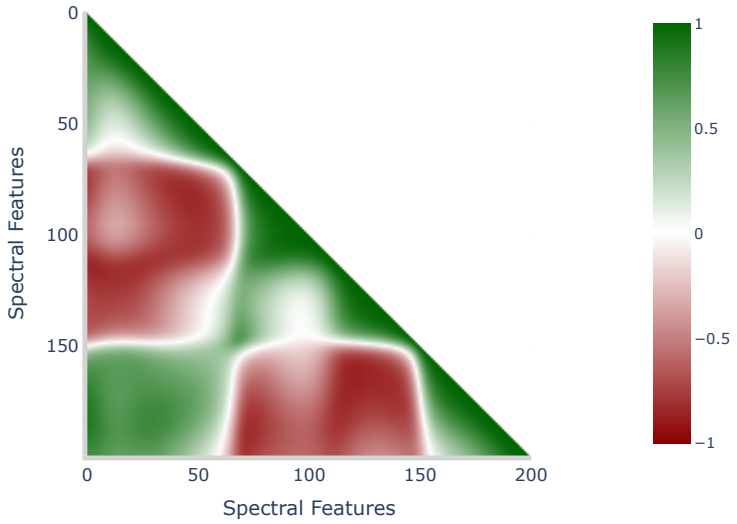


Figure 5.1: Pearson Correlation of Spectral Features - Task A

feature selection allows for the sparsification of the solution shortening running times and making predictions less noisy. Despite being a purely linear model ℓ_1-Lin-SVM performs only 0.7% worse than ν-SVM and takes a fraction of the training time while simultaneously ignoring on average 78% of the input features in the process.

Model	Metric	Train	Validation	Test
ν-SVM	ABA	0.982 ± 0.006	0.875 ± 0.004	0.874 ± 0.003
	ACC	0.991 ± 0.003	0.938 ± 0.002	0.937 ± 0.002
	F1	0.991 ± 0.003	0.932 ± 0.002	0.931 ± 0.002
ℓ_1-Lin-SVM	ABA	0.959 ± 0.008	0.869 ± 0.002	0.868 ± 0.002
	ACC	0.980 ± 0.004	0.935 ± 0.001	0.934 ± 0.001
	F1	0.978 ± 0.004	0.928 ± 0.001	0.928 ± 0.001
RF	ABA	0.954 ± 0.008	0.877 ± 0.005	0.873 ± 0.006
	ACC	0.978 ± 0.004	0.942 ± 0.002	0.940 ± 0.003
	F1	0.976 ± 0.004	0.934 ± 0.003	0.931 ± 0.003

Table 5.2: Performances on Simplicial Complexes Embedding for all Models - Task A

As discussed in Section 4.6, RF models are inherently able to evaluate the importance of input features with the MDI method. ℓ_1-Lin-SVM, on the other hand, are capable of deleting the least relevant features from the input data. These capabilities, combined with the fact that each feature in the simplicial complexes embedding represents the number of times a specific local sub-structure appears in a PCN, make it

possible to analyze which sub-structures are more relevant when trying to separate Enzymatic and Non-Enzymatic proteins.

Figure 5.2 presents the top 10 most relevant features according to RF models. RF feature importance scores were averaged across all 5 runs and normalized to 1. Figure 5.3 shows a similar plot on the hyperplane coefficients coming from ℓ_1-Lin-SVM: only features which were never removed in any of the five splits were considered, the absolute value of their coefficients was afterwards averaged and normalized to 1. With this approach it is possible to get a plot similar to Figure 5.2 giving insights on the relevance of features for the ℓ_1-Lin-SVM model.

It is interesting to note how, despite RF and ℓ_1-Lin-SVM being two very different models with nothing in common among their formulations, the same simplex *ASP-ASP-HIS* appears among the most relevant ones for both the models. The independent identification of said sub-structure by both classifiers suggests that this configuration might reflect a biologically meaningful structural or functional signature.

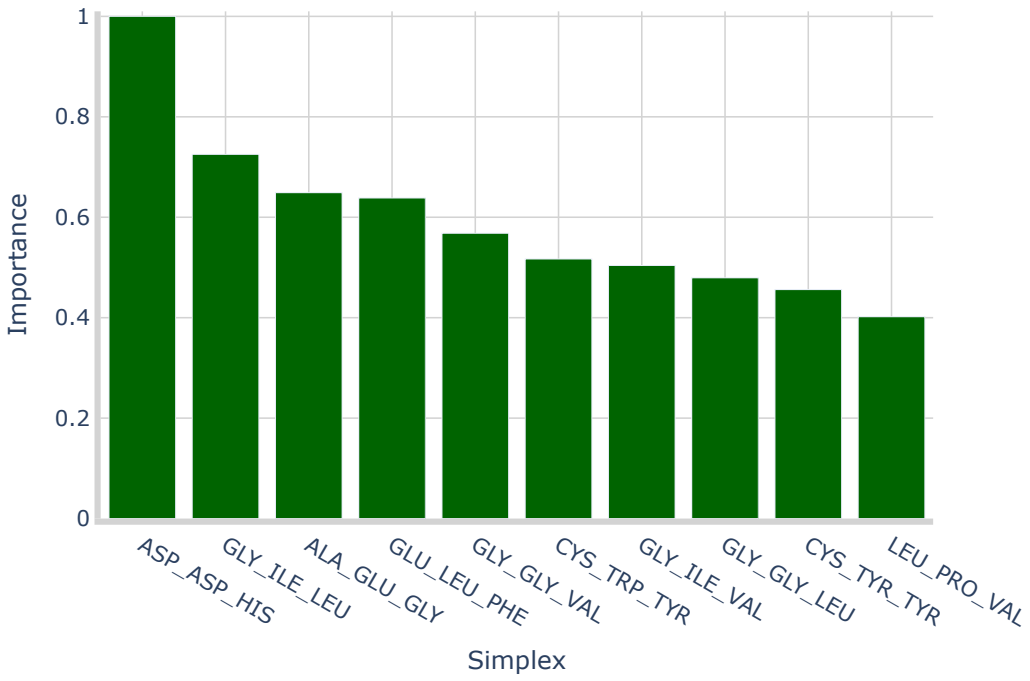


Figure 5.2: Top 10 Most Relevant Simplices According to RF MDI on Simplicial Complexes Embedding - Task A

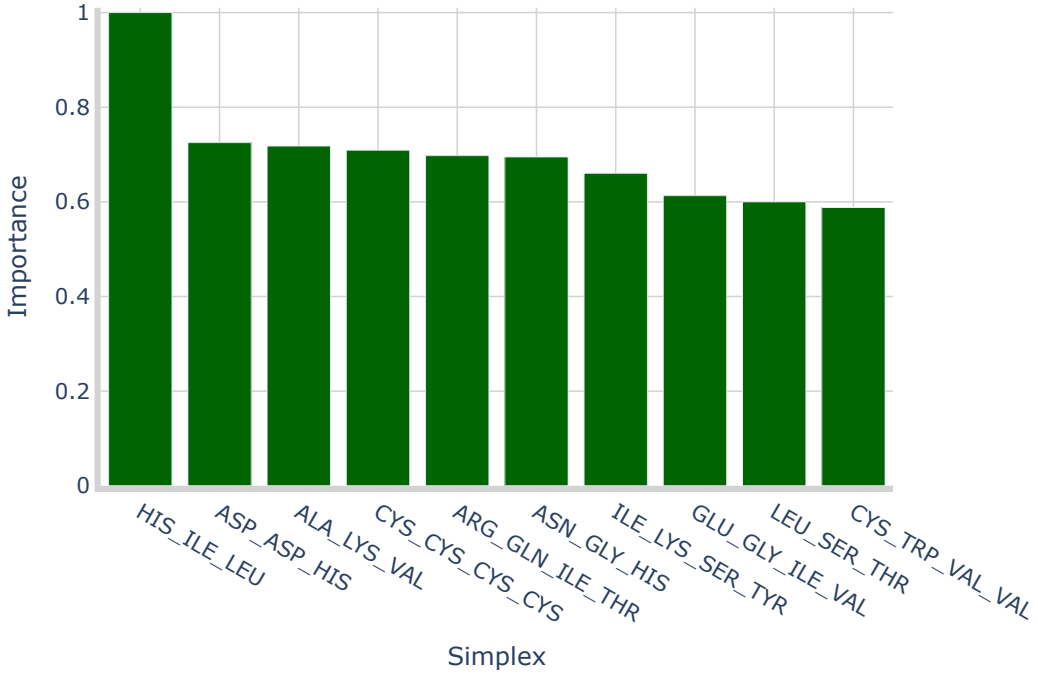


Figure 5.3: Top 10 Most Relevant Simplices According to ℓ_1-Lin-SVM Coefficients on Simplicial Complexes Embedding - Task A

5.1.3 INDVAL Embedding

Table 5.3 shows the performances for all models when paired with the INDVAL embedding in Task A. In this setting the RF model is the best performer with a test ABA 1.7% higher on average compared to ν-SVM. ℓ_1-Lin-SVM underperforms slightly compared to the other methods, this is probably due to the feature vector being much smaller ($\sim 10\%$ of the Simplicial Complexes Embedding) and already pre-filtered according to the INDVAL Score. The conditions combined with ℓ_1-Lin-SVM algorithm pushing coefficients to zero in this context was probably sub-optimal and resulted in the removal of potentially relevant variables.

