UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
Y CIENCIAS DE LA COMPUTACIÓN

IDENTIFICACIÓN DE BLANCOS MOLECULARES PARA EL TRATAMIENTO DE ALZHEIMER MEDIANTE INTERACCIONES PROTEÍNA-PROTEÍNA (PPI) Y ALGORITMOS DE DEEP LEARNING

POR

Macarena Alejandra Madrid Becerra

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniera Civil Informática

Profesora Guía
Mabel Vidal Miranda

Profesor Co-guía
David Ramírez Sánchez

Octubre 2025
Concepción (Chile)

Agradecimientos

En primer lugar, quiero agradecer a Dios y a mi familia por darme sabiduría, fortaleza y amor necesarios durante todo mi proceso universitario.

A mi papá Nelson, por ser siempre mi guía y enseñarme a tomar el camino correcto, transmitiéndome valores y convicciones que hoy me sostienen. A mi mamá Marcela, por su cariño incondicional, por los desayunos que me esperaban cada mañana antes de ir a la universidad y por hacer del hogar un lugar cálido y acogedor. A mis hermanas Camila y Natalia, quienes, a pesar de la distancia, siempre me hicieron llegar su cariño y ánimos. A mi perrita Diana, que me acompañó fielmente, incluso en las jornadas de clases online cuando comencé la universidad en pandemia. Este logro también les pertenece a cada uno de ustedes; sin su compañía, amor y apoyo nada de esto habría sido posible.

Quiero agradecer a mi pololo Felipe por acompañarme en cada etapa de este proceso, por celebrar conmigo cada pequeño logro y darme ánimo en los momentos difíciles. Estoy feliz de compartir este triunfo contigo y de proyectar nuevos desafíos y alegrías juntos.

A mis amigos y amigas de la universidad, con quienes compartí estos seis años de esfuerzo y crecimiento. Gracias por convertir este camino en una etapa inolvidable, llena de risas, apoyo y aprendizajes compartidos.

A mis profesores y profesoras de la Universidad de Concepción, quienes no solo me entregaron conocimientos valiosos para mi formación profesional, sino que también me enseñaron a crecer como persona. En especial, agradezco profundamente a mi profesora guía, Dra. Mabel Vidal, por su constante apoyo, orientación y excelente disposición durante todo el desarrollo de esta memoria. Su ejemplo como académica y como mujer en áreas STEM ha sido una gran inspiración para mí.

Asimismo, expreso mi gratitud a mi co-guía, Dr. David Ramírez, por su disposición y paciencia al aclarar mis dudas, por abrirme las puertas de su laboratorio y permitirme trabajar con los datos que hicieron posible esta investigación.

Finalmente, agradezco a todas las personas que, de manera directa o indirecta, fueron parte de este camino y contribuyeron a que hoy pueda concretar este importante logro en mi vida.

Resumen

La enfermedad de Alzheimer (EA) carece de terapias capaces de detener o revertir su progresión, por lo que priorizar *blancos moleculares* a partir de redes de interacción proteína–proteína (PPI) resulta clave. Este trabajo tuvo por objetivo detectar módulos funcionales en una red PPI asociada a EA e integrar dicha estructura con anotaciones de Gene Ontology (GO) para proponer candidatos a blanco. Para ello, se desarrolló un *pipeline* basado en redes neuronales del tipo Graph Neural Network (GNN) aplicado a una red de 5,390 proteínas y 430,206 interacciones, anotada con 6,928 términos GO.

La metodología consistió en la construcción de representaciones de nodos y aristas que permitieron entrenar un modelo basado en redes neuronales gráficas. Posteriormente, se generaron *embeddings* que fueron analizados mediante estrategias de reducción de dimensionalidad y agrupamiento, lo que posibilitó la identificación de módulos funcionales y su posterior caracterización biológica.

En cuanto a los resultados, el modelo seleccionado alcanzó un AUC de 0.9596 en el conjunto de prueba y permitió la detección de 14 módulos funcionales (denominados *clusters*) consistentes. Adicionalmente, se calcularon métricas de conectividad intra- e interclúster junto con el *Connectivity Score*. A partir de estas métricas se identificaron las cinco proteínas con los valores más altos en cada *cluster*, conformando así el conjunto final de 145 proteínas priorizadas para su estudio posterior.

En conclusión, el enfoque propuesto constituye una estrategia computacional para la integración de información topológica y funcional en redes PPI asociadas a la EA. Los hallazgos no sólo generan una lista priorizada de posibles blancos moleculares, sino que también establecen una base metodológica extensible a otras patologías, reforzando el potencial del aprendizaje profundo aplicado a problemas biológicos.

Palabras clave: enfermedad de Alzheimer; PPI; GNN; predicción de enlaces; *clustering*; enriquecimiento GO; priorización de blancos.

Abstract

Alzheimer's disease (AD) lacks therapies capable of halting or reversing its progression, making the prioritization of *molecular targets* from protein–protein interaction (PPI) networks a key step. The objective of this work was to detect functional modules in a PPI network associated with AD and to integrate this structure with Gene Ontology (GO) annotations in order to propose candidate targets. To this end, a pipeline was developed based on Graph Neural Networks (GNN), applied to a network of 5,390 proteins and 430,206 interactions, annotated with 6,928 GO terms.

The methodology consisted of building node and edge representations to train a graph neural network model. Subsequently, *embeddings* were generated and analyzed through dimensionality reduction and clustering strategies, enabling the detection of functional modules and their subsequent biological characterization.

Regarding the results, the selected model achieved an AUC of 0.9596 on the test set and allowed the detection of 14 consistent functional modules (denominated clusters). Additionally, intra- and inter-cluster connectivity metrics were computed together with the *Connectivity Score*. Based on these metrics, the five proteins with the highest values in each case were identified, thus constituting the final set of 145 prioritized proteins for further study.

In conclusion, the proposed approach represents a robust computational strategy for integrating topological and functional information in PPI networks associated with AD. The findings not only provide a prioritized list of potential molecular targets, but also establish a methodological foundation that can be extended to other pathologies, reinforcing the potential of deep learning applied to biological problems.

Keywords: Alzheimer's disease; PPI; GNN; link prediction; *clustering*; GO enrichment; molecular target prioritization.

Estructura del Documento

- **Capítulo 1. Introducción:** Presenta los antecedentes del problema, la relevancia de abordar la enfermedad de Alzheimer desde la perspectiva de redes PPI, la formulación del problema y los objetivos generales y específicos del trabajo.
- **Capítulo 2. Marco Teórico y Estado del Arte:** Revisa la literatura relacionada con detección de módulos en redes PPI, el concepto de blanco molecular, anotaciones funcionales mediante Gene Ontology, fundamentos de Deep Learning y GNN, así como técnicas de clustering y métricas de evaluación.
- **Capítulo 3. Datos:** Describe el conjunto de datos utilizado, su origen, las variables clave (proteínas, interacciones y anotaciones), y un análisis exploratorio de sus principales características topológicas y funcionales.
- **Capítulo 4. Metodología:** Expone el *pipeline* computacional propuesto, incluyendo el entorno de ejecución, la preparación de datos, la arquitectura y entrenamiento de la GNN, la detección de módulos mediante clustering y el análisis de enriquecimiento GO.
- **Capítulo 5. Resultados y Discusión:** Presenta los resultados obtenidos en la generación de *embeddings*, la identificación de módulos funcionales y su caracterización biológica. Además, discute las implicancias de los hallazgos en relación con la identificación de posibles blancos moleculares.
- **Capítulo 6. Conclusiones y Trabajo futuro:** Resume los principales aportes y resultados del estudio en relación con los objetivos planteados, y propone líneas de investigación futura que permitan profundizar y extender la metodología desarrollada.

Índice general

Índice de Tablas

Índice de Figuras

Capítulo 1. Introducción

El presente capítulo presenta los antecedentes del problema de investigación, centrado en la enfermedad de Alzheimer (EA) y la identificación de blancos moleculares. Se expone la relevancia de integrar información topológica y funcional mediante redes Protein-Protein Interaction (PPI) y datos de Gene Ontology (GO), y se detallan la formulación del problema y los objetivos del trabajo.

1.1. Antecedentes del problema

La demencia, de naturaleza crónica y progresiva, abarca diversas enfermedades que provocan deterioro cognitivo y limitan la capacidad para realizar actividades diarias, siendo el Alzheimer la forma más común, representando entre el 60 % y 70 % de los casos [1]. La EA es el proceso biológico que comienza con la aparición en el cerebro de una acumulación de proteínas en forma de placas amiloides y ovillos neurofibrilares, esto hace que las neuronas cerebrales mueran con el tiempo [2].

1.2. Relevancia del problema

De acuerdo con la Organización Mundial de la Salud (OMS), la demencia y, por ende, la EA, es una de las principales causas de discapacidad y dependencia en adultos mayores, afectando a más de 55 millones de personas a nivel global y generando un costo anual estimado de un billón de dólares en 2018 [3], es decir, generando un impacto social y económico considerable [1, 4]. A pesar de los numerosos avances científicos, la EA persiste sin cura definitiva, y los tratamientos disponibles solo pueden mejorar los síntomas o desacelerar el deterioro del pensamiento [2, 5, 6]. Contrario a la creencia popular, la OMS enfatiza que la demencia no es una consecuencia natural del envejecimiento y puede presentarse en etapas previas [1]. En la región de América Latina y el Caribe, su prevalencia aumenta rápidamente, con un impacto desproporcionado en las mujeres, cuyos años de vida ajustados por discapacidad (AVAD) superan en un 60 % la media global [7]. La pandemia de COVID-19 ha agravado esta situación, incrementando la carga sobre pacientes, familias y sistemas de salud, mientras persisten el desconocimiento, el estigma y las barreras para un diagnóstico y tratamiento oportunos [8].

1.3. Formulación del problema

Para estudiar estos blancos moleculares, las redes PPI se encuentran entre las redes más relevantes y ampliamente estudiadas [9]. En este tipo de redes un sistema biológico se representa

mediante nodos, que corresponden a proteínas, y aristas, que indican las relaciones entre ellas. Un módulo funcional, también denominado clúster, de una PPI corresponde a grupos de proteínas que participan en funciones específicas dentro del sistema, estos pueden actuar como una unidad aislada, siendo responsable de un proceso determinado, o bien integrarse en patrones de interacción más amplios, donde varios procesos se interconectan. Los módulos pueden definirse según su topología, como grupos densamente conectados, o su función, como conjuntos de proteínas que comparten una misma actividad biológica. Aunque un mismo grupo de nodos puede presentar ambas características, esto no implica que sean equivalentes, ya que corresponden a entidades distintas que no necesariamente coinciden. Esta diferencia justifica la importancia de considerar tanto la topología de la red como la información biológica asociada [10].

El problema radica en que los métodos actuales para identificar módulos funcionales en redes PPI se apoyan principalmente en la estructura de la red, favoreciendo subgrafos densamente conectados, y esta estrategia suele ignorar módulos funcionales pequeños o con baja conectividad que pueden contener proteínas clave.

Además, los blancos moleculares, entendidos como proteínas cuya modulación puede desencadenar efectos terapéuticos, identificados hasta ahora no han resultado en terapias eficaces contra la EA. Por este motivo, se propone estudiar la enfermedad desde una perspectiva de farmacología de sistemas, analizando cómo se relacionan entre sí los posibles blancos moleculares y explorando nuevos módulos dentro de la red PPI que permitan complementar el análisis topológico con atributos funcionales.

1.3.1. Objetivo General

El objetivo general de la Memoria de Título se define como:
Identificar potenciales blancos moleculares claves para el tratamiento terapéutico de la EA, integrando información topológica de redes PPI y funcional mediante datos de Gene Ontology (GO), utilizando algoritmos de deep learning.

1.3.2. Objetivos Específicos

Los objetivos específicos son:

OE1: Integrar información topológica y funcional proveniente de redes protein-protein interaction para el estudio de la Enfermedad del Alzheimer.

OE2: Implementar un modelo basado en algoritmos de deep learning que asocie proteínas con funciones biológicas relevantes en el contexto de la Enfermedad del Alzheimer.

OE3: Validar el enfoque propuesto mediante análisis cuantitativo los resultados obtenidos.

Capítulo 2. Marco Teórico y Estado del Arte

2.1. Relevancia de identificar nuevos blancos moleculares en EA

Un blanco molecular, también conocido como target, se define como una macromolécula, usualmente una proteína, cuya interacción con un fármaco desencadena un efecto terapéutico en el organismo. La correcta identificación de estos blancos constituye un paso esencial en el desarrollo de fármacos, pues determina en gran medida su eficacia y seguridad [11].

En el contexto de enfermedades neurodegenerativas, y particularmente en la EA, los esfuerzos de búsqueda de nuevos blancos moleculares se han orientado hacia mecanismos patológicos característicos, tales como la agregación de placas beta-amiloides, la disfunción mitocondrial y la neuroinflamación [12]. La identificación de estos blancos moleculares constituye el punto de partida para el desarrollo racional de fármacos, ya que permite comprender la enfermedad a nivel molecular y plantear un mecanismo de acción terapéutico en el que la actividad del blanco molecular asociada a la enfermedad es modulada por el fármaco. Estos procesos reflejan la complejidad de la enfermedad y justifican la necesidad de identificar nuevos blancos moleculares que permitan el diseño de terapias modificadoras de curso.

2.2. Redes de interacción proteína-proteína (PPI)

2.2.1. Definición de red PPI como grafo

Las PPI son un tipo fundamental de red biológica que representa las interacciones físicas directas y/o funcionales entre proteínas dentro de una célula u organismo [13]. Estas redes son herramientas esenciales en biología de sistemas y bioinformática para comprender la organización funcional y la dinámica de los sistemas biológicos.

Formalmente, una red PPI se modela como un grafo no dirigido $G = (V, E)$, donde los nodos (vértices) V representan proteínas y las aristas (enlaces) E simbolizan una interacción entre un par de proteínas. Una arista $(u, v) \in E$ indica que la proteína u interactúa con la proteína v.

2.2.2. Conceptos básicos de teoría de grafos

El análisis de redes PPI se apoya en conceptos fundamentales de la teoría de grafos, disciplina matemática que estudia estructuras formadas por nodos (o vértices) y aristas (o enlaces) [14].

Un grafo no dirigido, $G = (V, E)$, posee aristas $e \in E$ que representan una conexión bidireccional entre dos vértices de V. La *cardinalidad* de los conjuntos de vértices y aristas se expresa como $|V|$ y $|E|$, respectivamente, siendo $|E|$ conocido como el tamaño del grafo.

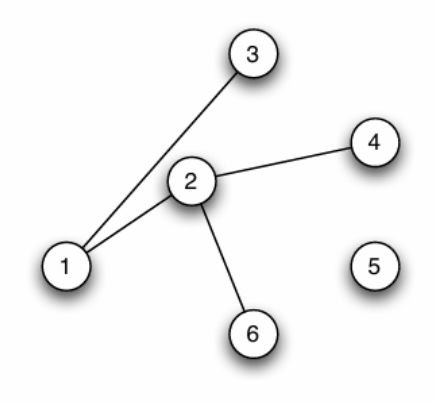


Figura 2.1: Ejemplo de un grafo no dirigido con su conjunto de vértices y aristas. El grafo se define como $G = (V, E)$, donde $V = \{1, 2, 3, 4, 5, 6\}$ corresponde al conjunto de vértices (nodos) y $E = \{\{1, 2\}, \{1, 3\}, \{2, 4\}, \{2, 6\}\}$ al conjunto de aristas (relaciones no dirigidas entre los vértices). Obsérvese que el nodo 5 no posee conexiones en este ejemplo.

Entre los conceptos más relevantes se encuentran:

- **Grado de un nodo**: número de aristas incidentes en un nodo, denotado como $\delta(v)$. Permite identificar nodos altamente conectados. Por ejemplo, en el grafo de la Figura 2.1, el vértice 2 tiene grado 3, es decir, $\delta(2) = 3$, ya que las aristas $(1, 2)$, $(2, 4)$ y $(2, 6)$ inciden en él.
- **Hubs**: nodos con un grado significativamente mayor al promedio, que suelen desempeñar funciones críticas en la estabilidad y la conectividad de la red.
- **Subgrafos**: subconjuntos de nodos densamente conectados entre sí, que suelen corresponder a unidades funcionales como complejos proteicos o rutas metabólicas.
- **Caminos y ciclos**: secuencias de nodos y aristas que conectan proteínas. Los ciclos reflejan redundancia funcional o robustez estructural en la red.

Estos conceptos permiten caracterizar la estructura y organización de las redes PPI más allá de la simple enumeración de interacciones.

2.2.3. Atributos de las proteínas y anotaciones funcionales

Proteínas y complejos proteicos

Las proteínas constituyen las unidades funcionales fundamentales en los sistemas biológicos. Cada proteína posee atributos estructurales y funcionales que permiten caracterizar su papel en la célula.

Una primera distinción relevante es entre proteínas individuales y complejos proteicos. Las proteínas individuales realizan funciones autónomas, mientras que los complejos proteicos representan asociaciones estables y específicas de dos o más cadenas polipeptídicas que actúan de manera integrada para llevar a cabo una función biológica [15, 16]. La función de un complejo

no reside en cada subunidad por separado, sino en la interacción sinérgica de todas ellas. En el caso de un complejo proteico, es esencial identificar la lista de proteínas individuales que lo componen, ya que su función emerge de la interacción y cooperación de sus subunidades.

Genes diferencialmente expresados (DEG)

Un atributo esencial para la caracterización funcional de proteínas es la expresión diferencial. Un gen es clasificado como **DEG** cuando presenta diferencias significativas en sus niveles de expresión entre dos o más condiciones biológicas (por ejemplo, Alzheimer vs. controles sanos). Esta clasificación se obtiene aplicando pruebas estadísticas que modelan la distribución de los conteos de ARN y estiman la significancia de los cambios observados. Herramientas ampliamente utilizadas, como *DESeq2*, implementan modelos basados en la distribución binomial negativa y métodos de corrección por comparaciones múltiples, permitiendo identificar genes diferencialmente expresados de manera robusta [17].

Estos genes pueden clasificarse en dos categorías principales:

- **Up-regulated**: genes cuya expresión aumenta en la condición estudiada en comparación con la condición control, lo que sugiere una activación o sobreexpresión asociada al proceso biológico o patológico en cuestión.