Since features in the INDVAL Embedding are a filtered sample of features in the Simplicial Complexes embedding it is possible to conduct the same exact procedure as in the previous section when it comes to feature importance scores.

Figures 5.4 and 5.5 present the top 10 most relevant features for both RF and ℓ_1-Lin-SVM. In this case there are no common simplices among the two models however it is

Model	Metric	Train	Validation	Test
ν-SVM	ABA	0.986 ± 0.003	0.875 ± 0.004	0.871 ± 0.002
	ACC	0.993 ± 0.002	0.937 ± 0.002	0.935 ± 0.001
	F1	0.992 ± 0.002	0.931 ± 0.002	0.929 ± 0.001
ℓ_1-Lin-SVM	ABA	0.850 ± 0.006	0.806 ± 0.002	0.803 ± 0.005
	ACC	0.925 ± 0.003	0.904 ± 0.001	0.902 ± 0.002
	F1	0.918 ± 0.003	0.894 ± 0.001	0.892 ± 0.003
RF	ABA	0.979 ± 0.006	0.889 ± 0.004	0.886 ± 0.003
	ACC	0.990 ± 0.003	0.947 ± 0.002	0.945 ± 0.002
	F1	0.989 ± 0.003	0.940 ± 0.002	0.938 ± 0.002

Table 5.3: Performances on INDVAL Embedding for all Models - Task A

interesting how the most important sub-structure according to RF is always *ASP-ASP-HIS* even after INDVAL feature selection. This phenomenon shows how such 3-simplex is relevant both in terms of INDVAL Score (as it surpasses the imposed threshold) and from a discriminative point of view (as it is consistently relevant across models)

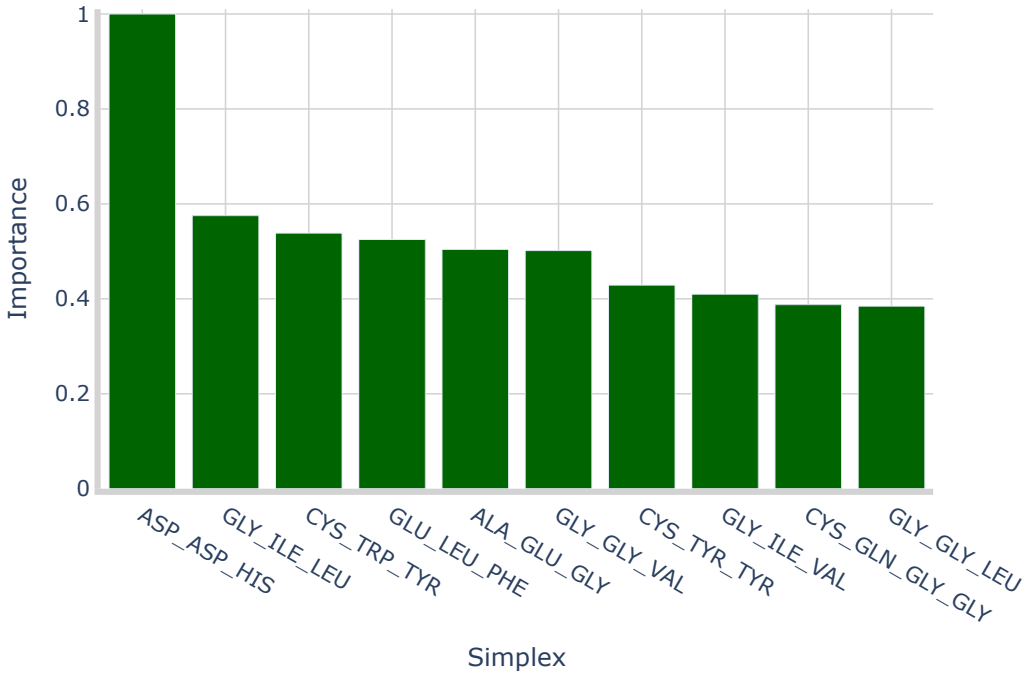


Figure 5.4: Top 10 Most Relevant Simplices According to RF MDI on INDVAL Embedding - Task A

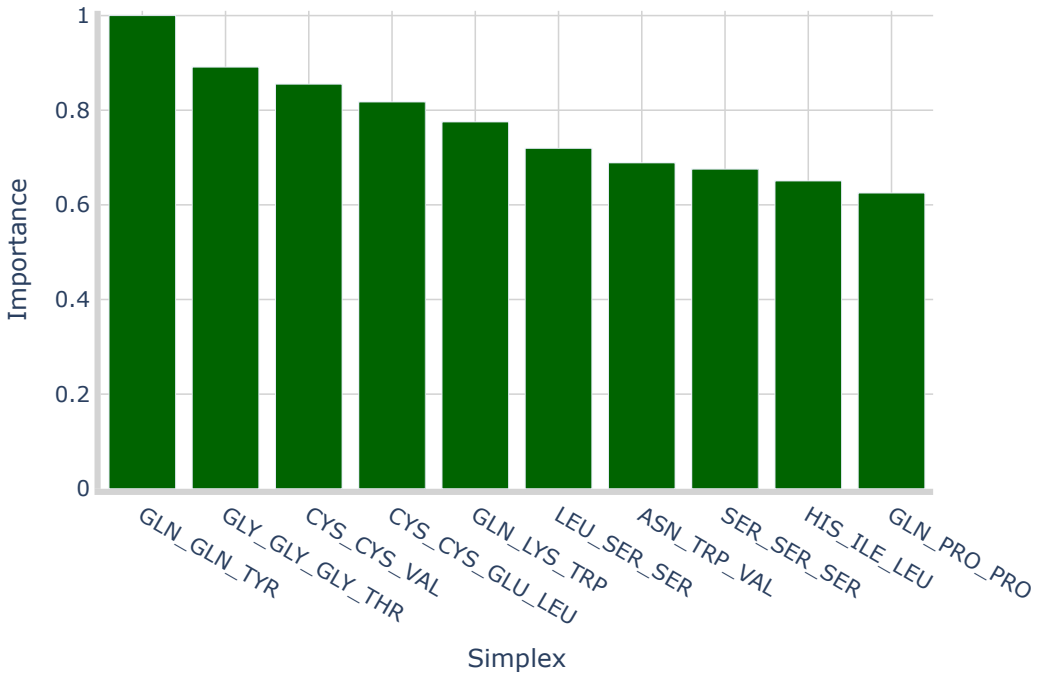


Figure 5.5: Top 10 Most Relevant Simplices According to ℓ_1-Lin-SVM Coefficients on INDVAL Embedding - Task A

5.1.4 (Hyper)Graph Kernels

Table 5.4 shows ν-SVM performances on both explored kernel methods in Task A. No other classification model was tested for kernel representations as ν-SVM was the only one that supported pre-computed kernels as inputs (see Section 4.6.2). Both kernels perform remarkably well, the JK especially exhibits the best results reaching ABA of 0.900 on the test set.

Model	Kernel	Metric	Train	Validation	Test
ν-SVM	HCK	ABA	0.954 ± 0.007	0.882 ± 0.003	0.879 ± 0.003
		ACC	0.977 ± 0.003	0.941 ± 0.002	0.939 ± 0.002
		F1	0.975 ± 0.004	0.935 ± 0.002	0.933 ± 0.002
	JK	ABA	0.994 ± 0.005	0.902 ± 0.003	0.900 ± 0.002
		ACC	0.997 ± 0.003	0.952 ± 0.002	0.951 ± 0.001
		F1	0.996 ± 0.003	0.946 ± 0.002	0.945 ± 0.001

Table 5.4: Performances on Kernel Methods — Task A

The kernel-based approaches confirm the effectiveness of non-linear similarity mea-

sures between symbolic histograms. In particular, the Jaccard Kernel consistently outperformed the Histogram Cosine Kernel across validation and test sets suggesting that Jaccard-style similarity is better suited for PCN-based simplicial histograms. It is important to acknowledge that, compared to explicit embeddings, the computational limitations of kernel methods are relevant. For example the construction of full Gram matrices scales quadratically with the number of proteins and their interpretability is reduced compared to explicit embeddings where feature importances can be directly inspected.

5.1.5 Binary GNN

Table 5.5 presents GNN performances for Task A. A different GNN topology has been optimized for each split of the dataset considering hyper-parameter candidates presented in Table 4.4. From these 5 separate optimization some clear trends have emerged in the optimal structures. 80% of the optimal topologies leveraged one-hot encoding for node labels and *graph convolution* for MP while all of them leveraged *max pooling* and batch normalization in MP layer. None of the optimal models leverages after pooling normalization and MLP heads have at most 3 layers. The hidden dimensions were close to the upper bound of the relative search space, 60% of the topologies selected 224 as optimal values while the others selected 192.

Model	Metric	Train	Validation	Test
GNN	ABA	0.963 ± 0.004	0.902 ± 0.005	0.898 ± 0.005
	ACC	0.981 ± 0.002	0.951 ± 0.002	0.949 ± 0.003
	F1	0.979 ± 0.002	0.946 ± 0.002	0.944 ± 0.003

Table 5.5: GNN Architecture Performances - Task A

5.1.6 Summary of Task A

Table 5.6 synthesizes the comparative results for the binary enzyme vs. non-enzyme classification task and three main conclusions emerge. First, spectral density embeddings under-perform across models (most markedly for ℓ_1-Lin-SVM) due to the strong

collinearity induced by KDE sampling similar distributions on a fixed grid, which undermines sparse linear discrimination. Second, representations based on symbolic histograms of simplicial complexes, with or without INDVAL selection, are consistently strong and stable across models: RF and ν-SVM achieve competitive performance, and the linear ℓ_1-Lin-SVM becomes viable in this high-dimensional, sparse regime thanks to its embedded feature selection. Third, non-linear similarity functions on the symbolic histograms yield the highest accuracies among all pipelines: the JK attains the top average test performance (testing ABA of 0.900), slightly surpassing the HCK, albeit at the cost of quadratic Gram-matrix construction and reduced interpretability compared to explicit embeddings.

End-to-end deep learning on raw PCNs via GNNs performs almost on par with the best kernel approach (testing ABA of 0.898), offering a compelling alternative that forgoes handcrafted features while preserving competitive generalization. Overall, Task A can be addressed effectively by both classical ML and DL approaches. When balancing accuracy and interpretability, simplicial-complex embeddings combined with tree-based or margin-based classifiers offer an excellent trade-off; when maximizing absolute accuracy with fixed training sizes, the JK is marginally superior while when prioritizing end-to-end learning and feature minimalism, GNNs provide a scalable solution with comparable performances.

Representation Strategy	ν-SVM	ℓ_1-Lin-SVM	RF	GNN
Spectral Density	0.745 ± 0.006	0.356 ± 0.009	0.721 ± 0.009	-
Simplicial Complexes	0.874 ± 0.003	0.868 ± 0.002	0.873 ± 0.006	-
INDVAL	0.871 ± 0.002	0.803 ± 0.005	0.886 ± 0.003	-
Histogram Kernel	0.879 ± 0.003	-	-	-
Jaccard Kernel	**0.900 ± 0.002**	-	-	-
Raw PCNs	-	-	-	0.898 ± 0.005

Table 5.6: Test Set ABA for all Models and Representation Strategies - Task A

Table 5.7 presents the test set confusion matrix of the ν-SVM model on JK as a final intuitive visualization of the maximal performances attained in Task A

		Predicted	
		Non-Enzyme	Enzyme
True	Non-Enzyme	5046	216
	Enzyme	246	4095

Table 5.7: Best Performer Confusion Matrix - Task A

5.2 Task B

Task B consists of a multiclass classification problem focused on assigning the correct first-level EC number to enzymatic proteins. The dataset for this task includes 21,679 enzymatic protein structures (after filtering out Translocases due to their scarcity), each annotated with a single first-level EC class.

5.2.1 Spectral Density Embedding

Table 5.8 presents the multiclass performances for all models combined with Spectral Embedding in Task B. Compared to the Task A scenario the ν-SVM model evidently outperforms the other two candidates and confirms itself as the best model for enzyme classification based on spectral density also in a multiclass setting.

Model	Metric	Train	Validation	Test
ν-SVM	ABA	0.992 ± 0.015	0.747 ± 0.012	0.733 ± 0.012
	ACC	0.998 ± 0.003	0.862 ± 0.004	0.859 ± 0.003
	F1	0.998 ± 0.003	0.861 ± 0.004	0.858 ± 0.004
ℓ_1-Lin-SVM	ABA	0.426 ± 0.003	0.433 ± 0.012	0.423 ± 0.009
	ACC	0.570 ± 0.004	0.572 ± 0.009	0.567 ± 0.004
	F1	0.566 ± 0.004	0.567 ± 0.009	0.563 ± 0.004
RF	ABA	0.955 ± 0.018	0.701 ± 0.011	0.686 ± 0.005
	ACC	0.931 ± 0.026	0.814 ± 0.011	0.809 ± 0.015
	F1	0.932 ± 0.026	0.816 ± 0.010	0.811 ± 0.014

Table 5.8: Performances on Spectral Embedding for all Models – Task B

The ℓ_1-Lin-SVM model performances are stained by high linear correlation in the input dataset also in the multiclass scenario, Figure 5.6 shows the Pearson correlation matrix for dataset $\mathbf{X}_B^{(\mathrm{SPECTRAL})}$ which is extremely similar to the correlation matrix presented in Figure 5.1 for $\mathbf{X}_A^{(\mathrm{SPECTRAL})}$. This phenomenon is to be expected as $\mathbf{X}_B^{(\mathrm{SPECTRAL})}$ is effectively a filtered version of $\mathbf{X}_A^{(\mathrm{SPECTRAL})}$ where rows regarding

Non-Enzymatic proteins and Translocases were removed.

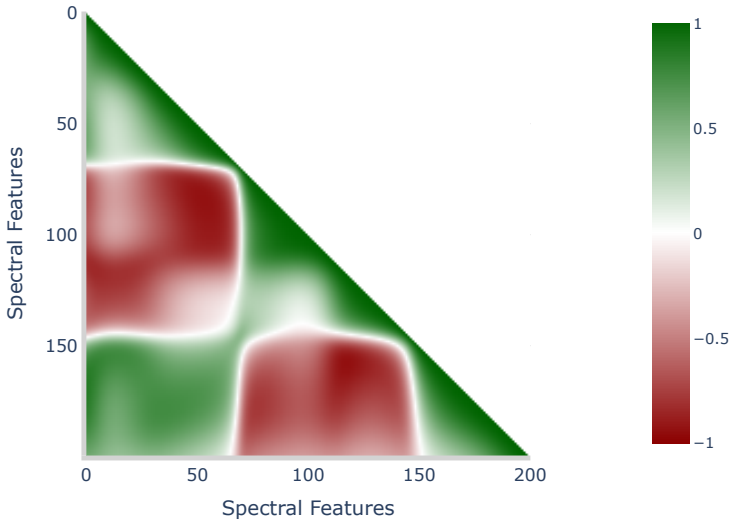


Figure 5.6: Pearson Correlation of Spectral Features - Task B

5.2.2 Simplicial Complexes Embedding

Table 5.9 shows the performances of all models working with the simplicial complexes embedding in Task B. Such results highlight the effectiveness of linear models in high-dimensional and sparse feature spaces. Despite its simplicity, the ℓ_1-Lin-SVM consistently emerges as the best performing classifier, surpassing both non-linear approaches such as ν-SVM and more flexible ensemble methods like Random Forests. This outcome shows off how the implicit feature selection mechanism induced by the ℓ_1 regularization is highly effective when dealing with symbolic histograms derived from clique hypergraphs. By enforcing sparsity in the weight vector, the model discards a large fraction of non-informative simplices while retaining the most discriminative substructures, which translates into superior generalization capabilities. The number of simplices deemed as irrelevant by ℓ_1-Lin-SVM averages to $\sim 10,600$ out of the $\sim 13,000$ in the dataset which means that approximately 81% of the features were ignored for classification purposes.