- **Down-regulated**: genes cuya expresión disminuye de manera significativa respecto al control, lo que puede reflejar pérdida de función, represión de rutas metabólicas o desregulación de procesos celulares clave.

La identificación de DEG permite priorizar proteínas relevantes en procesos patológicos, dado que cambios en su expresión suelen estar asociados a mecanismos de disfunción celular o de respuesta adaptativa.

Ontología Génica (GO)

Las proteínas no solo se definen por su estructura o por la expresión de sus genes, sino también por las funciones biológicas que desempeñan. La *Ontología Génica (Gene Ontology - GO)* constituye a una de las herramientas más robustas para clasificar y estandarizar el conocimiento sobre proteínas a partir de evidencia experimental [18]. GO categoriza las funciones de las proteínas en tres dominios principales:

- **Biological Process (BP)**: Describe los procesos biológicos en los que está involucrada una proteína, como el metabolismo, la respuesta al estrés o la división celular.

- **Molecular Function (MF)**: Se refiere a las actividades bioquímicas específicas que una proteína puede llevar a cabo.

- **Cellular Component (CC)**: Indica la localización subcelular donde la proteína actúa, por ejemplo, citoplasma, membrana, núcleo o mitocondria.

2.3. Deep Learning para análisis de PPIs

El aprendizaje profundo (*deep learning*) corresponde a una rama del aprendizaje automático que ha emergido como un marco excepcionalmente poderoso y de propósito general para el análisis de datos [19]. Este enfoque se fundamenta en el uso de redes neuronales artificiales, modelos computacionales inspirados en los mecanismos de aprendizaje y procesamiento de información del cerebro humano. A diferencia de los algoritmos tradicionales, diseñados manualmente, las redes profundas son capaces de aprender representaciones jerárquicas y progresivamente más abstractas a partir de grandes volúmenes de datos, habilitando así el desarrollo de sistemas de inteligencia artificial con un alcance antes inconcebible [19].

El Perceptrón, introducido como el modelo más simple de neurona artificial, constituye la base histórica de estas arquitecturas. Formalmente, combina un conjunto de entradas (x_i) ponderadas por pesos (w_i), incorpora un sesgo (b) y aplica una función de activación para determinar la salida (Figura 2.2). Este modelo, aunque limitado a problemas linealmente separables, estableció los cimientos conceptuales que dieron origen a redes multicapa y, más adelante, a los modelos profundos actuales [19].

El Perceptrón es considerado el modelo básico de una red neuronal artificial, introducido por Rosenblatt en 1962 [19].

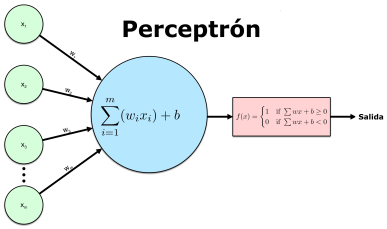


Figura 2.2: Representación esquemática de un perceptrón. Las entradas $x_1, x_2, \ldots, x_n$ son multiplicadas por sus respectivos pesos w_i y luego sumadas junto con un sesgo b. El resultado se procesa mediante una función de activación $f(\cdot)$ que determina la salida final.

Un perceptrón multicapa (*Multi-Layer Perceptron, MLP*), conecta múltiples perceptrones organizados en capas: una capa de entrada, una o más capas ocultas y una capa de salida (Figura 2.3). Gracias a la incorporación de funciones de activación no lineales, los MLP son capaces de aproximar funciones complejas y resolver problemas como la clasificación no lineal. El entrenamiento se realiza ajustando los pesos mediante algoritmos.

De forma más general, una red neuronal artificial puede visualizarse como un grafo compuesto por neuronas artificiales (nodos) dispuestas en capas, donde cada conexión transmite información en forma de pesos. El número de capas ocultas y de neuronas determina la profundidad y la capacidad de representación del modelo.

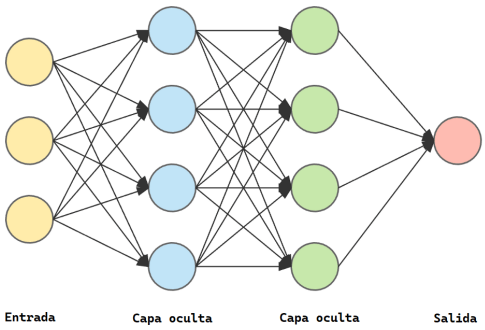
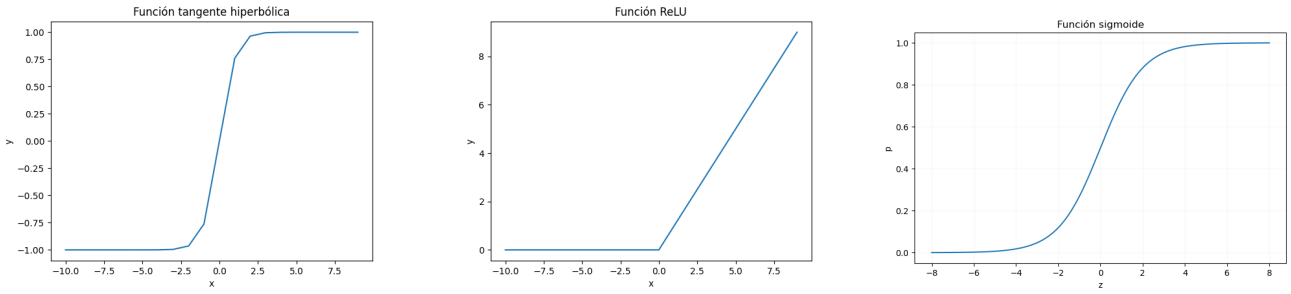


Figura 2.3: Estructura de un perceptrón multicapa (MLP). Los nodos amarillos representan la **capa de entrada**, los nodos azules y verdes corresponden a las **capas ocultas**, y el nodo rojo indica la **capa de salida**. Las conexiones entre capas transmiten información ponderada mediante los pesos.

Las funciones de activación introducen no linealidad en el modelo, lo que permite a las redes neuronales aproximar funciones complejas. Algunas son la tangente hiperbólica, ReLU y sigmoide (Figura 2.4).



(A) Función $\tanh(x)$ (B) Función ReLU (C) Función sigmoide

Figura 2.4: Funciones de activación consideradas en este trabajo: (A) tangente hiperbólica, (B) ReLU y (C) sigmoide.

- **Tangente hiperbólica**: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, con rango de salida en $[-1, 1]$, simétrica respecto al origen.
- **ReLU (Rectified Linear Unit)**: $f(x) = \max(0, x)$, computacionalmente eficiente y efectiva para mitigar el gradiente desvanecido.
- **Sigmoide**: $\sigma(x) = \frac{1}{1+e^{-x}}$, que transforma valores reales al rango $(0, 1)$, ampliamente usada en clasificación binaria.

En el desarrollo de este trabajo se priorizaron **ReLU** y **tanh**, dado que sus propiedades favorecen una propagación de gradientes más estable y una representación más expresiva en redes profundas [20]. En particular, la elección de tanh por sobre sigmoide responde a que sus salidas centradas en cero permiten un entrenamiento más equilibrado, reduciendo la saturación y la ralentización del aprendizaje [21].

Graph Neural Networks (GNN)

Las redes neuronales en grafos (*Graph Neural Networks, GNN*) constituyen una extensión del aprendizaje profundo diseñada para trabajar con datos estructurados en forma de grafo. A diferencia de las redes neuronales tradicionales, que procesan vectores o matrices (como imágenes o secuencias), las GNN operan directamente sobre grafos, donde los nodos representan entidades (por ejemplo, proteínas) y las **aristas** representan relaciones o interacciones (por ejemplo, enlaces en una red PPI).

El principio fundamental de una GNN es el message passing (propagación de mensajes). En este esquema, cada nodo actualiza su representación interna (embedding) al integrar información de sus vecinos en el grafo. De esta manera, las GNN permiten capturar tanto las propiedades locales de un nodo como el contexto topológico de su vecindad. Este proceso de agregación se repite a lo largo de varias capas, lo que posibilita aprender representaciones jerárquicas y multi-escala de los grafos.

Las GNN han demostrado gran versatilidad en distintos tipos de tareas (ver Figura 2.5) , entre las que destacan:

- **Clasificación de grafos**: asignar una etiqueta global a todo el grafo.
- **Clasificación de nodos**: predecir etiquetas individuales para cada nodo.
- **Predicción de enlaces**: inferir la probabilidad de que exista una arista entre dos nodos.
- **Detección de comunidades**: identificar subconjuntos de nodos densamente conectados.
- **Aprendizaje de representaciones (Graph Embedding)**: generar vectores de baja dimensión que capturen información estructural y de atributos.
- **Generación de grafos**: sintetizar nuevos grafos que respeten las propiedades estadísticas de los grafos de entrenamiento.

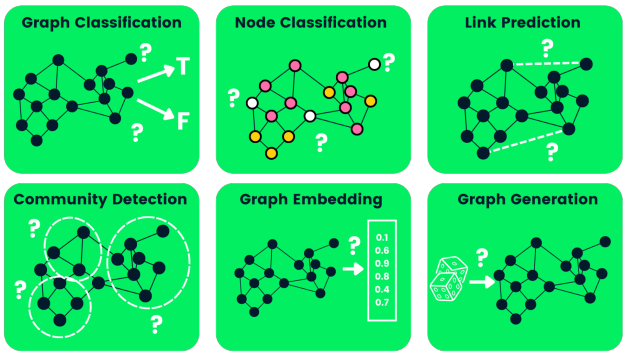


Figura 2.5: Principales tareas que pueden abordarse con Graph Neural Networks (GNN): clasificación de grafos, clasificación de nodos, predicción de enlaces, detección de comunidades, aprendizaje de embeddings y generación de grafos.

Fuente: Adaptado de Abid Ali Awan, *Comprehensive Introduction to Graph Neural Networks (GNNs) Tutorial*, DataCamp [22].

Convolutional Neural Network (CNN)

Las Convolutional Neural Networks (CNN) constituyen una de las arquitecturas más utilizadas en el aprendizaje profundo, particularmente en el procesamiento de imágenes. Su núcleo radica en la operación de convolución, que permite detectar patrones locales mediante la aplicación de filtros o *kernels* sobre regiones específicas de los datos de entrada [19].

Matemáticamente, dada una imagen $I \in \mathbb{R}^{m \times n}$ y un filtro $K \in \mathbb{R}^{p \times q}$, la operación de convolución discreta se define como:

$$(I * K)(i, j) = \sum_{u=0}^{p-1} \sum_{v=0}^{q-1} I(i + u, j + v) \cdot K(u, v) \tag{2.1}$$

donde (i, j) representan la posición espacial de la salida. Esta operación actúa como un detector de características locales (bordes, texturas), y al combinar múltiples filtros se obtiene una representación jerárquica de la entrada.

Graph Convolutional Networks (GCN)

Las Graph Convolutional Networks (GCN) son una variante de las redes neuronales en grafos que se caracterizan por realizar una agregación de información desde los nodos vecinos con un esquema de normalización (convolución). De esta manera, cada nodo actualiza su representación combinando sus propios atributos con los de su vecindario, ponderados según el grado de conectividad de los nodos involucrados. Esta normalización evita que nodos con un alto número de conexiones dominen el proceso de actualización, equilibrando así la influencia de cada vecino [23].

Formalmente, la operación de convolución sobre grafos puede expresarse como:

$$h_i = \sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{\deg(i) \cdot \deg(j)}} W h_j \tag{2.2}$$

donde $\deg(i)$ y $\deg(j)$ representan los grados de los nodos i y j, respectivamente, h_j corresponde al embedding del nodo vecino j, y W es la matriz de pesos aprendida durante el entrenamiento. El término $\frac{1}{\sqrt{\deg(i) \cdot \deg(j)}}$ actúa como factor de normalización, reduciendo el peso de los nodos con alta conectividad.

Una de las ventajas principales de las GCN es su eficiencia computacional, ya que el cálculo de todos los embeddings puede formularse como operaciones matriciales, lo que facilita su implementación en entornos de cómputo optimizados para la multiplicación rápida de matrices.

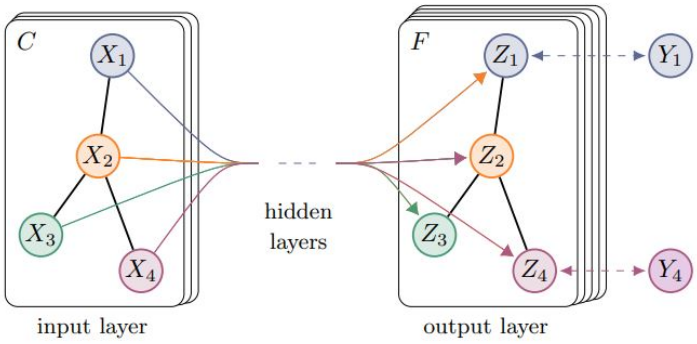


Figura 2.6: Representación esquemática de una *Graph Convolutional Network* (GCN). A la izquierda se muestra la capa de entrada, donde cada nodo X_i representa un vector de características iniciales. En las capas ocultas, la información de cada nodo se actualiza a partir de la agregación de sus vecinos en la red, aplicando pesos de normalización y la matriz de parámetros W. A la derecha se encuentra la capa de salida, donde cada nodo Z_i corresponde a un embedding transformado y se asocia con una etiqueta de predicción Y_i. Este esquema ilustra cómo las GCN permiten integrar simultáneamente atributos de los nodos y la estructura del grafo para tareas de clasificación. [23]

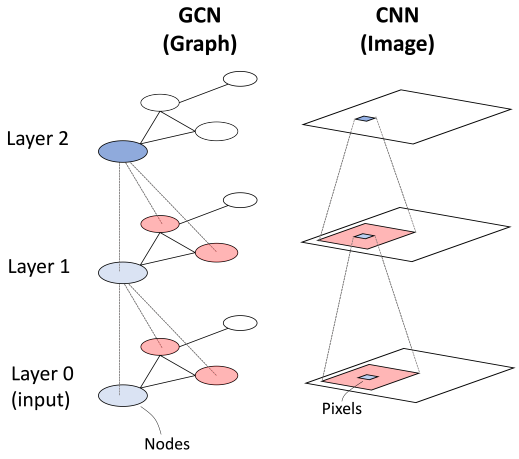


Figura 2.7: Esquema ilustrativo de una Graph Convolutional Network (GCN). Cada nodo actualiza su representación a partir de la información de sus vecinos, análogo a la forma en que las convoluciones en imágenes consideran los píxeles circundantes para extraer patrones locales.

Graph Attention Networks (GAT)

Las *Graph Attention Networks* (GAT) [24] introducen un mecanismo de atención en el proceso de agregación de información en los grafos. La idea principal es asignar un peso diferente a cada vecino de un nodo, de modo que la red pueda identificar cuáles conexiones son más relevantes para calcular su representación final.

Formalmente, dado un nodo i con embedding h_i y uno de sus vecinos j con embedding h_j, se define un coeficiente de atención α_{ij} que mide la importancia de j para la actualización de i. Este coeficiente se obtiene aplicando una transformación lineal a los embeddings, seguida de

una función de atención a, una activación *LeakyReLU* y finalmente una normalización de tipo softmax sobre el vecindario de i:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}(a(Wh_i, Wh_j))\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(\text{LeakyReLU}(a(Wh_i, Wh_k))\right)}. \tag{2.3}$$

Una vez obtenidos los coeficientes de atención, la actualización del embedding de i se realiza como una combinación ponderada de los embeddings de sus vecinos transformados, incluyéndose a sí mismo mediante una arista residual:

$$h_i' = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij} W h_j\right), \tag{2.4}$$

donde σ es una función de activación no lineal, usualmente *ReLU*. Este mecanismo permite que el modelo aprenda dinámicamente qué vecinos son más informativos, en lugar de tratarlos a todos con el mismo peso.

La Figura 2.8 ilustra este proceso:.

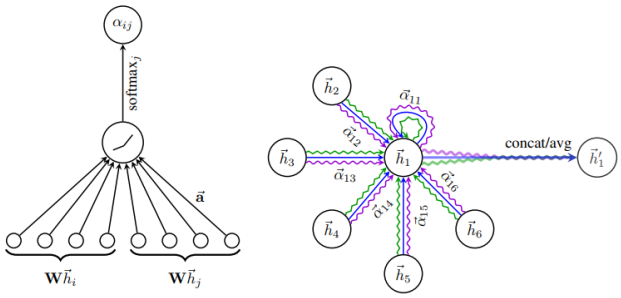


Figura 2.8: Esquema del mecanismo de atención en las *Graph Attention Networks* (GAT) [24]. **Izquierda**: cálculo del coeficiente de atención α_{ij} a partir de los embeddings transformados de un nodo i y su vecino j, utilizando una función de atención parametrizada por un vector de pesos y activada con *LeakyReLU*, seguida de una normalización *softmax* sobre el vecindario. **Derecha**: ejemplo de atención multi-cabeza (*multi-head attention*), donde un nodo combina la información de sus vecinos a través de varios mecanismos de atención independientes (indicados con distintos colores). Los embeddings resultantes de cada cabeza se concatenan o promedian para formar la nueva representación final del nodo.

Una limitación de usar un único mecanismo de atención es que la red podría enfocarse demasiado en un solo patrón de relaciones. Siguiendo la propuesta de los autores de GAT [24], se incorpora el mecanismo de **atención multi-cabeza** (*multi-head attention*), en la cual se calculan de manera independiente varios conjuntos de coeficientes α_{ij}, cada uno con sus propios parámetros de atención. El número de cabezas se controla mediante el hiperparámetro `num_heads`. Los resultados obtenidos de cada cabeza se concatenan (en capas intermedias) o se promedian (en la última capa).

En resumen:

- **Graph Neural Networks (GNN):** Marco general de modelos diseñados para operar sobre datos en forma de grafo. Basado en el esquema de *message passing*.

- **Graph Convolutional Networks (GCN):** Extienden la operación de convolución al dominio de grafos. Cada nodo agrega de forma normalizada la información de sus vecinos y de sí mismo.
- **Graph Attention Networks (GAT):** Introducen un mecanismo de atención en aristas. Cada nodo asigna pesos de importancia α_{ij} a sus vecinos, lo que permite focalizarse en interacciones más relevantes.