From a feature importance perspective, the same methodology adopted for Task A

Model	Metric	Train	Validation	Test
ν-SVM	ABA	0.998 ± 0.002	0.876 ± 0.011	0.864 ± 0.009
	ACC	0.999 ± 0.000	0.954 ± 0.002	0.949 ± 0.003
	F1	0.999 ± 0.000	0.954 ± 0.002	0.948 ± 0.003
ℓ_1-Lin-SVM	ABA	0.995 ± 0.003	0.905 ± 0.012	0.902 ± 0.013
	ACC	0.992 ± 0.004	0.956 ± 0.006	0.953 ± 0.006
	F1	0.992 ± 0.004	0.956 ± 0.005	0.954 ± 0.006
RF	ABA	0.992 ± 0.006	0.891 ± 0.009	0.882 ± 0.009
	ACC	0.989 ± 0.007	0.950 ± 0.006	0.946 ± 0.005
	F1	0.990 ± 0.006	0.950 ± 0.006	0.946 ± 0.005

Table 5.9: Performances on Simplicial Complexes Embedding for all Models – Task B

can be applied also for Task B to identify the most relevant substructures within the simplicial complexes embedding. Figure 5.7 presents the 10 most relevant simplices in the dataset according to the RF model, it is interesting to see how the 3-simplex *ASP-ASP-HIS* consistently emerges as the most influential feature across both classification scenarios, reinforcing its role as a potentially critical structural motif for enzymatic function recognition. In addition the identification of common motifs in both binary and multiclass settings underlines the interpretability advantage of representation strategies directly based on structural patterns.

When it comes to ℓ_1-Lin-SVM the feature importance analysis for a multiclass scenario becomes less intuitive: in accordance with the OvR strategy a different classifier is built for each class in the dataset and each of the classifiers is a completely independent model. At inference time all the models make their predictions and the one having the strongest prediction is the one effectively assigning the label.

Figure 5.8 shows feature importance plots similar to Figure 5.3 for each of the 6 distinct EC Classes in the dataset for Task B. These per-class coefficients must be interpreted within the OvR setting: each panel reflects a classifier trained against all remaining classes, hence coefficients encode class-specific discriminants and are not directly comparable across classes nor to RF MDI scores. In practice, the ℓ_1 penalty enforces marked sparsity and the vast majority of simplices receive zero weight so only a small class-relevant subset carries discriminative power. The patterns highlighted by ℓ_1-Lin-SVM complement those highlighted by RF indicating both shared and class-dependent structural signals captured by models with different classification strategies. Since each

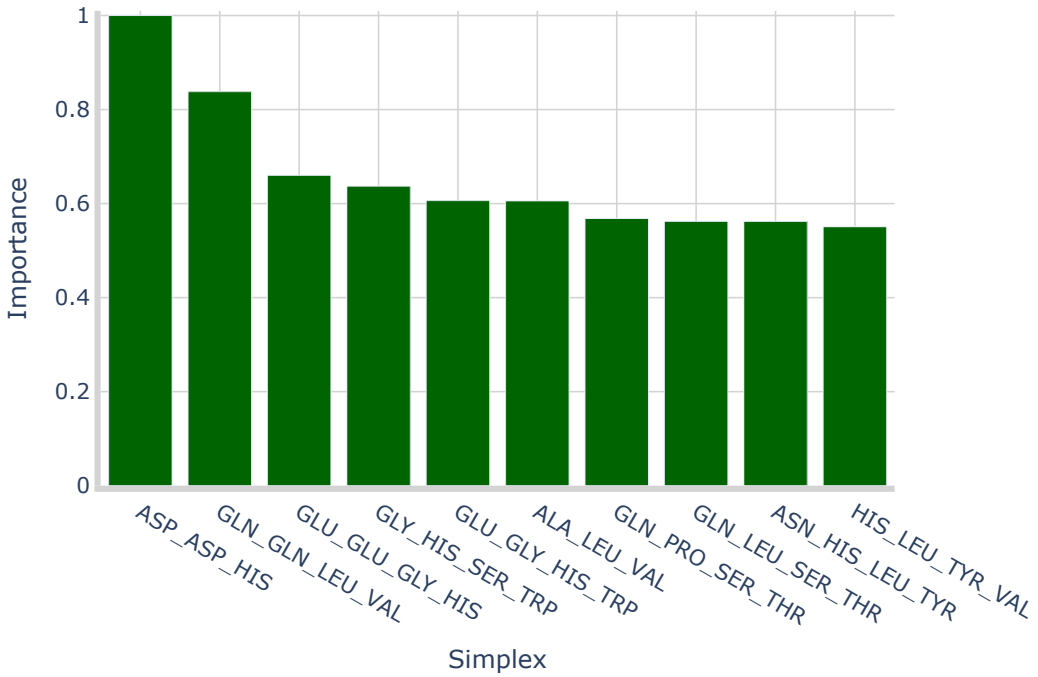


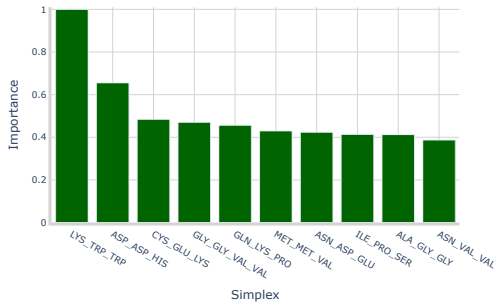
Figure 5.7: Top 10 Most Relevant Simplices According to RF MDI on Simplicial Complexes Embedding - Task B

OvR classifier is fit independently, raw coefficient magnitudes reflect its own regularization path and are best read as within-class rankings rather than absolute importances. Finally, given the construction of the histogram dictionary, families of closely related simplices may be correlated, given these conditions ℓ_1 solutions tend to select a single representative feature and can exhibit modest instability, for this reason minor variations in the top lists across classes are hard to correctly interpret.

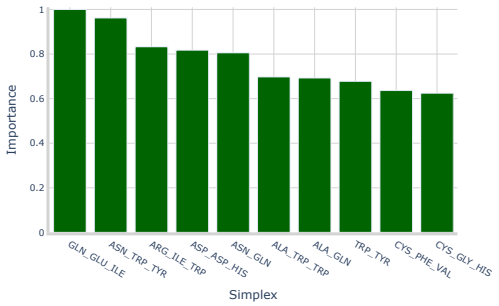
Despite such limitations it is worth noting that the 3-simplex *ASP-ASP-HIS* appears of great importance for both Oxidoreductases (EC Class 1) and Transferases (EC Class 2) as well as the notable differences in the distributions of the top 10 coefficients: for classes 2 to 5 the relevance distribution appears fairly evenly distributed (i.e., many of the best simplices have relatively close importance scores). Classes 1 and 6 on the other hand exhibit the opposite pattern, only the best 2 simplices have relatively similar importances while others display a great drop in relevance.

A final consideration on ℓ_1-Lin-SVM implicit feature selection is that there is no

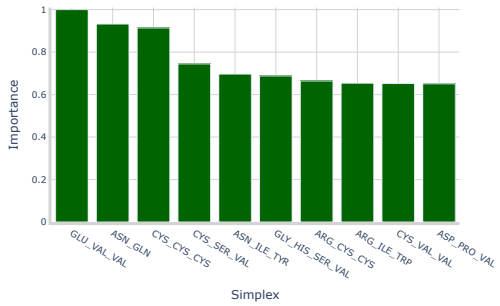
simplex that has been selected as relevant for all classes, this signals that different enzymatic classes are effectively differentiated by diverse structural motifs which further strengthens the grounds for PCN representation via simplicial complexes.



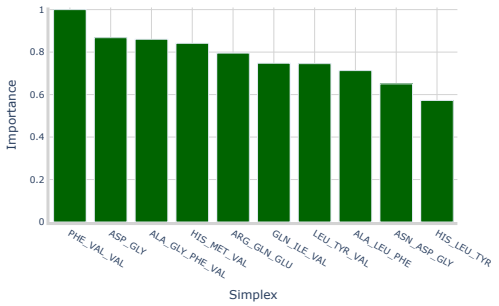
(a) EC Class 1



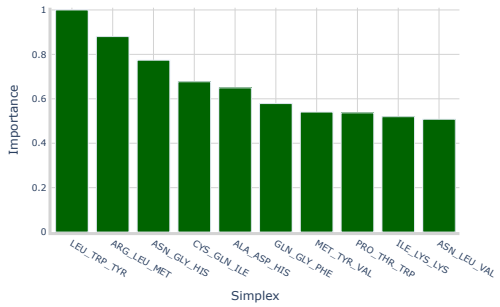
(b) EC Class 2



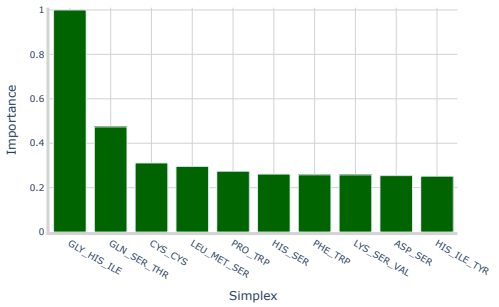
(c) EC Class 3



(d) EC Class 4



(e) EC Class 5



(f) EC Class 6

Figure 5.8: Most Relevant Simplices for each EC Class According to ℓ_1-Lin-SVM on Simplicial Complexes Embedding - Task B

5.2.3 INDVAL Embedding

Table 5.10 presents the performances of all models working with the INDVAL embedding in Task B. Also in this case the best performing model is ℓ_1-Lin-SVM which, despite having the lowest training ABA attains the best value in testing. This does not hold for standard accuracy results where it is the worst among the models also in testing. It is worth recalling that standard accuracy measures the proportion of correctly classified samples across the entire dataset, it is therefore strongly biased towards the most represented classes. ABA on the other hand computes per-class recalls first and then averages them, the same weight is assigned to each class regardless of its frequency. As a consequence, a higher ABA accompanied by a lower accuracy indicates that the classifier is able to capture informative patterns also in minority classes even if this comes at the cost of slightly reduced performance on majority ones. In practice, such behavior suggests that the model is better balanced across the class spectrum and performs better on under-represented EC classes.

The strong ℓ_1 penalty allows for a highly regularized model which on the INDVAL dataset excludes on average ~ 592 features from the classification, approximately 46% of the dataset. As expected the feature selection is less harsh compared to the previous section due to features being already pre-selected according to the INDVAL criterion. It is also important to note how, compared the results in Table 5.9 the INDVAL embedding achieves an average testing ABA only 1% lower than the full simplicial complexes embedding while being less than 10% the size. With this in mind the INDVAL scores appears to be exceptional in selecting the most relevant sub-structures for classification.

Figure 5.9 shows the feature importance plot for the RF model which again indicates as most important feature the 3-simplex *ASP-ASP-HIS* confirming even further the relevance of such substructure.

Figure 5.10 on the other hand presents the same visualization as in Figure 5.8 but for the INDVAL embedding. It is easy to see how the Feature Importances are more uniformly distributed compared to the ones coming from the full simplicial complexes embedding, this is probably due to the feature selection carried out via INDVAL scores that allows the model to consider only a very small class-relevant subset of features

Model	Metric	Train	Validation	Test
ν-SVM	ABA	0.995 ± 0.004	0.869 ± 0.009	0.861 ± 0.008
	ACC	0.999 ± 0.001	0.951 ± 0.002	0.946 ± 0.003
	F1	0.999 ± 0.001	0.950 ± 0.002	0.945 ± 0.003
ℓ_1-Lin-SVM	ABA	0.990 ± 0.004	0.899 ± 0.008	0.893 ± 0.012
	ACC	0.983 ± 0.006	0.948 ± 0.004	0.942 ± 0.006
	F1	0.983 ± 0.006	0.949 ± 0.004	0.943 ± 0.005
RF	ABA	0.991 ± 0.006	0.894 ± 0.010	0.888 ± 0.010
	ACC	0.989 ± 0.006	0.951 ± 0.006	0.948 ± 0.006
	F1	0.990 ± 0.006	0.952 ± 0.005	0.948 ± 0.006

Table 5.10: Performances on INDVAL Embedding for all Models – Task B

whose relevance is smoothed due to the decreased dimensionality of the dataset. The 3-simplex *ASP-ASP-HIS* appears also here for both EC classes 1 and 2.

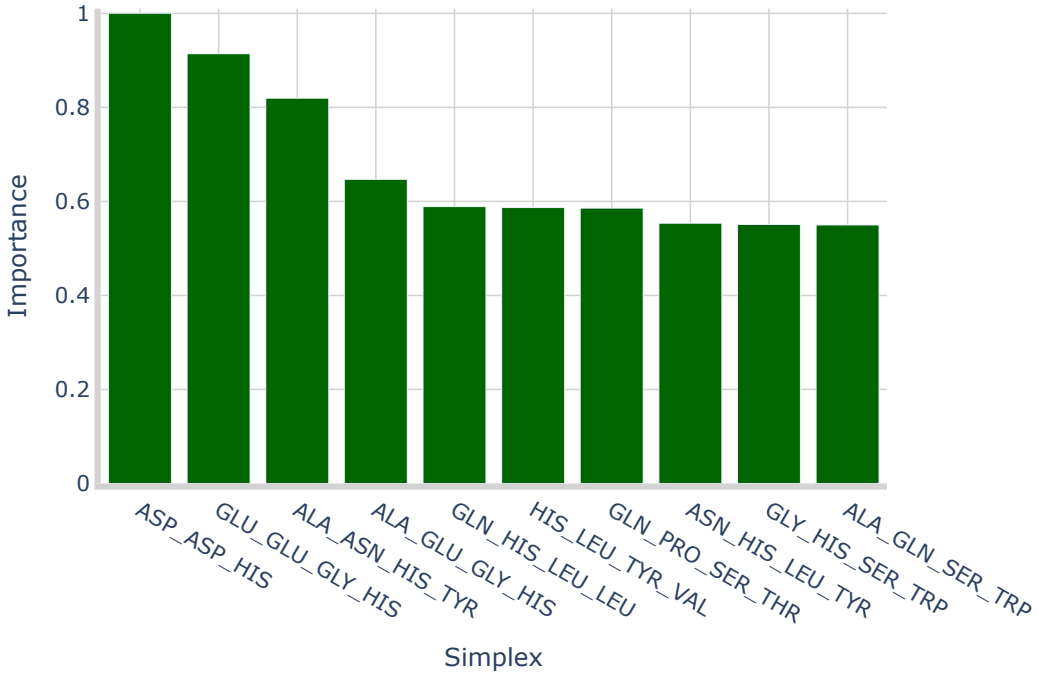
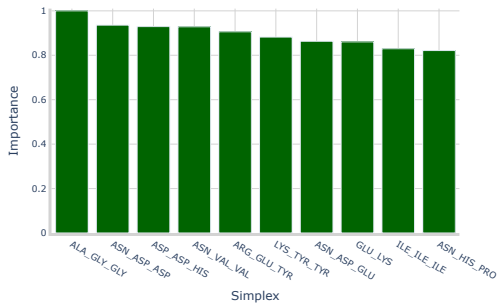
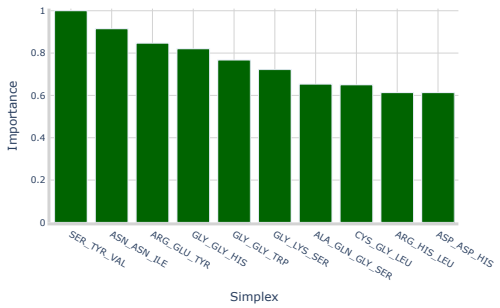