Predicción de enlaces en grafos

La predicción de enlaces (*Link Prediction*) [25] corresponde a la tarea de inferir si existe, o podría existir, una conexión entre dos nodos de un grafo. En otras palabras, busca anticipar relaciones que no aparecen explícitamente en los datos disponibles, pero que son consistentes con los patrones de la red y con los atributos de los nodos.

En el caso de las PPIs, esta tarea es especialmente útil porque la información experimental disponible suele estar incompleta: muchas interacciones aún no han sido descubiertas o no han sido confirmadas. Mediante *Link Prediction*, es posible proponer nuevas interacciones candidatas y, con ello, enriquecer la red original. Estas predicciones permiten priorizar conexiones con mayor probabilidad biológica, que luego pueden ser validadas experimentalmente.

Además, la predicción de enlaces no solo tiene un valor práctico en biología, sino que también se utiliza como objetivo de entrenamiento en arquitecturas de GNN. Al entrenar un modelo para distinguir entre enlaces existentes y no existentes, se generan representaciones internas (*embeddings*) que capturan tanto la estructura global del grafo como las relaciones locales entre nodos.

Tradicionalmente, la función de pérdida empleada para esta tarea es la entropía cruzada binaria (BCE), la cual busca asignar la etiqueta 1 a las aristas presentes y 0 a las ausentes:

$$BCE(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \big[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \big]. \tag{2.5}$$

Descenso del gradiente

El descenso del gradiente es la técnica fundamental para entrenar redes neuronales. Consiste en ajustar los parámetros del modelo en la dirección opuesta al gradiente de la función de pérdida, de manera que en cada iteración los pesos se muevan hacia valores que reducen el error. La regla básica de actualización se expresa como:

$$\theta_{t+1} = \theta_t - \eta \, \nabla_\theta J(\theta_t), \tag{2.6}$$

donde η representa la tasa de aprendizaje. A pesar de su simplicidad, esta estrategia puede ser ineficiente en problemas complejos, especialmente cuando los gradientes presentan alta varianza o cuando diferentes parámetros requieren escalas de aprendizaje distintas.

Para superar estas limitaciones se han desarrollado variantes del descenso del gradiente, entre ellas *Momentum*, *RMSProp* y la seleccionada para este trabajo, **Adam** (*Adaptive Moment Estimation*) [26].

Adam combina las ideas de momentum y RMSProp en un solo optimizador. Por un lado, mantiene un promedio móvil de los gradientes pasados (primer momento), lo que suaviza las

oscilaciones y acelera la convergencia. Por otro lado, conserva un promedio móvil de los gradientes al cuadrado (segundo momento), lo que permite asignar tasas de aprendizaje adaptativas a cada parámetro. Tras aplicar correcciones por sesgo, la regla de actualización final queda:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{2.7}$$

donde $\hat{m}_t$ y $\hat{v}_t$ son las estimaciones corregidas de primer y segundo momento, y ϵ es un pequeño valor que asegura la estabilidad numérica.

Regularización

Uno de los principales desafíos en el entrenamiento de modelos de aprendizaje profundo es prevenir el sobreajuste (*overfitting*), es decir, cuando la red aprende demasiado bien los patrones del conjunto de entrenamiento pero pierde capacidad de generalización frente a datos nuevos. Una técnica muy utilizada para mitigar este problema es el early stopping.

El *early stopping* consiste en monitorizar el desempeño del modelo sobre el conjunto de validación y detener el entrenamiento cuando dicho rendimiento deja de mejorar y comienza a deteriorarse. De este modo, se interrumpe el proceso antes de que el modelo memorice en exceso los datos de entrenamiento, actuando así como un mecanismo de regularización.

En la Figura 2.9 se ilustra este principio, donde el entrenamiento debe detenerse en el punto en que la validación alcanza su valor mínimo, aplicando la técnica de *early stopping*.

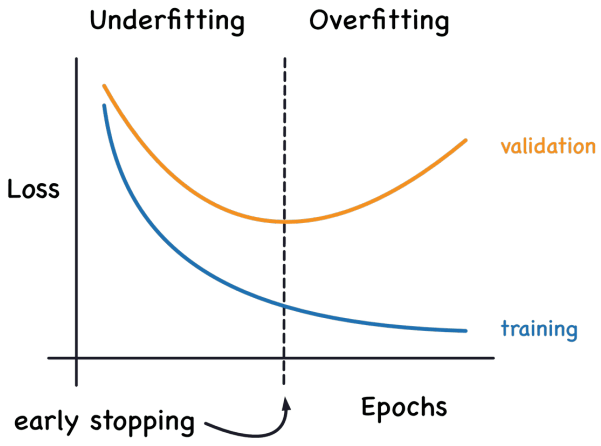


Figura 2.9: Esquema del comportamiento de las pérdidas de entrenamiento y validación en función de las épocas: La curva azul corresponde a la pérdida sobre el conjunto de entrenamiento, que tiende a decrecer de manera continua. En contraste, la curva naranja representa la pérdida de validación: al inicio también disminuye, pero tras cierto punto empieza a incrementarse, reflejando que el modelo está sobreajustando. El entrenamiento debe detenerse en el mínimo de la curva de validación, lo que permite evitar tanto el *underfitting* (modelo demasiado simple) como el *overfitting* (modelo demasiado específico a los datos de entrenamiento).

Optimización de hiperparámetros

Los hiperparámetros son valores configurados antes de iniciar el entrenamiento de un modelo, como la tasa de aprendizaje, el número de capas, el tamaño de cada capa o el coeficiente de regularización. A diferencia de los parámetros internos (pesos y sesgos), que se ajustan automáticamente mediante descenso de gradiente, los hiperparámetros deben definirse de forma explícita y su elección influye de manera determinante en el desempeño final.

Dado que el espacio de búsqueda de hiperparámetros suele ser amplio y no lineal, la optimización manual es poco eficiente. Una estrategia clásica es el *grid search*, que explora de forma exhaustiva una rejilla predefinida de combinaciones, aunque esto puede resultar muy costoso en términos de cómputo. Como alternativa, se emplean métodos más eficientes que asignan mayor esfuerzo de búsqueda a las regiones del espacio más prometedoras.

En este contexto, la librería Optuna [27] se ha consolidado como un marco de optimización de hiperparámetros flexible y eficiente. Optuna implementa algoritmos de *optimización bayesiana*, entre ellos el Tree-structured Parzen Estimator (TPE) [28]. Este enfoque construye dos distribuciones de probabilidad: una asociada a las configuraciones con mejor rendimiento y otra al resto. A partir de ellas, selecciona nuevas configuraciones maximizando la razón:

$$r(x|D) = \frac{p(x|D_g)}{p(x|D_b)}, \tag{2.8}$$

donde D_g corresponde al conjunto de configuraciones prometedoras y D_b al de configuraciones con peor desempeño. De esta forma, el algoritmo concentra la exploración en las regiones del espacio de hiperparámetros que tienen mayor probabilidad de producir buenos resultados [29].

Además, Optuna incorpora mecanismos de *early stopping* en las pruebas de configuraciones, descartando aquellas que desde las primeras épocas muestran un rendimiento inferior al promedio. Esto permite explorar un mayor número de combinaciones en menos tiempo, lo que resulta especialmente útil en modelos de alta complejidad como las GNN, donde la elección adecuada de hiperparámetros es crítica para obtener embeddings representativos y generalizables.

2.4. Clustering y reducción de dimensionalidad

Clustering corresponde a un conjunto de técnicas no supervisadas cuyo objetivo es agrupar datos en función de su similitud, de manera que los elementos dentro de un mismo grupo (o *cluster*) presenten mayor homogeneidad interna que con respecto a otros grupos. En el ámbito biológico, esta estrategia resulta fundamental para detectar módulos funcionales en redes PPI, donde se espera que proteínas con roles complementarios se agrupen en la misma comunidad.

Métodos de clustering

Entre los algoritmos clásicos de agrupamiento se encuentra **K-means** [30], el cual particiona los datos en k grupos, minimizando la distancia de cada punto al centroide de su cluster. Su hiperparámetro principal es el número de clusters (k), que debe fijarse de forma previa. La correcta elección de k resulta crítica: un valor demasiado bajo puede unir grupos distintos en un mismo cluster, mientras que un valor excesivo puede fragmentar la estructura en subgrupos poco representativos. Esta dependencia limita su aplicabilidad en contextos donde la cantidad

de clusters no es conocida de antemano o cuando las estructuras de los datos no son esféricas (Figura 2.11).

En contraste, los algoritmos basados en densidad no requieren especificar el número de clusters. Uno de los más utilizados es **DBSCAN** (*Density-Based Spatial Clustering of Applications with Noise*) [31]. Este algoritmo define clusters como regiones de alta densidad separadas por zonas menos densas. Sus hiperparámetros principales son: el radio de vecindad ε (`eps`), que determina el alcance en el que se buscan vecinos; y el número mínimo de puntos requeridos para considerar un nodo como núcleo (`min_samples`). Los clusters válidos surgen a partir de estos puntos núcleo, mientras que los puntos que no cumplen estos criterios se marcan como **ruido**. Una elección adecuada de ε y `min_samples` resulta fundamental: valores pequeños generan muchos clusters fragmentados, mientras que valores grandes pueden unir estructuras heterogéneas en un único grupo. En redes biológicas, este criterio es especialmente útil, ya que permite diferenciar proteínas altamente conectadas de aquellas periféricas que no pertenecen a módulos bien definidos (Figura 2.10).

Una extensión jerárquica de este enfoque es **HDBSCAN** (*Hierarchical DBSCAN*) [32], el cual construye una jerarquía de clusters en múltiples niveles de densidad y selecciona automáticamente aquellos más estables. Este método incorpora varios hiperparámetros adicionales que regulan la robustez y la granularidad de los resultados:

- `min_cluster_size`: tamaño mínimo permitido para que un conjunto de puntos sea considerado un cluster válido.
- `min_samples`: controla la tolerancia al ruido, influyendo en qué tan estrictamente se definen los puntos núcleo.
- `alpha`: parámetro que regula la suavidad de la jerarquía, afectando cómo se combinan los clusters en distintos niveles.
- `cluster_selection_epsilon`: ajusta la tolerancia para fusionar clusters cercanos (por defecto 0.0), lo que permite unir regiones densas separadas por pequeñas brechas.

Al igual que en DBSCAN, HDBSCAN identifica puntos ruidosos que no se asignan a ningún cluster. Su capacidad para detectar estructuras jerárquicas y diferenciar clusters válidos de ruido lo hace especialmente adecuado en redes PPI.

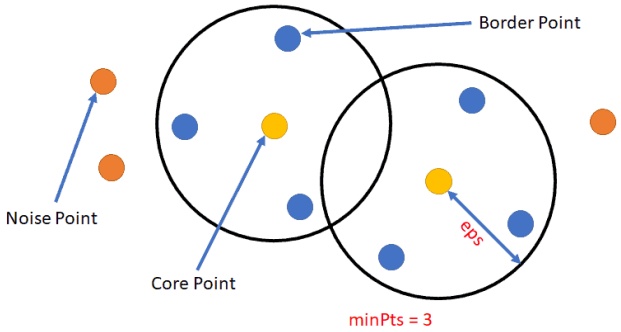


Figura 2.10: Esquema ilustrativo de *DBSCAN*. Los puntos azules representan *core points* (núcleos), los amarillos corresponden a *border points* (frontera) y los naranjos a *noise points* (ruido). El parámetro ε (`eps`) indica el radio máximo de vecindad, mientras que el número mínimo de vecinos requeridos (`minPts` o `minSamples`) determina si un punto puede ser considerado núcleo. Estos dos parámetros en conjunto definen la expansión de los clusters.

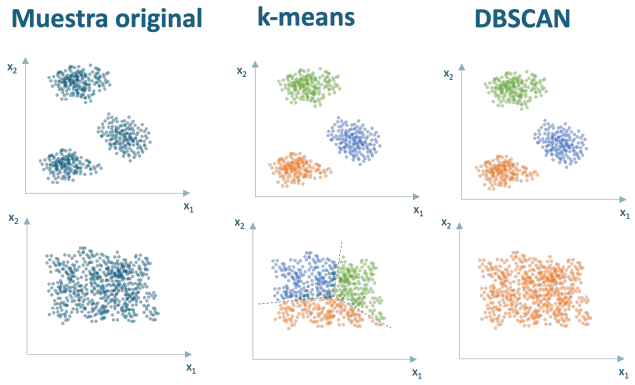


Figura 2.11: Comparación entre K-means y DBSCAN en un conjunto de datos bidimensional. Mientras que K-means requiere especificar el número de clusters, DBSCAN detecta automáticamente regiones densas y clasifica los puntos aislados como ruido.

Reducción de dimensionalidad con UMAP

UMAP (*Uniform Manifold Approximation and Projection*) es una técnica no lineal de reducción de dimensionalidad, cuyos fundamentos se inspiran en la geometría diferencial y la topología algebraica.

Formalmente, UMAP construye un grafo de vecinos más cercanos a partir de los datos en alta dimensión, donde las aristas codifican la probabilidad de similitud entre puntos. Posteriormente, busca una representación en un espacio de menor dimensión mediante la optimización de una función de costo que intenta mantener esas relaciones de similitud en la nueva proyección. De este modo, UMAP preserva tanto la forma de los grupos de datos como su organización general.

En términos prácticos, esto significa que UMAP no solo mantiene la cercanía entre puntos similares (estructura local), sino que también conserva relaciones más amplias entre grupos de datos (estructura global). UMAP generalmente preserva la estructura global de los datos en lugar de limitarse únicamente a las estructuras locales [33].

La Figura 2.12 ilustra este proceso, mostrando cómo una nube de puntos en tres dimensiones puede ser proyectada a un espacio bidimensional sin perder la coherencia de los *clusters* detectados. Esta capacidad de representar estructuras complejas en 2D constituye una herramienta valiosa para validar visualmente la consistencia de los módulos identificados y facilitar su interpretación biológica.

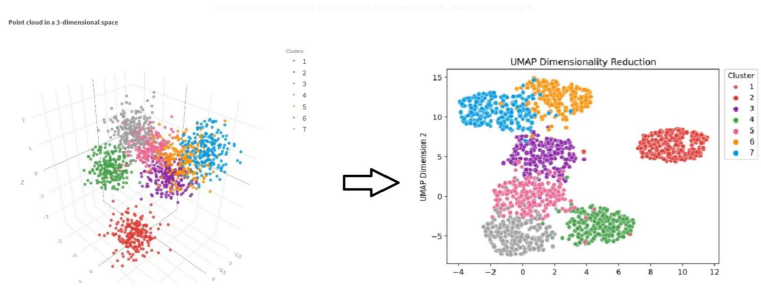


Figura 2.12: Ejemplo de reducción de dimensionalidad con UMAP. Una nube de puntos en tres dimensiones (izquierda) es proyectada a un espacio bidimensional (derecha), manteniendo la coherencia de los clusters originales.

2.5. Métricas de evaluación

Conjunto de datos

Para comprender las métricas, es crucial comprender cómo se utilizan los diferentes conjuntos de datos:

- **Conjunto de entrenamiento (Train Set, 80 %):** Se utiliza para que el modelo aprenda durante el proceso de entrenamiento.
- **Conjunto de validación (Validation Set, 10 %):** Se usa para ajustar los parámetros del modelo y tomar decisiones durante el entrenamiento.
- **Conjunto de Prueba (Test Set, 10 %):** Corresponde a un conjunto independiente y no visto por el modelo durante el entrenamiento ni la validación, utilizado como evaluación final e imparcial del rendimiento.

Métricas para el entrenamiento de GNN

Durante el entrenamiento de GNN, se emplean métricas de evaluación que permiten cuantificar el desempeño del modelo y guiar el proceso de optimización. Estas métricas son calculadas de manera continua tanto en el conjunto de entrenamiento como en el de validación. De esta forma, entregan información esencial para el ajuste de hiperparámetros y la prevención de sobreajuste.

La evaluación de un modelo se basa en la comparación entre las predicciones realizadas y las etiquetas reales. Para ello se definen:

- TP (*True Positives*): número de instancias positivas correctamente clasificadas como positivas.
- TN (*True Negatives*): número de instancias negativas correctamente clasificadas como negativas.
- FP (*False Positives*): número de instancias negativas clasificadas incorrectamente como positivas.
- FN (*False Negatives*): número de instancias positivas clasificadas incorrectamente como negativas.

En este trabajo, el entrenamiento se realizó mediante un esquema compuesto por un codificador basado en capas `GATConv` y un decodificador de tipo `LinkPredictor`. En este contexto, las etiquetas reales corresponden a la existencia o ausencia de enlaces en la red PPI original. Es decir, cada par de proteínas (u, v) posee una etiqueta y_{uv} que toma el valor 1 si en la red existe efectivamente una interacción, y 0 en caso contrario (pares muestreados como negativos). El modelo genera embeddings a través de las capas `GATConv`, y posteriormente el `LinkPredictor` estima la probabilidad $\hat{y}_{uv}$ de que exista un enlace entre dos nodos. La comparación entre $\hat{y}_{uv}$ y y_{uv} permite calcular las métricas de desempeño (TP, TN, FP, FN). De esta forma, aunque la codificación ocurre a nivel de nodos, la supervisión del aprendizaje se da a nivel de aristas, utilizando como referencia las interacciones conocidas de la red biológica.

A partir de estos valores se definen las métricas más relevantes:

- **Accuracy (Exactitud):** Mide la proporción de predicciones correctas sobre el total de instancias:
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precision (Precisión):** Indica la fracción de predicciones positivas que realmente pertenecen a la clase positiva:
$$\text{Precision} = \frac{TP}{TP + FP}$$
- **Recall (Sensibilidad):** Evalúa la capacidad del modelo para identificar correctamente los casos positivos reales:
$$\text{Recall} = \frac{TP}{TP + FN}$$
- **F1-Score:** Media armónica entre precisión y *recall*:
$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
- **AUC-ROC (Área bajo la curva ROC):** Representa el área bajo la curva que grafica la Tasa de Verdaderos Positivos (TPR) frente a la Tasa de Falsos Positivos (FPR):
$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}$$
- **Función de pérdida (Loss):** Es la función objetivo que el modelo minimiza durante el entrenamiento.