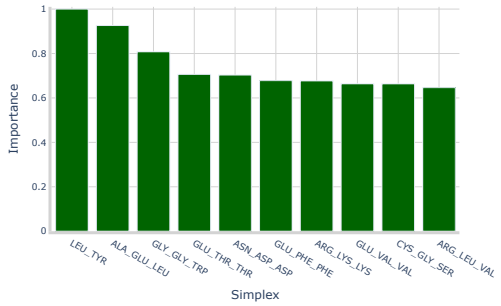
Figure 5.9: Top 10 Most Relevant Simplices According to RF MDI on INDVAL Embedding - Task B



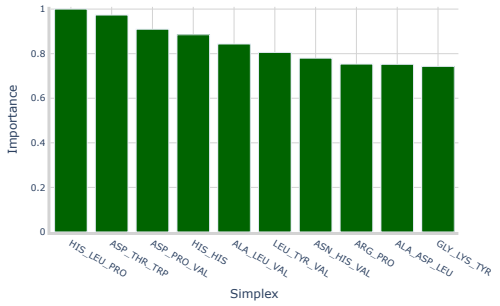
(a) EC Class 1



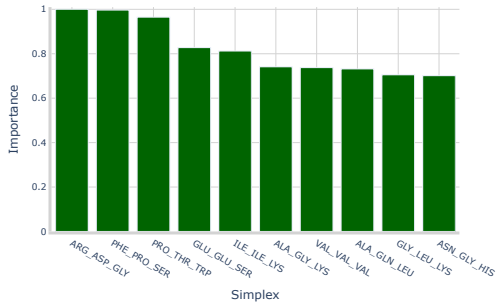
(b) EC Class 2



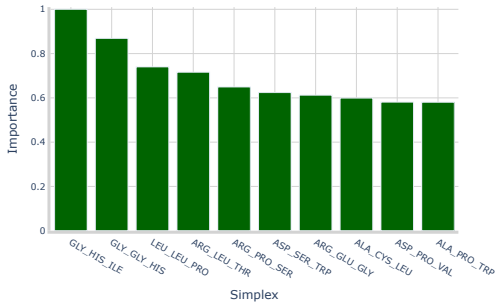
(c) EC Class 3



(d) EC Class 4



(e) EC Class 5



(f) EC Class 6

Figure 5.10: Most Relevant Simplices for each EC Class According to ℓ_1-Lin-SVM on INDVAL Embedding - Task B

5.2.4 (Hyper)Graph Kernels

Table 5.11 shows ν-SVM performances on both explored kernel methods in Task B. In the multiclass setting of Task B, the kernel-based approaches remain competitive but exhibit a reversed ranking with respect to Task A: the HCK attains $\mathrm{ABA} = 0.898$ on

the test set, surpassing the JK at $\mathrm{ABA} = 0.884$. This pattern suggests that, when discriminating among enzyme classes, cosine similarity over sparse symbolic histograms may be more tolerant to inter-class sharing of substructures, whereas Jaccard's overlap ratio can over-penalize such situations. As in Task A, these performances must be weighed against the computational footprint and reduced interpretability of kernel pipelines which remain practical considerations in large-scale proteome-level analyses.

Model	Kernel	Metric	Train	Validation	Test
ν-SVM	HCK	ABA	0.995 ± 0.008	0.905 ± 0.007	0.898 ± 0.011
		ACC	0.998 ± 0.002	0.967 ± 0.002	0.962 ± 0.004
		F1	0.998 ± 0.002	0.967 ± 0.002	0.962 ± 0.004
	JK	ABA	1.000 ± 0.000	0.888 ± 0.005	0.884 ± 0.007
		ACC	1.000 ± 0.000	0.967 ± 0.002	0.963 ± 0.003
		F1	1.000 ± 0.000	0.967 ± 0.002	0.963 ± 0.003

Table 5.11: Performances on Kernel Methods — Task B

5.2.5 Multiclass GNN

Table 5.12 presents GNN performances for Task B. The same training pattern presented in Section 5.1.5 was used resulting in a different topology for each of the data splits. The trends encountered for Task B are similar to the ones of Task A: 80% of the topologies exploit OHE representation of amino acid labels and 60% of them chose *GIN Convolution* as MP strategy. Coherently with Task A 80% of the structures used *max pooling* and all of them exploited batch normalization in MP layers. The biggest differences with respect to the results of Task A in terms of GNN topologies are the increased number of hidden dimensions (80% of the topologies leveraged the maximal available value of 256) and the ubiquitous presence of normalization layers after pooling. The sharp increase in hidden dimensions suggests that the multiclassification setting of Task B represents an inherently more complicated task, in this context more flexibility and expressive power (i.e. wider GNN structures) bring better performances without incurring in overfitting and degraded validation and test performances. The reaching of the upper limit of the search space for the hidden dimensions of the network could also indicate that, given more generous search spaces and computing power, it could be possible to

further refine the GNN approach to reach even better results.

In terms of performances GNN structures excel in Task B showing a test ABA of 0.921 surpassing the previous best model/embedding combination (ℓ_1-Lin-SVM on full simplicial complexes embedding) by a respectable 2.1%. Such performances establish GNNs as the best performing ML approach for first-level EC number prediction among the tested ones.

Model	Metric	Train	Validation	Test
GNN	ABA	0.981 ± 0.009	0.926 ± 0.011	0.921 ± 0.011
	ACC	0.968 ± 0.012	0.945 ± 0.013	0.942 ± 0.008
	F1	0.971 ± 0.011	0.948 ± 0.011	0.946 ± 0.006

Table 5.12: GNN Architecture Performances - Task B

5.2.6 Summary of Task B

In Task B, multiclass enzyme classification benefits markedly from representations grounded in higher-order structural motifs. As summarized in Table 5.13, spectral density embeddings are consistently the weakest option across models, mirroring the limitations observed in Task A and reflecting the detrimental impact of strong linear collinearity on sparse linear methods. In contrast, both the simplicial complexes and INDVAL embeddings achieve substantially higher ABA, with ℓ_1-Lin-SVM emerging as the most effective classifier for explicit histogram representations. The superiority of ℓ_1-Lin-SVM over ν-SVM and RF in this setting corroborates the advantage of embedded feature selection in very high-dimensional, sparse spaces: by shrinking most coefficients to zero, the model isolates a compact set of class-discriminative simplices and delivers optimal performances. Notably, the INDVAL embedding attains only a marginal ABA reduction relative to the full simplicial dictionary while using less than one tenth of the features, underscoring the strength of INDVAL scores as an inexpensive, interpretable pre-filter that preserves discriminatory signals.

Kernel approaches provide a competitive alternative without explicit feature engineering. However, the ranking between kernels reverses relative to Task A: the HCK surpasses the JK in ABA terms both in validation and testing. This inversion suggests

that, in the presence of inter-class sharing of local substructures cosine similarity offers a more tolerant notion of relatedness than the set-overlap-based Jaccard measure, which can over-penalize partial sharing.

Finally end-to-end graph neural networks trained on raw PCNs deliver the strongest overall results in Task B. The optimized GNNs reach a test ABA of 0.921. The prevalence of wider hidden dimensions, post-pooling normalization and MP operators with higher representational capacity (*GIN Convolution*) indicate that multiclass EC prediction benefits from increased model expressivity without incurring overfitting. These findings combined show that first-level EC assignment can be addressed effectively by both classical ML and DL approaches; among explicit representations, simplices-based histograms with ℓ_1-regularized linear classification offer an attractive accuracy-efficiency-interpretability trade-off, while end-to-end GNNs constitute the optimal choice in terms of pure predictive performance with minimal feature engineering.

Representation Strategy	ν-SVM	ℓ_1-Lin-SVM	RF	GNN
Spectral Density	0.733 ± 0.012	0.423 ± 0.009	0.686 ± 0.005	-
Simplicial Complexes	0.864 ± 0.009	0.902 ± 0.013	0.882 ± 0.009	-
INDVAL	0.861 ± 0.008	0.893 ± 0.012	0.888 ± 0.010	-
Histogram Kernel	0.898 ± 0.011	-	-	-
Jaccard Kernel	0.884 ± 0.007	-	-	-
Raw PCNs	-	-	-	**0.921 ± 0.011**

Table 5.13: Test Set ABA for all Models and Representation Strategies - Task B

Table 5.14 shows the per-class average results of the best-performing model (GNN) in Task B. The GNN achieves consistently high values across most evaluation metrics, particularly for the well-represented classes EC 2 and EC 3, where both recall and precision remain above 0.95 on average. These outcomes indicate that the classifier is highly effective at capturing the structural signatures of the dominant enzyme families. Conversely, the least represented classes (EC 5 and EC 6) exhibit lower stability, with EC 6 in particular showing reduced precision and F1-score, reflecting the greater difficulty of learning from small sample sizes. Nonetheless, the model demonstrates remarkable specificity across all classes, exceeding 0.97 in every case and reaching 0.998 for EC 4, highlighting its robustness in avoiding false positives. Overall, the model not only generalizes well on abundant classes but also maintains strong discriminative power across

the class spectrum despite substantial data imbalance.