Métricas para evaluación de *clustering*

Para evaluar la cohesión y separación de los *clusters* obtenidos a partir de los *embeddings*, se utilizan métricas internas, es decir, que no requieren etiquetas externas:

- **Silhouette Score:** Mide la similitud de un objeto con su propio *cluster* en comparación con otros. Toma valores entre -1 (agrupación deficiente) y +1 (agrupación óptima), mientras que 0 indica solapamiento entre *clusters*.
- **Davies-Bouldin Index (DBI):** Cuantifica la relación entre la dispersión intra-*cluster* y la separación inter-*clusters*. Valores más bajos indican mejor definición y compacidad de los *clusters*.
- **Calinski-Harabasz Index (CHI):** Evalúa la razón entre la varianza inter-*cluster* y la varianza intra-*cluster*. Valores altos reflejan agrupamientos bien definidos.
- **Método del Codo (Elbow Method):** Aplicado principalmente a K-means, permite estimar el número óptimo de *clusters* observando el punto de inflexión en la curva de inercia (suma de cuadrados de las distancias) frente al número de *clusters*.

2.6. Fundamento estadístico

El análisis de enriquecimiento funcional se basa en comparar la ocurrencia observada de un término dentro de un conjunto de interés (por ejemplo, un módulo o grupo de proteínas) con la ocurrencia esperada bajo un modelo de asignación aleatoria. En este contexto, la hipótesis nula establece que los términos GO se distribuyen de manera aleatoria, por lo que la frecuencia observada no difiere significativamente de la esperada. Para contrastar esta hipótesis se emplean distintos criterios estadísticos, entre los cuales destacan los siguientes:

Prueba hipergeométrica. Evalúa la probabilidad de observar una cantidad igual o mayor de ocurrencias de un término específico en un conjunto, bajo el supuesto de muestreo aleatorio sin reemplazo [34]. La prueba depende de los siguientes parámetros: N (tamaño del universo), M (número de elementos en N asociados al término), n (tamaño del subconjunto) y k (ocurrencias observadas del término en el subconjunto). La probabilidad se calcula mediante la distribución hipergeométrica:

$$p = \mathbb{P}(X \geq k) = \sum_{x=k}^{\min(n,M)} \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}}. \tag{2.9}$$

Este valor corresponde al P-value crudo, que mide la significancia estadística del enriquecimiento.

Z-score. El z-score cuantifica la desviación del conteo observado respecto al esperado, estandarizado por la varianza bajo la hipótesis nula [35].

$$z = \frac{k - \mathbb{E}[k]}{\sqrt{\mathrm{Var}[k]}}, \quad \text{donde} \quad \mathbb{E}[k] = n \cdot \frac{M}{N}, \quad \mathrm{Var}[k] = n \cdot \frac{M}{N} \cdot \left(1 - \frac{M}{N}\right) \cdot \frac{N-n}{N-1}. \tag{2.10}$$

Corrección por comparaciones múltiples. Dado que en un análisis de enriquecimiento se evalúan decenas o cientos de términos de manera simultánea, es necesario ajustar los P-values para controlar la tasa de falsos descubrimientos (FDR). Una de las estrategias utilizadas es el método de Benjamini–Hochberg [36], que fija un umbral de significancia ajustado, típicamente $\alpha = 0{,}05$.

Combined Score. Para integrar la significancia estadística (P-value corregido) con la magnitud del efecto (z-score), se utiliza el *combined score* [37], definido como:

$$\text{Combined Score} = -\log_{10}(P\text{-value corregido}) \cdot z. \tag{2.11}$$

Esta métrica favorece términos que son a la vez estadísticamente significativos y fuertemente sobrerrepresentados en el conjunto analizado.

2.7. Estado del arte

En la Tabla 2.1 se resume la revisión bibliográfica que abarca el periodo comprendido entre los años 2000 y 2024, centrada en el estudio de algoritmos para la detección de comunidades o módulos funcionales en redes PPI. Estos enfoques se basan principalmente en dos perspectivas: por un lado, la estructura topológica de la red, y por otro, la combinación de dicha topología con la funcionalidad biológica de los nodos, con el fin de enriquecer la interpretación de la red [38].

Referencia	**Año**	**Enfoque**
MCL [39]	2000	Topología
MCODE [35]	2003	Topología
RNSC [40]	2004	Topología + Funcionalidad
LCMA [41]	2005	Topología
CFinder [42]	2005	Topología
COACH [43]	2009	Topología
ClusterONE [44]	2012	Topología
GMFTP [45]	2014	Topología + Funcionalidad
DCAFP [46]	2015	Topología + Funcionalidad
MTGO [10]	2018	Topología + Funcionalidad
LCDA-GO [38]	2022	Topología + Funcionalidad
DC-GC-ANN [47]	2023	Topología + Funcionalidad + DL
digID [48]	2024	Topología + Funcionalidad + DL

Tabla 2.1: Resumen de algoritmos para detección de módulos en redes PPI (2000–2024).

Fuente: Elaboración propia.

Enfoques basados en la topología de la red

En cuanto a los enfoques centrados en la topología, estos se fundamentan en propiedades estructurales de las redes PPI, tales como la densidad de conexión, la conectividad global y

la existencia de subgrafos densamente interconectados, bajo el supuesto de que los complejos proteicos se manifiestan como regiones densamente conectadas [35, 41]. Históricamente, los primeros algoritmos de detección de módulos se centraban exclusivamente en la topología de la red [49, 38].

Entre los métodos existentes se incluyen **MCL (Markov Cluster Algorithm)** [39], algoritmo de agrupamiento no supervisado, rápido y escalable, basado en la simulación de flujos estocásticos, emplea operadores de "expansión" e "inflación" para identificar clústeres, generando comúnmente agrupamientos no superpuestos [39, 50]. **MCODE (Molecular COmplex DEtection)** [35], detecta regiones densamente conectadas en grandes redes PPI mediante la ponderación de vértices según la densidad de vecindario local, seguido de un recorrido desde una proteína definida como punto de partida. **LCMA (Local Clique Merging Algorithm)** [41], algoritmo basado en minería de grafos que detecta vecindarios densos en redes PPI, localizando y fusionando cliques locales (subgrafos completamente conectados). **CFinder** [42], se basa en un método de percolación de cliques para detectar comunidades superpuestas en redes complejas. **COACH** [43], propone un método de identificación mediante la detección de un núcleo denso y la adición de proteínas periféricas. **ClusterONE** [44], diseñado para identificar nodos potencialmente superpuestos, emplea una estrategia que considera tanto la densidad como la cohesión local.

A pesar de su eficiencia en la identificación de regiones densas, los métodos puramente topológicos presentan limitaciones respecto a su relevancia biológica, dado que las comunidades detectadas no siempre reflejan funciones proteicas compartidas [38, 49].

Enfoques Híbridos: Integrando Topología y Funcionalidad

La disponibilidad de bases de datos funcionales, como los términos GO, ha permitido enriquecer estos enfoques mediante la incorporación de información biológica, impulsando la integración de aspectos funcionales con propiedades topológicas [10, 38, 49, 18].

Uno de los primeros esfuerzos en esta dirección fue **RNSC (Restricted Neighborhood Search Clustering Algorithm)** [40],un algoritmo de agrupamiento que optimiza una función de costo, dividiendo la red en subgrupos y refinando las particiones de acuerdo con criterios de tamaño, densidad y homogeneidad funcional. Posteriormente, tras una década, surgieron propuestas como **GMFTP**[45], basado en un modelo generativo, y **DCAFP** [46], centrado en técnicas de clustering, ambos lograron combinar la información funcional y topológica, mejorando la detección de proteínas solapadas. **MTGO (Module detection via Topological information and GO knowledge)** [10] es un algoritmo que combina la información funcional (términos GO) con propiedades topológicas para detectar módulos funcionales, comienza con una partición aleatoria y asigna proteínas a los módulos si estos comparten una función GO. Por su parte, **LCDA-GO (Local Community Detection Algorithm with GO)** [38] propone un enfoque local que integra funcionalidad GO sin depender de particiones aleatorias, se enfoca en comunidades locales a partir de puntos de partida bien definidos.

Aplicación de Deep Learning en la detección de módulos PPI

En los últimos años, el aprendizaje profundo (deep learning, DL) ha emergido como una herramienta poderosa en el análisis genómico, destacando por su capacidad para capturar patrones

complejos y relaciones no lineales en grandes volúmenes de datos biológicos [51]. En este contexto, destaca los siguientes desarrollos recientes: **DC-GC-ANN (Degree Centrality-Graph Colouring-Artificial Neural Network) [47]**, propone un modelo que combina topología de red con deep learning para predecir genes candidatos asociados a enfermedades complejas, aplicado como caso de estudio a la EA. Utiliza medidas topológicas como la centralidad de grado y el coloreado de grafos (asignación de colores a nodos de manera que dos nodos adyacentes no compartan el mismo color, una técnica de teoría de grafos), integradas en una GNN, logrando una precisión del 96 % en la identificación de genes candidatos. **digID (Disease Gene Identification)[48]**, introduce un marco computacional que emplea un clasificador deep learning semi-supervisado para predecir genes, validado en el contexto de la EA. Posteriormente, realiza un análisis sobre redes PPI para priorizar su relevancia funcional.

Como resultado de esta revisión, se observa una evolución progresiva desde enfoques basados únicamente en la topología hacia modelos híbridos que integran información funcional y, más recientemente, técnicas de aprendizaje profundo. En el presente proyecto se propone implementar un enfoque inspirado en los algoritmos **MTGO** [10] y **LCDA-GO** [38], no por el uso de sus metodologías específicas, sino por la lógica que plantean al combinar atributos topológicos con información funcional derivada de GO. La contribución de este trabajo radica en extender dicha integración hacia un marco basado en Deep Learning.

Capítulo 3. Conjunto de datos

Este capítulo presenta en detalle el conjunto de datos utilizado en este estudio, el cual constituye la base para el desarrollo y evaluación del modelo propuesto.

El conjunto de datos utilizado en este estudio proviene del repositorio público del Laboratorio del Dr. David Ramírez[1], y corresponde a una selección de información biomolecular relacionada con la enfermedad de Alzheimer. Dicho repositorio compila información obtenida a partir de bases de datos biológicas públicas utilizadas en farmacología de sistemas, tales como *ChEMBL*, *Therapeutic Target Database (TTD)*, *Uniprot*, *STRING*, *Complex Portal* y *Open Targets Platform* .

Posteriormente, esta base de datos fue sometida a un preprocesamiento exploratorio, con el objetivo de comprender su estructura y extraer atributos relevantes para la posterior modelación mediante GNN (ver notebook 01_Preprocesamiento en el repositorio de GitHub del proyecto).

3.1. Descripción de variables

A continuación, se describen los atributos más relevantes contenidos en los cuatro archivos que conforman el conjunto de datos.

- **interaction_score**: Valor numérico asociado a cada par de proteínas en la red PPI. Representa una medida continua de confianza de interacción entre proteínas.
- **target_type**: Clasifica a cada blanco terapéutico según su naturaleza estructural. Los valores posibles son:
  - *SINGLE PROTEIN*: proteína individual como blanco.
  - *COMPLEX PROTEIN*: complejo proteico que actúa como unidad funcional.
- **target_group**: Variable categórica que representa el nivel de validación terapéutica de cada proteína, según criterios establecidos por el Ramírez Lab. Las clases posibles se resumen como:
  - `T1`: Blancos moleculares de fármacos para el tratamiento de la EA que actualmente se encuentran en fase IV (aprobados por la FDA)[2]
  - `T2`: Blancos moleculares de compuestos en etapas clínicas (fase I, II o III).
  - `T3`: Blancos moleculares de compuestos en etapa preclínica o fase 0, evaluados en modelos biológicos no humanos.

[1] https://github.com/AlePV/Disease-Target-Tracker
[2] Food and Drug Administration (FDA)

- `T4`: Proteínas que interactúan con T1, T2 y/o T3, pero para las cuales no existe evidencia en la base de datos ChEMBL (base de datos de interacciones droga-proteína) respecto de ensayos clínicos o preclínicos en EA.

Esta variable puede presentar múltiples clases simultáneamente por proteína.

- **target_group_score_normalized**: Variable numérica continua que cuantifica el nivel de validación terapéutica de cada combinación de clases en `target_group`, en una escala normalizada entre 0 y 1. Este score permite modelar transiciones graduales entre grupos, por ejemplo, entre T2 y T3.

- **DEG**: Acrónimo de *Differentially Expressed Genes*. Esta variable indica si una proteína está asociada a un gen cuya expresión ha sido reportada como diferencial en el contexto de la EA. Para ello se utilizaron conjuntos de datos transcriptómicos de cerebro completo del *Mount Sinai Hospital*, *Mayo Clinic* y el consorcio *ROSMAP*, que incluyen individuos sanos, con deterioro cognitivo leve y pacientes con Alzheimer [52].

  DEG toma tres posibles valores:

  - `none`: no se encuentra diferencialmente expresada.
  - `Up`: expresión aumentada (sobreexpresada) en condiciones patológicas.
  - `Down`: expresión disminuida (subexpresada) respecto a condiciones normales.

  Todas las proteínas DEG en este conjunto pertenecen al grupo terapéutico T4.

3.2. Análisis exploratorio

La variable `Interaction_Score` cuantifica la confianza o fortaleza de cada interacción de la red PPI.

La Figura 3.1 muestra su distribución. Se observa que, al inicio de la escala ($\approx$ 0.40–0.60), existe una concentración principal de interacciones, con frecuencias que oscilan aproximadamente entre 20 000 y 35 000 ocurrencias. Posteriormente, la distribución presenta una cola decreciente hacia valores más altos y, finalmente, un pico secundario cercano a 1.0. Lo anterior indica que la red contiene muchas interacciones de confianza moderada y un subconjunto menor de interacciones altamente confiables.

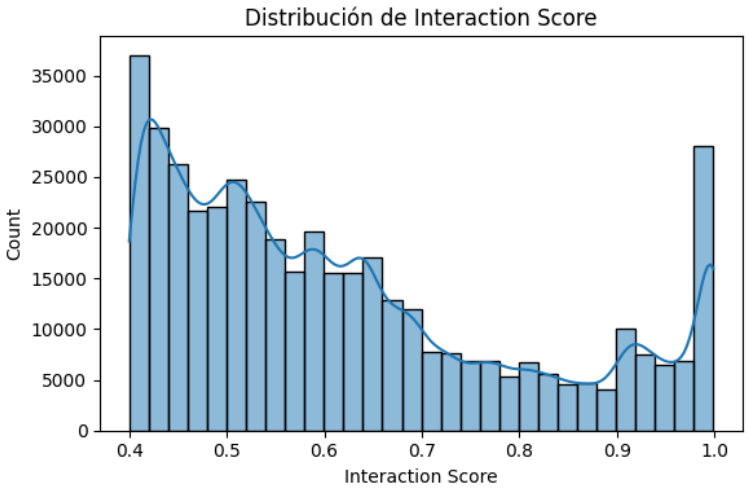


Figura 3.1: Distribución de *Interaction_Score*.

Fuente: Elaboración propia.

Uno de los atributos clave es el tipo de blanco molecular, representado en la variable `Target_type`, donde se observó un predominio de blancos de tipo *single protein*, seguido por *complex protein*, como se muestra en la Tabla 3.1.

Target_type	**Frecuencia**	**Porcentaje (%)**
SINGLE PROTEIN	3,916	72.65 %
COMPLEX PROTEIN	1,474	27.35 %

Tabla 3.1: Distribución de clases en la variable `Target_type`.

Fuente: Elaboración propia.

En cuanto al atributo `Target_group`, que clasifica las proteínas según su nivel de validación terapéutica, se evidenció un marcado desbalance: el grupo T4, correspondiente a blancos emergentes con poca evidencia experimental, concentra el 70.93 % de los casos (Ver Tabla 3.2).

Target_group	Frecuencia	Porcentaje (%)
T4	3,823	70.93 %
T2	1,086	20.15 %
T2, T3	200	3.71 %
T2, T4	199	3.69 %
T2, T3, T4	28	0.52 %
T1, T2, T3	22	0.41 %
T3	15	0.28 %
T1, T2	9	0.17 %
T1, T2, T3, T4	7	0.13 %
T3, T4	1	0.02 %

Tabla 3.2: Distribución de clases en la variable `Target_group`.

Fuente: Elaboración propia.

Este patrón se reafirma al considerar la frecuencia individual de aparición de cada clase, incluyendo combinaciones, donde T4 está presente en el 68.55 % de las proteínas, mientras que T1 (blancos clínicamente validados) apenas alcanza el 0.64 % (Tabla 3.3).

Clase	Frecuencia	Porcentaje (%)
T1	38	0.64 %
T2	1551	26.20 %
T3	273	4.61 %
T4	4058	68.55 %

Tabla 3.3: Frecuencia individual de aparición por clase en `Target_group`.

Fuente: Elaboración propia.

La variable `Target_group_score_normalized` representa una estimación continua del nivel de validación terapéutica de cada combinación de clases. Como se observa en la Tabla 3.4, los grupos con presencia de T1 obtienen los puntajes más altos, mientras que T4 aislado presenta el valor más bajo.

Target_group	Score normalizado
T1, T2, T3, T4	1.000000
T1, T2, T3	0.994375
T1, T2	0.991667
T1	0.987500
T2, T3, T4	0.787500
T2, T4	0.766667
T2, T3	0.763889
T2	0.750000
T3, T4	0.516667
T3	0.500000
T4	0.250000

Tabla 3.4: Valor único de `Target_group_score_normalized` por combinación.

Fuente: Elaboración propia.

La variable `DEG` permite identificar proteínas cuya expresión diferencial ha sido reportada en el contexto de la EA. En total, existen 158 proteínas diferencialmente expresadas, lo que representa el 2.93 % del total de proteínas del conjunto de datos, tal como se muestra en la Tabla 3.5.

Estado DEG	Frecuencia	Porcentaje (%)
DEG (total)	158	2.93 %
No DEG	5,232	97.07 %

Tabla 3.5: Cantidad total de proteínas diferencialmente expresadas.

Fuente: Elaboración propia.

Del total de proteínas DEG, se observa la presencia tanto de regulación negativa (`Down`) como positiva (`Up`). La distribución detallada se muestra en la Tabla 3.6.

Tipo de regulación	Frecuencia	Porcentaje (%)
Down	91	1.69 %
Up	67	1.24 %

Tabla 3.6: Distribución de proteínas DEG según tipo de regulación.

Fuente: Elaboración propia.

Un hallazgo relevante es que el 100 % de las proteínas DEG pertenecen al grupo terapéutico `T4`, correspondiente a blancos poco estudiados. Este resultado sugiere que, a pesar de su escasa validación experimental, estas proteínas muestran alteraciones transcriptómicas relevantes, por lo que podrían representar blancos emergentes de alto interés para la investigación.

3.3. Características de la red y anotaciones funcionales

La red PPI analizada consta de 5,390 proteínas (nodos) conectadas mediante 430,206 interacciones (aristas). Las proteínas están anotadas con un total de 6,928 términos GO, distribuidos entre las tres ontologías principales (ver Tabla 3.7).

Ontología	Frecuencia
Biological Process (BP)	5,352
Molecular Function (MF)	1,109
Cellular Component (CC)	467

Tabla 3.7: Distribución de términos GO por tipo de ontología.

Fuente: Elaboración propia.

3.4. Archivos de entrada para el modelo

A partir de la red PPI de EA previamente construida por *Ramírez Lab*, se generaron y organizaron cuatro archivos estructurados para su utilización en el entrenamiento del modelo GNN (ver notebook 02_Construccion_DataSet en el repositorio de GitHub del proyecto).

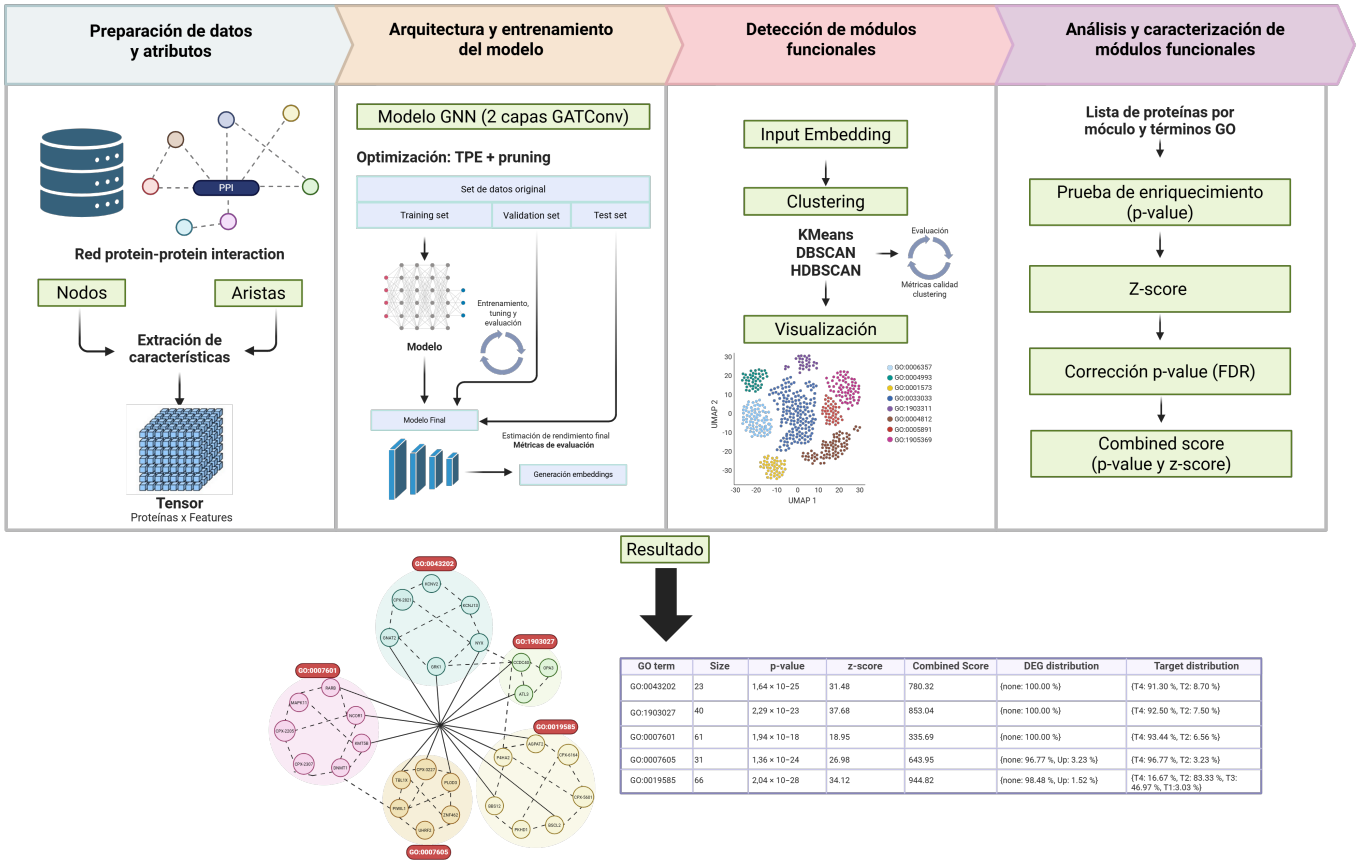
- **Red de interacción (Edge.csv)**: contiene las aristas del grafo con columnas `proteina1`, `proteina2` e `interaction_score`. Define la estructura de la red.
- **Anotaciones GO (GO.csv)**: contiene la relación de cada proteína con uno o más términos GO. Se utiliza para construir atributos funcionales multi-hot por nodo.
- **Metadatos de proteínas (metadata_proteina.csv)**: incluye variables como `target_type`, `target_group`, `target_group_score_normalized` y `DEG`. Estos atributos fueron seleccionados para complementar las características funcionales y aportar información sobre validación terapéutica.
- **Metadatos de términos GO (metadata_go.csv)**: contiene el identificador del término GO y su nombre asociado. Aunque el archivo incluye estadísticas como *P*`-value`, `z-score` y `combined score`, dichas métricas no fueron utilizadas como entrada del modelo GNN. En cambio, se calcularán posteriormente a partir de los módulos detectados, con el fin de realizar análisis de enriquecimiento funcional y caracterizar los módulos de proteínas obtenidos.

Los atributos `target_group` y `target_group_score_normalized`, aunque reflejan la misma dimensión biológica (nivel de validación terapéutica), aportan información complementaria. El primero constituye una variable categórica que agrupa las proteínas en niveles discretos de validación, útil para representar diferencias claras entre grupos. El segundo ofrece una estimación continua y normalizada, capaz de capturar gradientes y matices intermedios dentro de dichos grupos. De esta forma, al incluir ambas representaciones en el vector de atributos, el modelo integra simultáneamente relaciones discretas y continuas, lo que enriquece la caracterización de los nodos y permite una mejor transferencia de información durante el entrenamiento de la GNN.

Capítulo 4. Metodología

El presente estudio se fundamenta en un pipeline computacional estructurado, diseñado para la detección y caracterización de módulos funcionales en redes PPIs. Como se observa en la Figura 4.1, el flujo metodológico incluye: (1) la preparación de datos y atributos a partir de redes PPI y metadatos proteicos; (2) el entrenamiento de una GNN basada en capas GATConv, optimizada mediante búsqueda bayesiana; (3) la detección de módulos funcionales mediante reducción de dimensionalidad y clustering; y (4) el análisis funcional de los módulos, basado en enriquecimiento GO y caracterización biológica.

La metodología propuesta se implementó íntegramente en un repositorio público de GitHub, el cual recopila los notebooks, scripts y archivos de entrada/salida utilizados en cada etapa del pipeline: github.com/macamadridb/alzheimer-target-prediction.



GO term	Size	p-value	z-score	Combined Score	DEG distribution	Target distribution
GO:0043202	23	1,64 × 10−25	31.48	780.32	{none: 100.00 %}	{T4: 91.30 %, T2: 8.70 %}
GO:1903027	40	2,29 × 10−23	37.68	853.04	{none: 100.00 %}	{T4: 92.50 %, T2: 7.50 %}
GO:0007601	61	1,94 × 10−18	18.95	335.69	{none: 100.00 %}	{T4: 93.44 %, T2: 6.56 %}
GO:0007605	31	1,36 × 10−24	26.98	643.95	{none: 96.77 %, Up: 3.23 %}	{T4: 96.77 %, T2: 3.23 %}
GO:0019585	66	2,04 × 10−28	34.12	944.82	{none: 98.48 %, Up: 1.52 %}	{T4: 16.67 %, T2: 83.33 %, T3: 46.97 %, T1:3.03 %}

Figura 4.1: Diagrama general del pipeline computacional propuesto

Entorno computacional

- **Sistema operativo:** Windows 10 (versión 10.0.19045)
- **Arquitectura:** AMD64
- **CPU:** Intel64 Family 6 Model 140 Stepping 1 (8 núcleos lógicos)
- **Memoria RAM:** 11.79 GB
- **Unidad de procesamiento utilizada:** CPU (sin GPU disponible)
- **Lenguaje de programación:** Python 3.11.0
- **Librerías principales:**
  - **PyTorch** y **PyTorch Geometric**: para la construcción y entrenamiento de modelos GNNs, específicamente mediante capas `GATConv`.
  - **Optuna**: para la búsqueda automatizada de hiperparámetros, utilizando el algoritmo TPE y técnicas de pruning.
  - **scikit-learn**: para preprocesamiento de datos, clustering, métricas de evaluación y reducción de dimensionalidad.
  - **pandas** y **numpy**: para la manipulación de datos tabulares y operaciones numéricas.
  - **matplotlib** y **seaborn**: para la visualización de resultados y análisis gráfico.
  - **scipy** y **statsmodels**: para análisis estadístico, incluyendo pruebas de enriquecimiento funcional y corrección por múltiples hipótesis.

4.1. Preparación de datos y atributos

La transformación de datos biológicos en una representación computacional orientada a grafos es un paso crítico para el éxito del modelo. En esta etapa se construye un grafo no dirigido donde cada nodo representa una proteína y cada arista representa una interacción entre dos proteínas, con un atributo de peso (interaction_score) (ver script data_preprocessing.py en el repositorio de GitHub del proyecto).

Representación de nodos y sus atributos (x)

A cada nodo se le asocia un vector de características que incorpora información de la proteína correspondiente. Para construir la matriz de características de los nodos (x), se consideran los siguientes atributos:

- **Términos GO:** Cada proteína puede estar asociada a múltiples términos GO. Esta relación multi-etiqueta se codifica utilizando `MultiLabelBinarizer`, generando una representación multi-hot que preserva la pertenencia simultánea a varios términos funcionales.
- **Target_type:** Atributo categórico que describe si la proteína es un objetivo único (`single`) o parte de un complejo (`complex`). Se codifica con `One-Hot Encoding` a valores numéricos discretos.

- **Target_group:** Describe la validación de la proteína como blanco terapéutico, pudiendo pertenecer a uno o más grupos (T1, T2, ...). Esta variable multivaluada se codifica también con `MultiLabelBinarizer`. Las proteínas sin asignación explícita se codifican con vectores cero.

- **Target_group_score_normalized:** Valor numérico continuo que indica la confianza asociada al grupo terapéutico de la proteína. Este valor se normaliza con `StandardScaler` para garantizar compatibilidad en escala con el resto de atributos.

- **DEG:** Atributo categórico que indica si la proteína está diferencialmente expresada como `up-regulated`, `down-regulated` o `none`. Se codifica numéricamente con `LabelEncoder`, asignando el valor "none" en los casos faltantes.

Finalmente, todos los atributos recopilados se integran en una única estructura matricial. El resultado de este proceso es la matriz $X \in \mathbb{R}^{N \times F}$ (Figura 4.2), donde $N = 5390$ corresponde al número de proteínas y $F \approx 6935$ al total de características codificadas. Dicha matriz se transforma en un tensor de tipo `torch.float` para su procesamiento posterior en el modelo GNN.

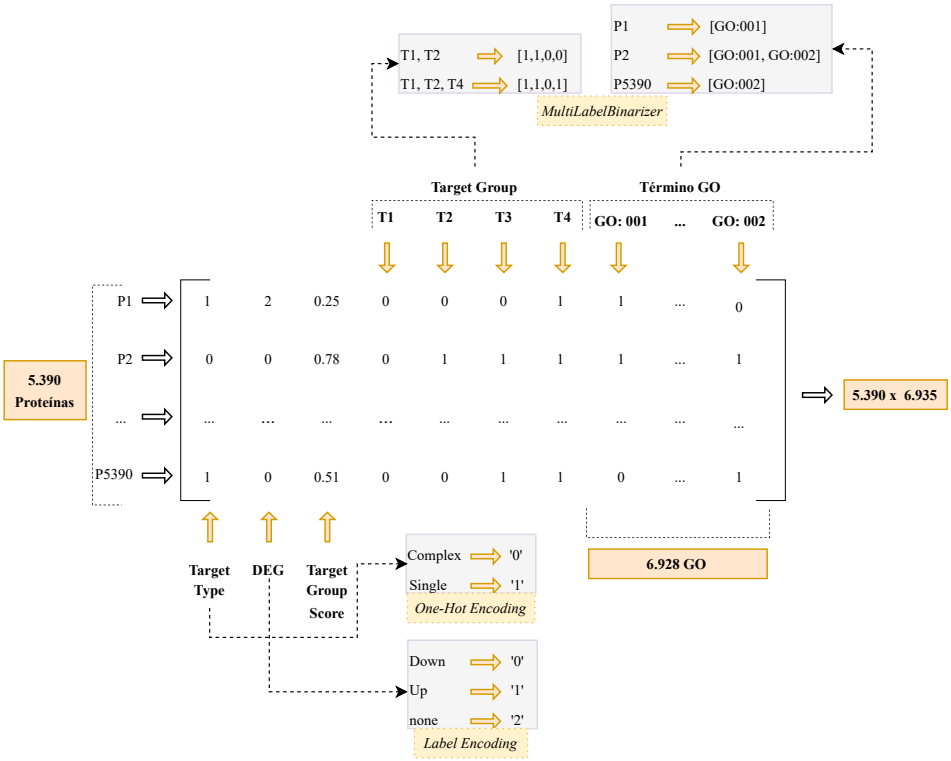


Figura 4.2: Ejemplo esquemático de la construcción de la matriz de atributos X. Cada proteína se representa como una fila y sus características como columnas.

Fuente: Elaboración propia.

Representación de aristas y atributos (*edge_attr*)

Las interacciones entre proteínas provienen de un archivo de red PPI, donde cada fila representa una interacción con un valor de `Interaction_Score`. Durante la construcción del grafo se consideran los siguientes aspectos:

- **Bidireccionalidad:** Para capturar la simetría biológica de las interacciones, cada arista se duplica en ambas direcciones.
- **Atributo de arista:** El valor de Interaction_score se utiliza como atributo de la arista, codificado en un tensor unidimensional (`torch.float`) mediante `.unsqueeze(1)`.

  Durante el entrenamiento del modelo GNN, este atributo es incorporado en el mecanismo de atención de las capas `GATConv` mediante el parámetro `edge_attr`. De esta manera, la atención que un nodo presta a sus vecinos no solo depende de sus propias representaciones, sino también del valor de interacción entre ellos. Este enfoque permite que el modelo aprenda a ponderar dinámicamente las conexiones en función de su relevancia biológica, integrando explícitamente el atributo de arista en el proceso de aprendizaje.

Con este procedimiento se da cumplimiento al **OE1**, pues se integró la información topológica de la red PPI con atributos funcionales provenientes de GO y metadatos de proteínas.

4.2. Arquitectura y Entrenamiento del Modelo GNN

Esta sección describe el diseño e implementación del modelo GNN, cuyo objetivo es generar embeddings para cada nodo (proteína) de la red.

Diseño de la arquitectura

El modelo se compone de dos módulos principales:

- **GNNEncoder:** Es el codificador de grafos basado en dos capas `GATConv` (Graph Attention Convolution), el cual transforma las características de entrada de cada nodo en un embedding de menor dimensión. Las capas `GATConv` implementan un mecanismo de atención que permite ponderar la contribución de los vecinos de un nodo en función tanto de sus características como del atributo de la arista que los conecta (ver script model_architecture.py en el repositorio de GitHub del proyecto).
- **LinkPredictor:** Es un MLP que toma como entrada los embeddings de dos nodos generados por el `GNNEncoder` (uno por cada nodo en un par), los concatena y produce una puntuación que representa la probabilidad de que exista un enlace entre ellos (implementado junto con el `GNNEncoder` en el mismo script).

.

El enfoque general de este estudio es no supervisado, ya que no se dispone de etiquetas explícitas que indiquen a qué módulo funcional pertenece cada proteína. Sin embargo, para entrenar la GNN se adopta una estrategia de **aprendizaje auto-supervisado**, basada en el LinkPredictor dentro de la red PPI, la idea central es utilizarlo como tarea auxiliar que permita al modelo aprender embeddings de los nodos que sean informativos, sin la necesidad de etiquetas externas.

Configuración del entrenamiento

Antes del entrenamiento, los enlaces del grafo se dividieron en tres subconjuntos utilizando `RandomLinkSplit`, donde se destinaron el **80 % para entrenamiento, 10 % para validación y 10 % para prueba**.

Cada ciclo de entrenamiento (epoch) consistió en los siguientes pasos:

1. **Fase de entrenamiento:** El `GNNEncoder` generó los embeddings de los nodos. Luego, el `LinkPredictor` predijo la probabilidad de enlace entre pares de nodos (positivos y negativos). Se calculó la pérdida con la función `BCEWithLogitsLoss`.

2. **Fase de evaluación:** Se evaluó el modelo sobre los conjuntos de validación y prueba. Los pesos no se actualizaron en esta fase.

3. **Actualización de parámetros:** Se utilizó el optimizador `Adam` para ajustar los pesos de ambas redes (GNN y predictor) de forma conjunta.