EC Class	Recall	Precision	F1-score	Specificity	Support
EC 1	0.951 ± 0.019	0.952 ± 0.013	0.952 ± 0.014	0.992 ± 0.002	591
EC 2	0.928 ± 0.010	0.973 ± 0.006	0.950 ± 0.004	0.982 ± 0.004	1776
EC 3	0.954 ± 0.009	0.956 ± 0.008	0.955 ± 0.007	0.978 ± 0.004	1437
EC 4	0.975 ± 0.009	0.978 ± 0.008	0.977 ± 0.007	0.998 ± 0.001	311
EC 5	0.942 ± 0.013	0.892 ± 0.027	0.916 ± 0.013	0.996 ± 0.001	132
EC 6	0.854 ± 0.053	0.460 ± 0.059	0.594 ± 0.045	0.978 ± 0.007	89

Table 5.14: Best Performer per-class Performance Metrics in Testing - Task B

Chapter 6

Conclusions and Future Prospects

This thesis investigated whether protein structures represented as PCNs carry sufficient topological signal to support accurate prediction of physiological roles. Two complementary classification settings were addressed on a curated subset of the human proteome: a binary discrimination between enzymatic and non-enzymatic proteins (Task A) and a multiclass assignment of first-level EC classes for enzymatic proteins (Task B). The analysis involved $48,019$ proteins for Task A and $21,679$ enzymes for Task B, with all experiments conducted under a stratified five-fold protocol using fixed splits across methods to enable a systematic paired comparison between representation strategies and learning algorithms.

Both explicit graph embeddings and end-to-end graph learning were considered for the experiments. Within explicit representations, three families were explored: (i) spectral density descriptors of the Normalized Laplacian associated with PCNs; (ii) symbolic histograms derived from simplicial complexes evaluated on clique hypergraphs of PCNs; and (iii) their INDVAL-filtered variants for model-agnostic feature reduction. In parallel, two graph kernels over the complete simplicial complexes embedding (the HCK and the JK) were employed to capture non-linear similarities without additional feature engineering. Message-passing GNNs, on the other hand, were optimized to learn directly from residue–residue contact graphs in an end-to-end fashion.

Three standard classifiers (kernel ν-SVM, ℓ_1-Linear-SVM, and RF) were selected to probe complementary learning strategies: maximum-margin learning with a non-linear

kernel, embedded feature selection via lasso regularization, and non-parametric ensembles sensitive to feature interactions. End-to-end GNNs on the other hand were tuned over a plethora of architectural choices (MP operator, hidden dimensionality, pooling strategy, and related design factors) to identify effective PCN encoders under feasible computational constraints. Results were satisfactory for both tasks: in Task A, the top-performing pipeline was ν-SVM on JK with test ABA = 0.900, marginally ahead of the GNN counterpart (ABA = 0.898). This phenomenon indicates that end-to-end learning can almost match kernel baselines without additional hand-crafted features.

The results from Task B strengthen such considerations as end-to-end GNNs were the best overall with test ABA = 0.921, while among explicit representations the ℓ_1-Linear-SVM on the full simplicial histogram reached the best results with test ABA = 0.902. Per-class analysis of GNN results showed excellent precision and recall on well-represented classes (EC classes 1–4) with slight instability on the sparser classes (EC classes 5–6).

Feature importance analyses consistently highlighted recurring motifs across tasks and models, with the 3-simplex *ASP–ASP–HIS* emerging as a salient discriminant for every symbolic histogram based model. The INDVAL based feature selection proved to be able to delete many non-discriminative substructures with minimal loss in ABA confirming its role as an interpretable model-agnostic selector of signature motifs.

The evaluation benefited from fixed stratified folds and a methodological breadth which together provided a balanced perspective on accuracy, efficiency, and interpretability for both tasks. Nonetheless, several limitations should be acknowledged: (i) the dataset excluded multifunctional and moonlighting proteins and restricted attention to a single first-level EC label, enforcing mutually exclusive roles; and (ii) to balance computational costs and scale, only standard GNNs with residue-identity features were considered, leaving geometric encoders and sequence-informed models unexplored.

In follow-up studies it will be possible to extend end-to-end architectures to 3D-aware, E(3)-equivariant GNNs and geometric vector perceptrons to better exploit fine-grained geometric information as well as moving beyond mutually exclusive labels to multi-label classification addressing directly multifunctional and moonlighting proteins

to align the modeling objectives with biological reality. Furthermore, deep topological learning strategies—such as message passing on simplicial or cell complexes with specialized operators—could be incorporated to encode higher-order interactions and global shape priors that remain inaccessible to purely pairwise GNN models.

Overall, the findings indicate that structural information encoded in PCNs is highly predictive of protein physiological roles at scale. Topology-driven embeddings provide accurate and interpretable baselines, while carefully tuned GNNs deliver the strongest multiclass performance, opening a path toward higher order, geometry-aware, and explicitly multi-label structural analysis.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[2] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. Protein function. In *Molecular Biology of the Cell. 4th edition*. Garland Science, 2002.

[3] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[4] Gábor Bacsó, Sylvain Gravier, András Gyárfás, Myriam Preissmann, and András Sebo. Coloring the maximal cliques of graphs. *SIAM Journal on Discrete Mathematics*, 17(3):361–376, 2004.

[5] Andrzej Bargiela and Witold Pedrycz. The roots of granular computing. In *2006 IEEE International Conference on Granular Computing*, pages 806–809, 2006.

[6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[7] Helen M Berman, Buvaneswari Coimbatore Narayanan, Luigi Di Costanzo, Shuchismita Dutta, Sutapa Ghosh, Brian P Hudson, Catherine L Lawson, Ezra

Peisach, Andreas Prlić, Peter W Rose, et al. Trendspotting in the protein data bank. *FEBS letters*, 587(8):1036–1045, 2013.

[8] B Bhargavi, K Swarupa Rani, and Arunjyoti Neog. Finding multidimensional constraint reachable paths for attributed graphs. *EAI Endorsed Trans. Scalable Inf. Syst.*, 10(1):e8, 2022. doi: https://doi.org/10.4108/eetsis.v9i4.2581.

[9] Frimpong Boadu, Hongyuan Cao, and Jianlin Cheng. Combining protein sequences and structures with transformers and equivariant graph neural networks to predict protein function. *Bioinformatics*, 39(Supplement_1):i318–i325, 2023.

[10] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[11] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[12] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[13] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[14] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[15] Horst Bunke. Graph-based tools for data mining and machine learning. In *International workshop on machine learning and data mining in pattern recognition*, pages 7–19. Springer, 2003.

[16] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998. doi: https://doi.org/10.1023/A:1009715923555.

[17] Steve Butler. Algebraic aspects of the normalized laplacian. In *Recent Trends in Combinatorics*, pages 295–315. Springer, 2016.

[18] Seth Chaiken and Daniel J Kleitman. Matrix tree theorems. *Journal of combinatorial theory, Series A*, 24(3):377–381, 1978.

[19] Chih-Chung Chang and Chih-Jen Lin. Training v-support vector classifiers: theory and algorithms. *Neural computation*, 13(9):2119–2147, 2001.

[20] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[21] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422, 2009.

[22] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[23] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1367–1372, 2004.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, EC-14(3):326–334, 1965.

[26] Enrico De Santis, Alessio Martino, Antonello Rizzi, and Fabio Massimo Frattale Mascioli. Dissimilarity space representations and automatic feature selection for

protein function prediction. In *2018 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2018.

[27] Luisa Di Paola, Micol De Ruvo, Paola Paci, Daniele Santoni, and Alessandro Giuliani. Protein contact networks: an emerging paradigm in chemistry. *Chemical reviews*, 113(3):1598–1613, 2013.