Además, se implemmenta un proceso de busqueda de hiperparámetros con la librería Optuna, empleando un TPE sampler y la técnica de pruning. Donde el espacio de búsqueda incluye variaciones en número de capas ocultas, cabezas de atención, tasa de aprendizaje y función de activación (ver script training_evaluation.py y script main_hyperparameter_tuning.py en el repositorio de GitHub del proyecto).

De esta forma se cumple el **OE2**, al implementar un modelo de deep learning (GNN) que captura relaciones topológicas y funcionales en la red, asociando proteínas con funciones biológicas relevantes en el contexto de la enfermedad de Alzheimer.

4.3. Detección de Módulos Funcionales

Una vez entrenado el modelo de GNN, se obtuvienen embeddings para cada nodo de la red PPI, generando una matriz de menor dimensión. Cada proteína queda representada por un vector en un espacio latente de menor dimensiones al original, bajo el supuesto de que proteínas con funciones similares se ubican próximas entre sí en dicho espacio.

Con base en este supuesto, la detección de módulos funcionales se definió como un problema de clustering sobre los embeddings. Para asegurar una adecuada preparación de los datos, los embeddings fueron normalizados mediante la norma L2, y posteriormente proyectados a un espacio bidimensional con UMAP, lo que permitió contar con una visualización exploratoria de los patrones de agrupamiento.

Para la identificación de módulos se exploraron tres enfoques de clustering: K-Means, DBSCAN y HDBSCAN. El desempeño de cada configuración se evaluó mediante métricas internas de validación, incluyendo el *Silhouette Score*, el índice de Davies–Bouldin y el índice de Calinski–Harabasz. Además, se consideraron como indicadores complementarios el número de clusters válidos detectados y la proporción de nodos asignados a ruido, con el fin de guiar la selección del algoritmo más adecuado para este contexto biológico [35] (ver carpeta clustering en el repositorio de GitHub del proyecto).

4.4. Análisis y Caracterización de Módulos Funcionales

La etapa final del pipeline se centra en la interpretación biológica de los módulos identificados, con el objetivo de asignarles una función representativa y caracterizar sus propiedades diferenciales. Este análisis busca traducir las agrupaciones detectadas a conocimiento funcional relevante mediante términos GO.

Para cada módulo obtenido a partir del clustering de embeddings, se realizó un análisis de enriquecimiento funcional que permite identificar las funciones biológicas predominantes de cada grupo. Este procedimiento consiste en:

1. **Recopilación de datos:** Se extraen las proteínas de cada módulo y sus respectivos términos GO.

2. **Cálculo del enriquecimiento:** Se evalua la sobrerrepresentación de términos GO mediante una prueba hipergeométrica, estimando la probabilidad de observar al menos k ocurrencias de un término dado bajo un modelo nulo de distribución aleatoria (esta prueba se implementa usando la función `scipy.stats.hypergeom.sf`.). Este análisis se complementa con el cálculo de un z-score que mide la magnitud del efecto.

3. **Corrección por comparaciones múltiples:** Se aplica el método de Benjamini-Hochberg (FDR) para ajustar los P-values, reduciendo la probabilidad de falsos positivos al considerar múltiples términos por módulo (esta corrección se implementa usando la función `statsmodels.stats.multitest.multipletests` con el método `"fdr_bh"`.).

4. **Prioritización de términos:** Para los términos con P-value corregido significativo y z-score positivo, se calcula un **combined score**, integrando significancia y magnitud. El término con mayor combined score es asignado como representativo del módulo.

Este análisis permite interpretar funcionalmente los módulos detectados y evaluar la coherencia biológica entre la topología del embedding y las funciones compartidas de las proteínas agrupadas.

De este modo, se da cumplimiento al **OE3**, al validar cuantitativamente la coherencia de los módulos detectados mediante métricas de clustering y análisis funcional (ver script analyze_embeddings.py y script module_analysis.py en el repositorio de GitHub del proyecto).

Capítulo 5. Resultados y Discusión

5.1. Resultados de optimización de hiperparámetros para la GNN

La búsqueda de hiperparámetros consistió en 20 Trials con la librería `Optuna` y permitió identificar combinaciones que maximizaban el rendimiento del modelo GNN.

El espacio de búsqueda se definió de forma exploratoria, considerando variaciones en el número de canales ocultos {64, 128, 256}, canales de salida {32, 64, 128}, num_heads {2, 4, 8}, learning_rate (10^{-4}–10^{-2}), predictor_hidden_channels {32, 64, 128}, dropout_rate (0.0–0.5), activation_function (`ReLU` o `tanh`) y epochs {50, 100, 150, 200}. Esta selección busca capturar una variedad de combinaciones posibles sin restringir el modelo a configuraciones preestablecidas, lo cual es especialmente relevante en contextos exploratorios como este, donde no existen recomendaciones universales para arquitecturas GNN en el dominio específico abordado.

Para guiar la búsqueda se utilizó el algoritmo TPE sampler. Adicionalmente, se empleó la técnica de **pruning** mediante el `MedianPruner`, la cual permite detener trials de forma anticipada si su rendimiento intermedio está por debajo de la mediana de los resultados anteriores. Esta técnica evita desperdiciar recursos computacionales en configuraciones poco prometedoras y acelera la convergencia del estudio.

La Figura 5.1 muestra la importancia relativa de cada hiperparámetro en el rendimiento del modelo.

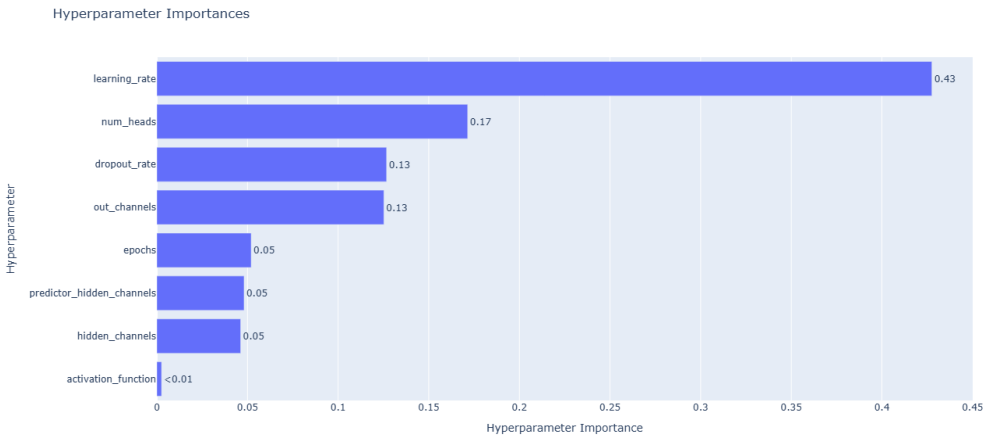


Figura 5.1: Importancia relativa de los hiperparámetros en la optimización del modelo GNN, calculada a partir de la métrica objetivo (AUC en validación). Se observa que la **tasa de aprendizaje** (*learning rate*) es el factor más determinante (0.43), lo que evidencia la sensibilidad del modelo a este parámetro de entrenamiento. En segundo lugar, el **número de cabezas de atención** (*num_heads*) y la **tasa de abandono** (*dropout rate*) también muestran un impacto relevante (> 0.1), indicando su influencia directa en la capacidad del modelo para equilibrar complejidad y regularización. El resto de los hiperparámetros, incluidos *out_channels*, número de épocas y dimensionalidad de capas ocultas, presentan importancias menores (≤ 0.05), sugiriendo un efecto limitado en el rendimiento global.

La evolución del valor objetivo a lo largo de los 20 *trials* se muestra en la Figura 5.2. Se observa una mejora progresiva durante las primeras iteraciones.

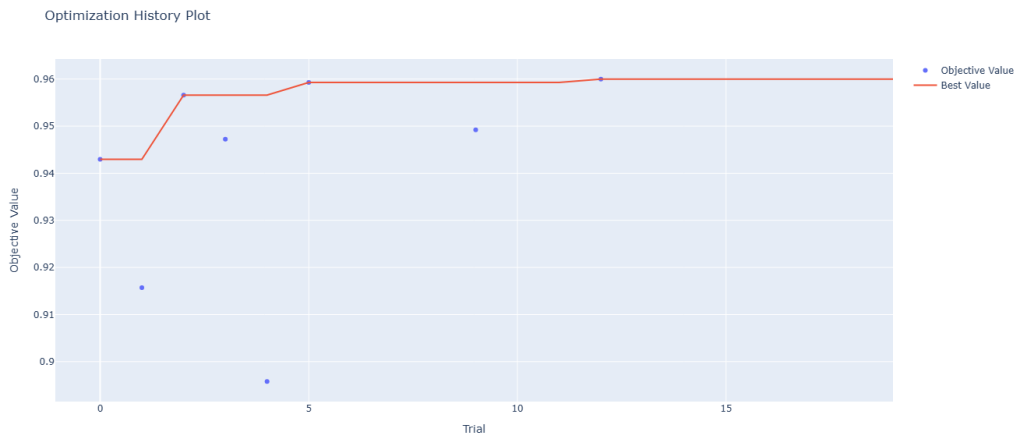


Figura 5.2: Evolución del valor objetivo (AUC en validación) a lo largo de los ensayos de optimización de hiperparámetros. Se aprecia una mejora rápida en las primeras iteraciones, alcanzando valores superiores a 0.95 desde el *trial* 3, lo que indica que configuraciones prometedoras fueron identificadas tempranamente. Posteriormente, el rendimiento se mantiene estable en torno al mejor valor registrado ($\sim$0.96), lo que sugiere una convergencia adecuada del proceso de búsqueda y una baja variabilidad en las configuraciones óptimas encontradas.

La Figura 5.3 permite visualizar, mediante coordenadas paralelas, los patrones globales de

las combinaciones de hiperparámetros exploradas y su relación con el rendimiento del modelo.

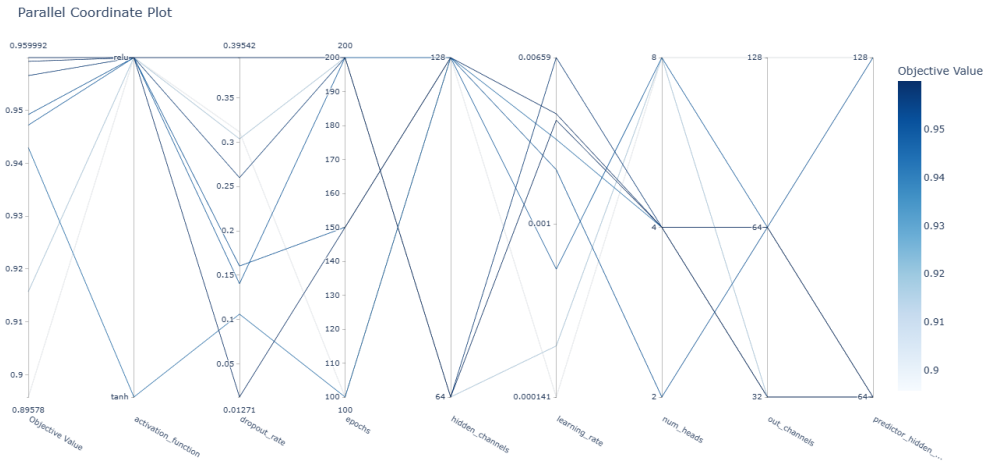


Figura 5.3: Gráfico de coordenadas paralelas que resume las combinaciones de hiperparámetros exploradas, coloreadas según el valor objetivo alcanzado (AUC en validación). Las configuraciones más exitosas (líneas más oscuras) muestran patrones comunes: uso consistente de la función de activación `ReLU`, tasas de aprendizaje moderadas en el rango de 10^{-3}, y un número de cabezas de atención intermedio (4–8). En contraste, combinaciones con tasas de aprendizaje extremas o menor número de cabezas tienden a presentar rendimientos inferiores, lo que resalta la importancia de un ajuste equilibrado de estos parámetros.

El resumen de los *trials* ejecutados, junto con sus configuraciones y puntajes finales, se presenta en la Tabla 5.1. Entre los 20 ensayos realizados, se identificaron tres configuraciones con puntajes sobresalientes (trials 2, 5 y 12), las cuales fueron evaluada en sus mejores épocas para validar su rendimiento. Los resultados detallados por conjunto de datos (entrenamiento, validación y prueba) se muestran en la Tabla 5.2.

Trial	Hidden Channels	Out Channels	Num Heads	Learning Rate	Predictor Hidden Ch.	Dropout Rate	Activation Function	Epochs	State	Score	Time
0	128	32	4	2.61E-03	64	0.106	tanh	100	COMPLETE	0.94297	15m 46s
1	64	32	8	2.51E-04	64	0.304	relu	200	COMPLETE	0.91572	36m 4s
2	64	64	4	6.59E-03	64	0.260	relu	200	COMPLETE	**0.95659**	19m 55s
3	128	64	8	5.99E-04	64	0.140	relu	200	COMPLETE	0.94722	159m 3s
4	128	128	8	1.41E-04	128	0.312	relu	100	COMPLETE	0.89578	72m 9s
5	128	32	4	3.48E-03	64	0.0127	relu	150	COMPLETE	**0.95928**	35m 39s
6	256	128	8	2.10E-04	32	0.436	relu	200	PRUNED	0.74461	174m 46s
7	256	32	8	1.03E-04	32	0.0599	tanh	100	PRUNED	0.73315	75m 12s
8	128	32	2	1.19E-04	32	0.139	relu	50	PRUNED	0.75569	15m 52s
9	128	64	2	1.85E-03	128	0.160	relu	150	COMPLETE	0.94922	53m 20s
10	64	32	4	7.72E-03	64	0.0194	tanh	150	PRUNED	0.74514	20m 57s
11	64	64	4	7.79E-03	64	0.217	relu	150	PRUNED	0.74715	22m 16s
12	64	64	4	3.24E-03	64	0.395	relu	200	COMPLETE	**0.95999**	51m 8s
13	64	64	4	2.65E-03	64	0.474	relu	150	PRUNED	0.83830	24m 42s
14	256	128	4	8.88E-04	64	0.383	tanh	200	PRUNED	0.94332	118m 55s
15	128	32	4	3.80E-03	128	0.379	relu	150	PRUNED	0.83841	31m 33s
16	64	64	4	1.36E-03	64	0.212	relu	50	PRUNED	0.77252	7m 2s
17	64	32	2	5.08E-03	64	0.363	relu	200	PRUNED	0.74561	38m 22s
18	128	64	4	7.05E-04	128	0.463	tanh	150	PRUNED	0.75217	39m 29s
19	256	128	4	3.64E-03	32	0.0166	relu	100	PRUNED	0.69732	27m 7s

Tabla 5.1: Resumen de configuraciones de hiperparámetros y score por trial. Se destacan los mejores valores.

Fuente: Elaboración propia.

Conjunto	Métrica	Trial 12 (Epoch 200)	Trial 5 (Epoch 150)	Trial 2 (Epoch 199)
Train	Loss	0.29231	0.281204	**0.267288**
	AUC	0.949887	0.952602	**0.958134**
	Accuracy	0.879862	0.873731	**0.882378**
	F1-Score	0.879712	0.866393	**0.875996**
	Precision	0.88081	0.919852	**0.926252**
	Recall	**0.878617**	0.818805	0.830913
Validation	Loss	**0.282134**	0.289436	0.267288
	AUC	0.951798	0.949374	**0.958931**
	Accuracy	0.882345	0.875918	**0.894805**
	F1-Score	0.879661	0.870541	**0.894831**
	Precision	0.900197	**0.909978**	0.897624
	Recall	0.860042	0.834379	**0.89126**
Test	Loss	0.278446	0.285811	**0.262205**
	AUC	0.953161	0.950519	**0.959642**
	Accuracy	0.883322	0.878115	**0.894491**
	F1-Score	0.880802	0.872923	**0.894137**
	Precision	**0.900245**	0.911758	0.89715
	Recall	0.86218	0.837262	**0.891144**

Tabla 5.2: Comparación de métricas por conjunto de datos para los tres mejores trials en su mejor época. Se destacan los mejores valores.

Fuente: Elaboración propia.

A partir del análisis comparativo, se concluye que el **trial 2**, evaluado en la **época 199**, ofrece el mejor equilibrio entre las métricas, destacando por su AUC de 0.9596 en el conjunto

de prueba, precisión y *F1-score* superiores a 0.89 en validación y test, y una baja pérdida en todos los conjuntos. Esta configuración fue seleccionada como la óptima.

Además de las métricas resumen empleadas para seleccionar el mejor modelo, se generaron visualizaciones detalladas que muestran la evolución de las métricas durante el entrenamiento del mejor *trial*. Dichos resultados se presentan en el Anexo A (Figura A.1), donde se ilustran las métricas de *Precision*, *Recall*, *F1-Score*, *Accuracy*, *AUC* y *Loss* calculadas por época en los conjuntos de entrenamiento y validación. En conjunto, estas curvas correspondientes al modelo seleccionado (Trial 2, Epoch 199) evidencian una mejora progresiva y consistente en las primeras etapas del entrenamiento, seguida de una estabilización sin indicios claros de sobreajuste.

La Figura 5.4 muestra la arquitectura final del modelo, compuesta por dos capas GATConv, con 4 cabezas de atención en la primera y salida de 64 canales en la segunda, seguidas por un Link Predictor de dos capas lineales. El resultado es un embedding de dimensión (5390×64), en el cual cada una de las 5,390 proteínas está representada como un vector en un espacio latente de 64 dimensiones.

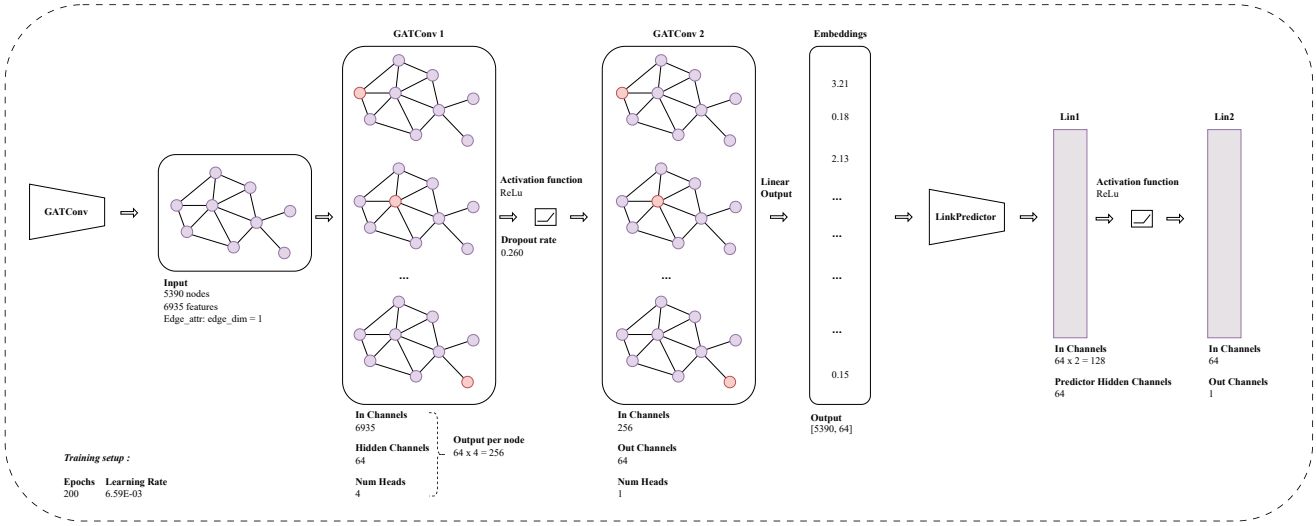


Figura 5.4: Arquitectura final del modelo GAT seleccionado en el **trial 2**. El encoder está compuesto por dos capas *GATConv*, seguidas por un módulo *Link Predictor* que evalúa pares de nodos a partir de sus embeddings. La salida del encoder es una matriz de dimensión $(5{,}390 \times 64)$, donde cada nodo/proteína queda representado en un espacio latente de 64 dimensiones.

Fuente: Elaboración propia.

5.2. Resultados de optimización de hiperparámetros para la detección de módulos funcionales

Una vez entrenado el modelo de GNN y generados los embeddings, se procedió a su análisis en términos de clustering. La Figura 5.5 muestra la proyección bidimensional obtenida mediante UMAP, donde cada punto representa una proteína. La distribución evidencia la presencia de regiones de mayor densidad, lo que sugiere la existencia de módulos funcionales.

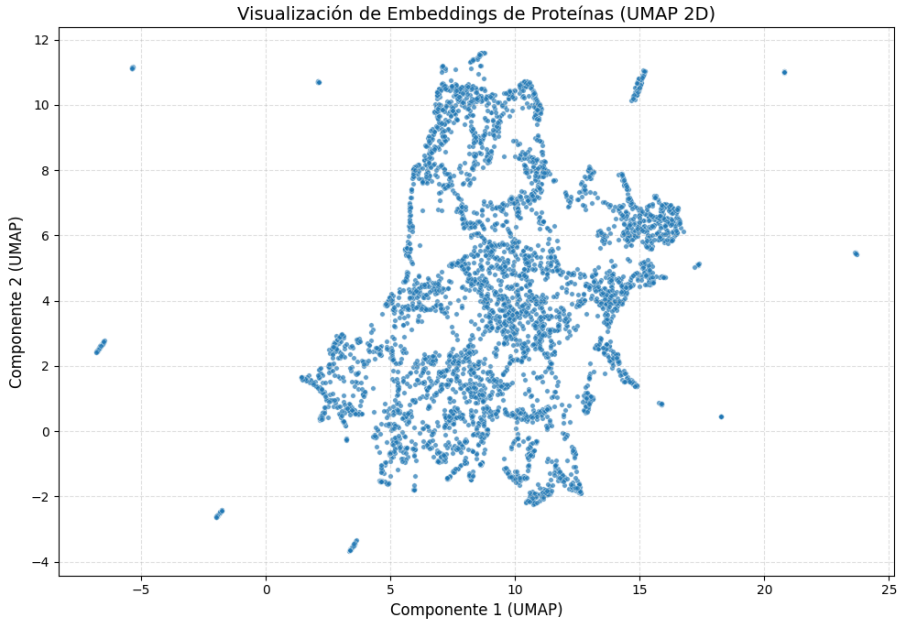


Figura 5.5: Visualización 2D de los embeddings generados por la GNN mediante reducción de dimensionalidad con UMAP. Cada punto representa una proteína.

Fuente: Elaboración propia.

Para definir con que algoritmo de clustering detectar módulos funcionales, se evaluaron sus hiperparámetros.

K-Means El primer enfoque considerado fue el algoritmo **K-Means**, el cual requiere especificar previamente el número de clusters K. Para estimar un valor óptimo de K, se evaluó en rango de $K \in [2, 50]$. Los resultados de esta evaluación se encuentran en el Anexo B y se resumen en la Tabla B.1 y Figura B.1.

El mejor compromiso entre métricas fue alcanzado en $K = 5$. La Figura 5.6 muestra la distribución de los clusters obtenidos proyectados en el espacio UMAP.

Figura 5.6: Proyección 2D con UMAP de los embeddings, coloreados según los clusters asignados por K-Means.

Fuente: Elaboración propia.

DBSCAN Luego, se probó el algoritmo **DBSCAN**, ya que permite identificar agrupaciones de forma arbitraria y manejar explícitamente los puntos considerados como ruido.

Se realizó una búsqueda exhaustiva de hiperparámetros explorando los valores del radio ε (eps) y el número mínimo de puntos requeridos para definir una región densa, `min_samples`. El parámetro ε fue evaluado en el rango $[0.001,\ 1.5]$, con mayor densidad de valores entre 0.001 y 0.8 para capturar comportamientos más sensibles, y espaciados más amplios en el rango superior. Por su parte, `min_samples` fue evaluado entre 2 y 200, empleando incrementos más finos en los valores bajos para observar con mayor detalle la transición entre agrupaciones densas y dispersas.

Los resultados de esta exploración se presentan en el Anexo B, en la Figura B.2, donde se muestran los valores obtenidos para métricas mencionadas anteriormente, el porcentaje de puntos etiquetados como ruido y el número total de clusters válidos para cada combinación de parámetros. Además, la Tabla B.2 resume las mejores combinaciones de hiperparámetros según cada métrica.

La mejor combinación fue alcanzada por $\varepsilon = 0.0064$ y $min\_samples = 90$. La Figura 5.7 muestra la distribución de los clusters obtenidos proyectados en el espacio UMAP.

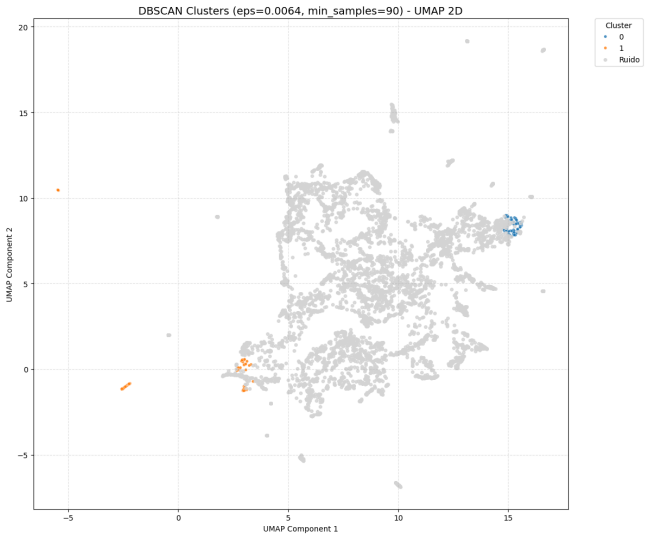


Figura 5.7: Proyección 2D con UMAP de los embeddings, coloreados según los clusters asignados por DBSCAN.

Fuente: Elaboración propia.

Si bien algunos resultados presentan altos valores en las métricas de agrupamiento, la mayoría de las combinaciones generaron un elevado porcentaje de puntos etiquetados como ruido (superior al 94 %), lo que motivó la exploración de alternativas como HDBSCAN.

HDBSCAN Con respecto al algoritmo **HDBSCAN**, el cual extiende a DBSCAN incorporando una estructura jerárquica de densidad, se realizó una búsqueda exhaustiva de combinaciones de hiperparámetros evaluando los siguientes valores de `min_cluster_size`: $\{2, 4, 6, \ldots, 200\}$, *min_samples*: $\{1, 2, 3, \ldots, 100\}$, `cluster_selection_epsilon`: 0.0 (valor por defecto) y `alpha`: $\{0.5,\ 1.0,\ 1.5,\ 2.0\}$. Esto resultó en un total de $20 \times 20 \times 1 \times 4 = 1{,}600$ combinaciones de parámetros evaluadas.

Los resultados detallados se encuentran en el Anexo B, en la Figura B.3, y las mejores combinaciones por métrica se resumen en la Tabla B.3.

Si bien la configuración con `min_cluster_size = 40`, `min_samples = 40` y `alpha = 1.5` obtuvo el mayor *Silhouette Score*, se optó por utilizar la configuración con `min_cluster_size = 20`, `min_samples = 20` y `alpha = 2.0`, la cual entregó 14 clusters válidos (versus sólo 8 en la configuración anterior), un *Silhouette Score* cercano al anterior, y el mejor valor del índice de Davies-Bouldin. La Figura 5.8 muestra los clusters obtenidos en el espacio UMAP.

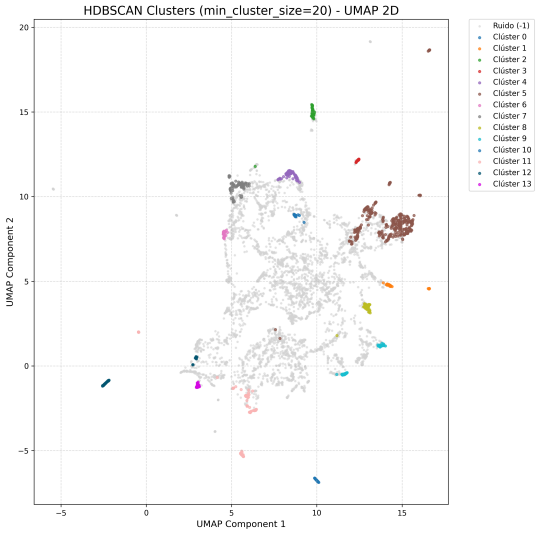


Figura 5.8: Proyección 2D con UMAP de los embeddings, coloreados según los clusters asignados por HDBSCAN.

Fuente: Elaboración propia.

A pesar de que un 79.91 % de las proteínas fueron etiquetadas como ruido, el modelo logró identificar agrupaciones consistentes en distintas regiones densas de la red.

A modo de comparación global entre algoritmos, la Tabla 5.3 resume las mejores configuraciones encontradas para K-Means, DBSCAN y HDBSCAN, junto con sus métricas de evaluación y porcentaje de puntos etiquetados como ruido.

Si bien DBSCAN alcanzó los valores más altos en todas las métricas internas , dicha configuración agrupó sólo al 5 % de las proteínas (2 clusters válidos) y clasificó al 94.92 % como ruido. Esta situación lo vuelve poco práctico para el análisis funcional posterior, ya que se perdería la información de la gran mayoría de los nodos.

Por su parte, K-Means no presenta puntos etiquetados como ruido y logró un número fijo de clusters (K = 5), sin embargo, sus métricas internas fueron sustancialmente más bajas, especialmente en Silhouette Score (0.2806), lo cual indica una peor separación entre grupos.

En cambio, HDBSCAN, con la configuración de `min_cluster_size = 20`, `min_samples = 20` y `alpha = 2.0`, logró un equilibrio razonable entre métricas de agrupamiento (Silhouette Score = 0.6605) y porcentaje de ruido (79.91 %), detectando 14 clusters válidos.

Por estos motivos, se seleccionó HDBSCAN como el algoritmo de agrupamiento final para la detección de módulos funcionales, al ofrecer una mejor relación entre calidad de clusters y cobertura sobre los datos.

Algoritmo de clustering	ConFiguración	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Index	Clústers válidos	Porcentaje de ruido
KMeans	K = 5	0.2806	1.2480	1676.70	5	N/A
DBSCAN	eps = 0.0064 min_samples = 90	**0.9006**	**0.1465**	**10792.87**	2	94.92 %
HDBSCAN	min_cluster_size = 20 min_samples = 20 cluster_selection_epsilon = 0.0 alpha = 2.0	0.6605	0.3897	1370.82	**14**	**79.91 %**

Tabla 5.3: Comparación de algoritmos de clustering evaluados según tres métricas internas. Se muestran las configuraciones que produjeron los mejores resultados dentro de cada algoritmo.

Fuente: Elaboración propia.

5.3. Resultado de análisis funcional

Se presentan a continuación las distribuciones de las métricas estadísticas obtenidas en el análisis de enriquecimiento funcional: z-score, P-value corregido (FDR) y combined score. Estas visualizaciones permiten evaluar globalmente la robustez del enriquecimiento de términos GO asignados a los módulos y validar la coherencia estadística de los resultados.

En este análisis, la hipótesis nula establece que los términos GO se distribuyen de manera aleatoria entre todas las proteínas de la red.

La Figura 5.9 muestra que todos los valores de z-score se encuentran claramente por encima del umbral de referencia ($z > 2$, línea roja vertical), alcanzando incluso valores superiores a 70. Este umbral se fundamenta en que, en una distribución normal estándar, alrededor del 95.45 % de los valores se concentran en el rango comprendido entre -2σ y $+2\sigma$, mientras que el 4.55 % restante se ubica en las colas de la distribución (aproximadamente un 2.28 % en cada extremo). De este modo, un valor de $z > 2$ corresponde a una desviación que tendría una probabilidad inferior al 5 % de ocurrir bajo la hipótesis nula de distribución aleatoria, lo cual se traduce en un nivel de significancia estadística convencional ($p < 0{,}05$). Los elevados z-scores observados constituyen, por tanto, una evidencia robusta de que los términos GO analizados están enriquecidos en los módulos.

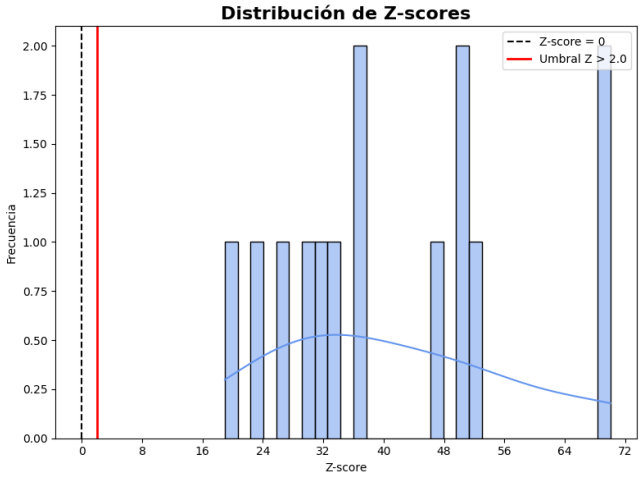


Figura 5.9: Distribución de z-scores para los términos GO evaluados. La línea roja vertical indica el umbral de referencia en $z = 2$

Fuente: Elaboración propia.

La Figura 5.10 presenta la distribución de los P-values ajustados por FDR. Todos los términos evaluados muestran valores extremadamente bajos, en su mayoría en el rango de 10^{-18} o menores. El umbral de significancia convencional ($\alpha = 0.05$, línea roja discontinua) se encuentra muy por encima de los valores observados, lo que confirma la solidez estadística del enriquecimiento, incluso tras corrección por múltiples comparaciones.

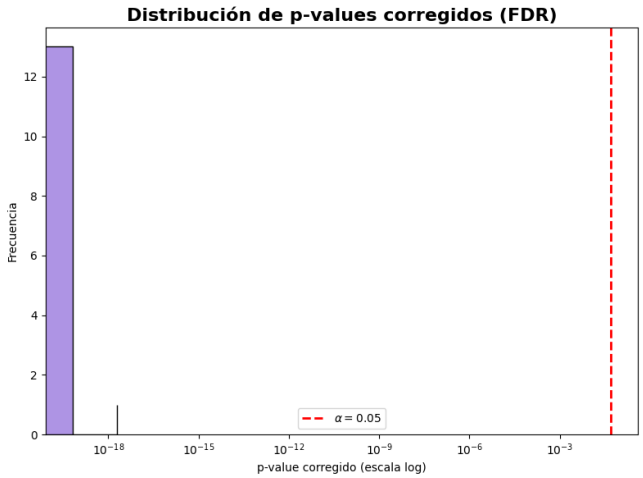


Figura 5.10: Distribución de P-values corregidos mediante FDR. La línea roja horizontal corresponde al umbral de significancia $\alpha = 0.05$.

Fuente: Elaboración propia.

Finalmente, la Figura 5.11 presenta la distribución del *combined score*, métrica que resulta de multiplicar el z-score por el $-\log_{10}$ del p-value corregido. Este valor combina simultáneamente la magnitud del efecto (z-score) y la significancia estadística (p-value), proporcionando así una medida integrada de la relevancia biológica de cada término. La distribución se observa sesgada hacia la derecha, con la mayoría de los términos concentrados en valores elevados, y algunos casos destacados que superan los 6,000, lo que refleja un enriquecimiento particularmente robusto.

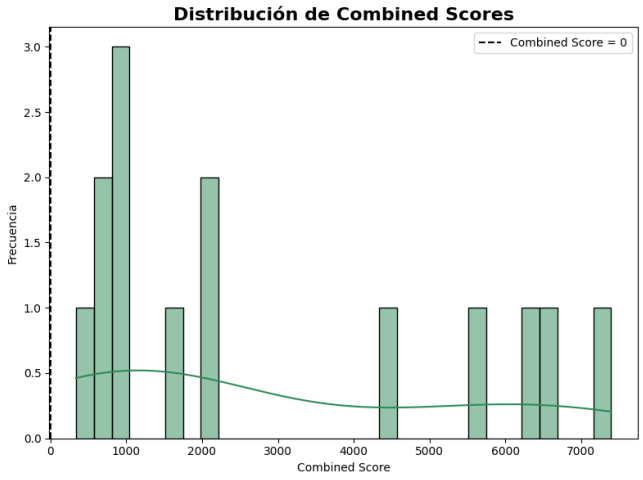


Figura 5.11: Distribución de combined scores para los términos GO evaluados.