[28] Reinhard Diestel. *Graph Theory*. Springer, 5 edition, 2017.

[29] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[30] Paul D Dobson and Andrew J Doig. Predicting enzyme class from protein structure without alignments. *Journal of molecular biology*, 345(1):187–199, 2005.

[31] Marc Dufrêne and Pierre Legendre. Species assemblages and indicator species: the need for a flexible asymmetrical approach. *Ecological monographs*, 67(3):345–366, 1997.

[32] Ernesto Estrada and Juan A. Rodríguez-Velázquez. Subgraph centrality and clustering in complex hyper-networks. *Physica A: Statistical Mechanics and its Applications*, 364:581–594, 2006.

[33] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874, 2008.

[34] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[35] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[36] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[37] Corrado Gini. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.[Fasc. I.]*. Tipogr. di P. Cuppini, 1912.

[38] Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.

[39] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[40] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[41] Qian-Ping Gu and Jiajian Leo Liang. Algorithms and computational study on a transportation system integrating public transit and ridesharing of personal vehicles. *Computers & Operations Research*, 164:106529, 2024.

[42] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[43] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[44] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415, 2008.

[45] International Union of Biochemistry. Commission on Enzymes. *Report of the Commission on Enzymes*. I.U.B. Symposium Series, Vol. 20. Pergamon Press, Oxford, 1961.

[46] Kei (hvy) Ishikawa. "multivariate" tpe makes optuna even more powerful. *Medium (Optuna blog)*, October6 2020.

[47] Sarika Jalan and Jayendra N Bandyopadhyay. Random matrix analysis of network laplacians. *Physica A: Statistical Mechanics and its Applications*, 387(2-3):667–674, 2008.

[48] Constance J Jeffery. Moonlighting proteins. *Trends in biochemical sciences*, 24(1):8–11, 1999.

[49] Constance J Jeffery. Multifunctional proteins: examples of gene sharing. *Annals of medicine*, 35(1):28–35, 2003.

[50] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

[51] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[52] Jonas L Juul, Austin R Benson, and Jon Kleinberg. Hypergraph patterns and collaboration structure. *Frontiers in Physics*, 11:1301994, 2024.

[53] Henrik Kaessmann, Sebastian Zöllner, Anton Nekrutenko, and Wen-Hsiung Li. Signatures of domain shuffling in the human genome. *Genome research*, 12(11):1642–1650, 2002.

[54] TN Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[55] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pages 315–322, 2002.

[56] Theodore Lewis and William L. Stone. *Biochemistry, Proteins Enzymes*. StatPearls Publishing, Treasure Island, FL, Jan 2025.

[57] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019.

[58] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[59] Lorenzo Livi, Alessandro Giuliani, and Alireza Sadeghian. Characterization of graphs for protein structure modeling and recognition of solubility. *Current Bioinformatics*, 11(1):106–114, 2016.

[60] Lorenzo Livi, Enrico Maiorino, Alessandro Giuliani, Antonello Rizzi, and Alireza Sadeghian. A generative model for protein contact networks. *Journal of Biomolecular Structure and Dynamics*, 34(7):1441–1454, 2016.

[61] Enrico Maiorino, Antonello Rizzi, Alireza Sadeghian, and Alessandro Giuliani. Spectral reconstruction of protein contact networks. *Physica A: Statistical Mechanics and its Applications*, 471:804–817, 2017.

[62] Alessio Martino, Fabio Massimo Frattale Mascioli, and Antonello Rizzi. On the optimization of embedding spaces via information granulation for pattern recognition. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[63] Alessio Martino, Alessandro Giuliani, and Antonello Rizzi. (hyper)graph embedding and classification via simplicial complexes. *Algorithms*, 12(11), 2019.

[64] Alessio Martino, Enrico Maiorino, Alessandro Giuliani, Mauro Giampieri, and Antonello Rizzi. Supervised approaches for function prediction of proteins contact networks from topological structure information. In *Scandinavian Conference on Image Analysis*, pages 285–296. Springer, 2017.

[65] Alessio Martino and Antonello Rizzi. (hyper) graph kernels over simplicial complexes. *Entropy*, 22(10):1155, 2020.

[66] Andrew G McDonald and Keith F Tipton. Enzyme nomenclature and classification: the state of the art. *The FEBS journal*, 290(9):2214–2231, 2023. doi: https://doi.org/10.1111/febs.16274.

[67] Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of symbolic computation*, 60:94–112, 2014.

[68] James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

[69] Aziz Mithani, Gail M Preston, and Jotun Hein. Rahnuma: hypergraph-based tool for metabolic pathway prediction and network comparison. *Bioinformatics*, 25(14):1831–1832, 2009.

[70] Christopher Morris, Fabrizio Frasca, Nadav Dym, Haggai Maron, İsmail İlkan Ceylan, Ron Levie, Derek Lim, Michael Bronstein, Martin Grohe, and Stefanie Jegelka. Future directions in the theory of graph machine learning. *arXiv preprint arXiv:2402.02287*, 2024.

[71] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[72] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[73] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021.

[74] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford infolab, 1999.

[75] Youngser Park, C Priebe, D Marchette, and Abdou Youssef. Anomaly detection using scan statistics on time series hypergraphs. In *Link analysis, counterterrorism and security (LACTS) conference*, page 9. SIAM Pennsylvania, 2009.

[76] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[77] Alice Patania, Giovanni Petri, and Francesco Vaccarino. The shape of collaborations. *EPJ Data Science*, 6(1):18, 2017.

[78] Linus Pauling, Robert B Corey, and Herman R Branson. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 37(4):205–211, 1951.

[79] William R Pearson and David J Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448, 1988.

[80] Witold Pedrycz. Granular computing: an introduction. In *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, volume 3, pages 1349–1354. IEEE, 2001.

[81] Sebastian Raschka. Biopandas: Working with molecular structures in pandas dataframes. *The Journal of Open Source Software*, 2(14), jun 2017.

[82] RCSB Protein Data Bank. Primary sequences, 2025. Accessed on 12 August 2025.

[83] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[84] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[85] Michael T Schaub, Austin R Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie. Random walks on simplicial complexes and the normalized hodge 1-laplacian. *SIAM Review*, 62(2):353–391, 2020.

[86] Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

[87] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[88] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[89] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[90] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[91] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[92] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[93] Mikko Taipale. Disruption of protein function by pathogenic mutations: common and uncommon mechanisms. *Biochemistry and Cell Biology*, 97(1):46–57, 2019.

[94] Robert L Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

[95] Keith Tipton. Translocases (EC 7): A new EC Class. Technical report, Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (IUBMB), August 2018.

[96] Keith Tipton and Andrew McDonald. A Brief Guide to Enzyme Nomenclature and Classification. Technical report, International Union of Biochemistry and Molecular Biology (NC-IUBMB), 2018. Revision November 2018.

[97] Kentaro Tomii, Yoshito Sawada, and Shinya Honda. Convergent evolution in structural elements of proteins investigated using cross profile analysis. *BMC bioinformatics*, 13(1):11, 2012.

[98] Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[99] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[100] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[101] Junqi Wang, Hailong Li, Gang Qu, Kim M Cecil, Jonathan R Dillman, Nehal A Parikh, and Lili He. Dynamic weighted hypergraph convolutional network for brain functional connectome analysis. *Medical image analysis*, 87:102828, 2023.

[102] Edwin C. Webb. *Enzyme nomenclature 1992. Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the Nomenclature and Classification of Enzymes.* Academic Press, 6 edition, 1992.

[103] David Whitford. *Proteins: structure and function*. John Wiley & Sons, 2013.

[104] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[105] Kai Siong Yow, Ningyi Liao, Siqiang Luo, Reynold Cheng, Chenhao Ma, and Xiaolin Han. A survey on machine learning solutions for graph pattern extraction. *arXiv preprint arXiv:2204.01057*, 2022.

[106] Ji Zhu, Saharon Rosset, Robert Tibshirani, and Trevor Hastie. 1-norm support vector machines. *Advances in neural information processing systems*, 16, 2003.

[107] Afra Zomorodian. Fast construction of the vietoris-rips complex. *Computers & Graphics*, 34(3):263 – 271, 2010. Shape Modelling International (SMI) Conference 2010.

[108] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.