Fuente: Elaboración propia.

En conjunto, estos resultados respaldan la validez estadística del enriquecimiento funcional, permiten rechazar la hipótesis nula de distribución aleatoria y constituyen una base sólida para la posterior interpretación biológica de los módulos identificados.

5.4. Resultado final del pipeline

Finalmente, como resultado del pipeline desarrollado, se obtuvo una red de entrada compuesta por **5,390** proteínas interconectadas a través de **430,206** interacciones únicas (**860,412** conexiones considerando la bidireccionalidad). La integración de **6,928** términos GO provenientes de las tres ontologías principales permitió construir una matriz de características de dimensión **(5,390 , 64)**. De los nodos, **5,161** (95.8 %) se encontraban anotados con al menos un término GO, reflejando una cobertura funcional amplia y proporcionando una representación biológica robusta de la red (Figura 5.12).

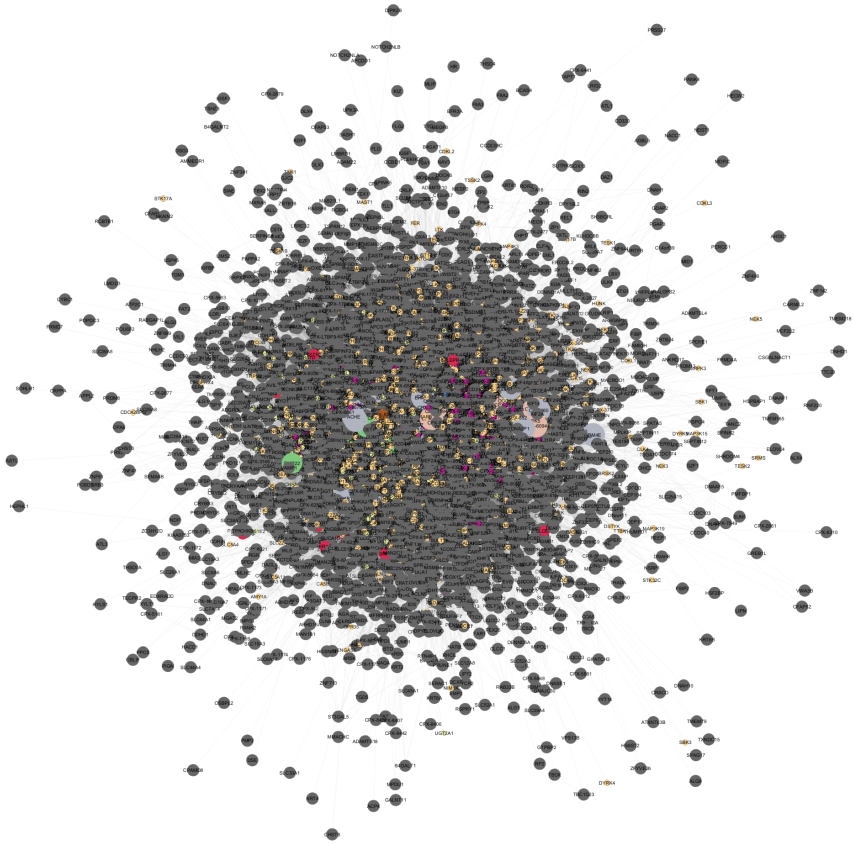


Figura 5.12: Visualización global de la red proteína–proteína (PPI) en Cytoscape. La representación evidencia la alta densidad de interacciones y la complejidad topológica de la red asociada a Alzheimer, que constituyó la base para el análisis de embeddings mediante GNN.

Fuente: Elaboración propia.

La Tabla 5.4 sintetiza los resultados de los 14 módulos funcionales detectados mediante HDBSCAN, indicando para cada uno su término GO representativo y el nivel de significancia estadística (p-value, z-score y *combined score*), además de las distribuciones de `DEG` y `Target_Group`. Por su parte, la Figura 5.13 muestra la visualización global de la red de módulos en Cytoscape, diseñada para facilitar la interpretación biológica de las agrupaciones obtenidas.

Módulo	GO term	Total	*P*-value	z-score	Combined Score	Distribución DEG	Distribución Target
0	GO:0043202 (Lysosomal Lumen)	23	$1{,}64 \times 10^{-25}$	31.48	780.32	{none: 100.00 %}	{T4: 91.30 %, T2: 8.70 %}
1	GO:1903027 (Regulation Of Opsonization)	40	$2{,}29 \times 10^{-23}$	37.68	853.04	{none: 100.00 %}	{T4: 92.50 %, T2: 7.50 %}
2	GO:0007601 (Visual Perception)	61	$1{,}94 \times 10^{-18}$	18.95	335.69	{none: 100.00 %}	{T4: 93.44 %, T2: 6.56 %}
3	GO:0007605 (Sensory Perception Of Sound)	31	$1{,}36 \times 10^{-24}$	26.98	643.95	{none: 96.77 %, Up: 3.23 %}	{T4: 96.77 %, T2: 3.23 %}
4	GO:0019585 (Glucuronate Metabolic Process)	66	$2{,}04 \times 10^{-28}$	34.12	944.82	{none: 98.48 %, Up: 1.52 %}	{T4: 16.67 %, T2: 83.33 %, T3: 46.97 %, T1:3.03 %}
5	GO:0098916 (Anterograde Trans-Synaptic Signaling)	366	$9{,}48 \times 10^{-124}$	36.15	**4446.78**	{none: 91.80 %, Down: 7.38 %, Up: 0.82 %}	{T2: 40.98 %, T3: 16.12 %, T4: 65.03 %, T1: 2.46 %}
6	GO:0004812 (aminoacyl-tRNA Ligase Activity)	28	$1{,}07 \times 10^{-41}$	51.20	2097.54	{none: 100.00 %}	{T4: 96.43 %, T2: 7.14 %}
7	GO:0005759 (Mitochondrial Matrix)	88	$1{,}86 \times 10^{-37}$	22.97	843.81	{none: 90.91 %, Up: 6.82 %, Down: 2.27 %}	{T4: 90.91 %, T2: 10.23 %, T3: 1.14 %}
8	GO:0062023 (Collagen-Containing Extracellular Matrix)	53	$1{,}79 \times 10^{-54}$	30.17	1621.38	{none: 100.00 %}	{T4: 100.00 %}
9	GO:0038196 (Type III Interferon-Mediated Signaling Pathway)	44	$5{,}15 \times 10^{-46}$	46.24	2094.19	{none: 100 %}	{T4: 59.09 %, T2: 40.91 %, T3: 2.27 %}
10	GO:0071565 (nBAF Complex)	41	$2{,}52 \times 10^{-94}$	70.04	**6556.19**	{none: 100.00 %}	{T4: 97.56 %, T2: 9.76 %}
11	GO:0035102 (PRC1 Complex)	116	$8{,}02 \times 10^{-112}$	50.79	**5642.32**	{none: 95.69 %, Down: 4.31 %}	{T4: 93.97 %, T2: 11.21 %, T3: 4.31 %, T1: 1.72 %}
12	GO:0045116 (Protein Neddylation)	87	$8{,}50 \times 10^{-142}$	52.42	**7394.65**	{none: 100.00 %}	{T2: 100.00 %, T4: 10.34 %}
13	GO:0031145 (Anaphase-Promoting Complex-Dependent Catabolic Process)	39	$9{,}89 \times 10^{-93}$	69.20	**6366.72**	{none: 100.00 %}	{T2: 100.00 %, T4: 100.00 %}

Tabla 5.4: Módulos detectados por HDBSCAN y su enriquecimiento GO, obtenido del análisis de enriquecimiento del capítulo 4.4. Se destacan los mejores valores.

Fuente: Elaboración propia.

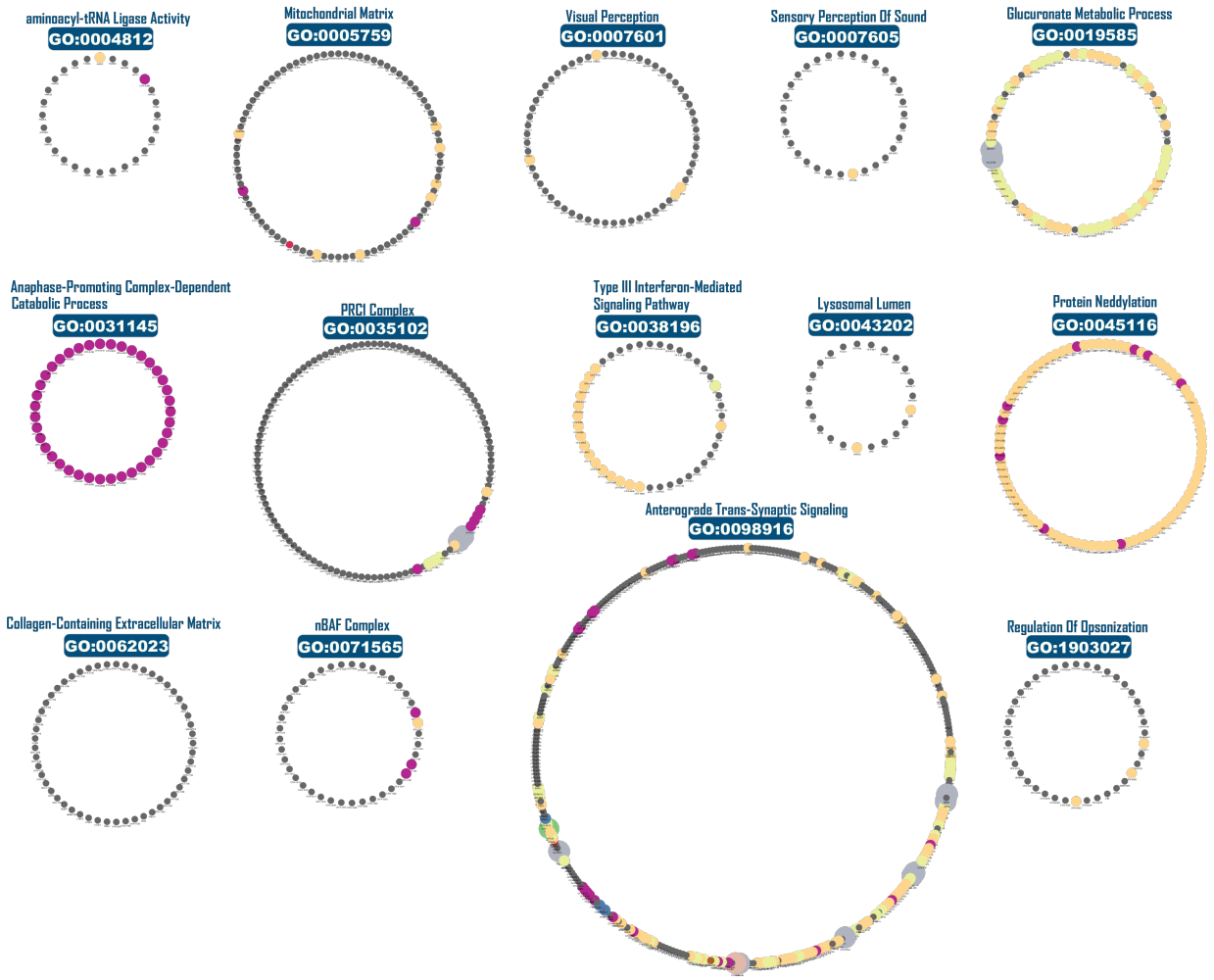


Figura 5.13: Visualización de los módulos en Cytoscape. Cada nodo corresponde a una proteína y está asociado a su término GO representativo. La asignación de colores se realizó según la combinación de `Target_Group`, siguiendo la leyenda mostrada en la Tabla 5.5. El tamaño de los nodos refleja el nivel de validación: los pertenecientes a T1 son más grandes, mientras que aquellos que incluyen T4 se representan con tamaños más pequeños, indicando menor validación biológica.

Fuente: Elaboración propia.

Target Group	Score normalizado	Color
T4	0.250000	
T3	0.500000	
T3, T4	0.516667	
T2	0.750000	
T2, T3	0.763889	
T2, T4	0.766667	
T2, T3, T4	0.787500	
T1, T2	0.991667	
T1, T2, T3	0.994375	
T1, T2, T3, T4	1.000000	

Tabla 5.5: Leyenda de colores para la variable *Target_group*, ordenada de menor a mayor según el *Score normalizado*.

Fuente: Elaboración propia.

La Figura 5.14 muestra la proyección bidimensional de los embeddings mediante UMAP, considerando únicamente los nodos agrupados por HDBSCAN (excluyendo el ruido). De los 5,390 nodos, 1,083 (20.1 %) fueron asignados a un clúster válido y 4,307 (79.9 %) fueron considerados como ruido. En el contexto de HDBSCAN, los nodos son clasificados como *ruido* cuando no alcanzan un nivel mínimo de densidad local que permita asignarlos de manera confiable a un clúster.

Con respecto a que solo un 20.1 % de las proteínas fueron agrupados en módulos, este comportamiento es consistente con la naturaleza de los datos biológicos, ya que solo una fracción de la red suele estar bien caracterizada funcionalmente, mientras que el resto corresponde a proteínas con funciones poco definidas o con información insuficiente para ser agrupadas de manera robusta. En este sentido, la eliminación del ruido no se considera una pérdida de información, sino más bien un mecanismo de filtrado que prioriza aquellos módulos con mayor coherencia, un patrón descrito en el análisis de redes biológicas [35].

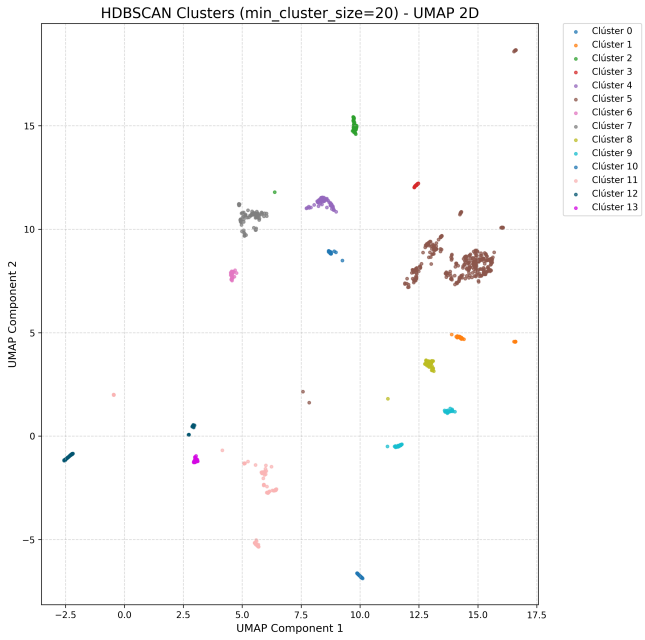


Figura 5.14: Proyección bidimensional mediante UMAP de los embeddings, coloreados según los clústeres asignados por HDBSCAN. El protocolo permitió identificar **14 clústeres** válidos a partir de 1,083 proteínas, excluyendo los nodos clasificados como ruido.

Fuente: Elaboración propia.

Para explorar las características biológicas por módulo, se analizaron las distribuciones de `DEG` y `Target_Group`. La Figura 5.15 muestra la proporción de proteínas diferencialmente expresadas en cada clúster. Si bien predomina la categoría *none*, algunos módulos (como los módulos 5, 7 y 11) exhiben una fracción apreciable de *Down*, potencialmente vinculada a procesos alterados en EA.

Es importante destacar que la expresión diferencial (DEG) no sesgó la conformación de los clústeres, sino que las proteínas diferencialmente expresadas se distribuyeron entre distintos módulos.

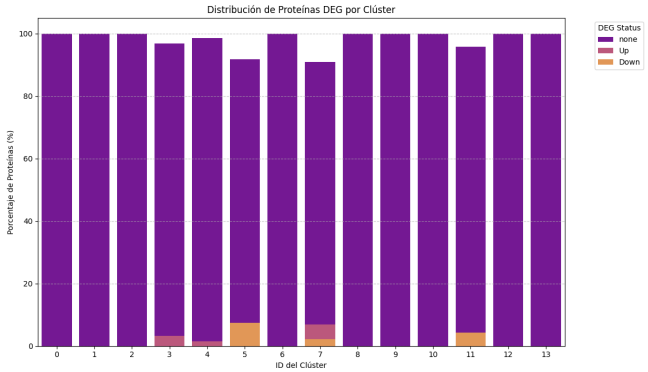


Figura 5.15: Distribución de proteínas diferencialmente expresadas (DEG) por clúster.

Fuente: Elaboración propia.

Respecto de los niveles de validación terapéutica (`Target_Group`), la Figura 5.16 muestra la proporción T1–T4 por clúster. La mayoría de los módulos concentran proteínas T4 (blancos poco explorados), aunque clústeres como 4, 5 y 11 presentan mayor diversidad e incluyen T1 y T2.

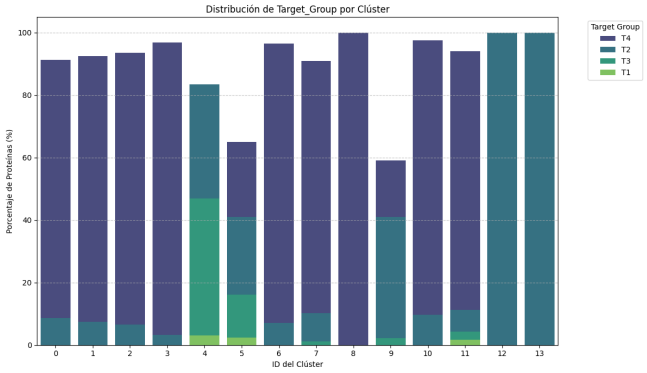


Figura 5.16: Distribución de grupos terapéuticos (Target Group) por clúster.

Fuente: Elaboración propia.

Para complementar el análisis, se calculó el índice de Jaccard entre pares de clústeres considerando los términos GO anotados. La Figura 5.17 muestra un mapa de calor donde, en general, se observan similitudes bajas (< 0.15), lo que sugiere que los módulos tienden a ser funcionalmente distintos. Sólo algunos pares (10-11 y 11-12) alcanzan valores mayores (Jaccard ≈ 0.23), posiblemente por compartir procesos biológicos generales.

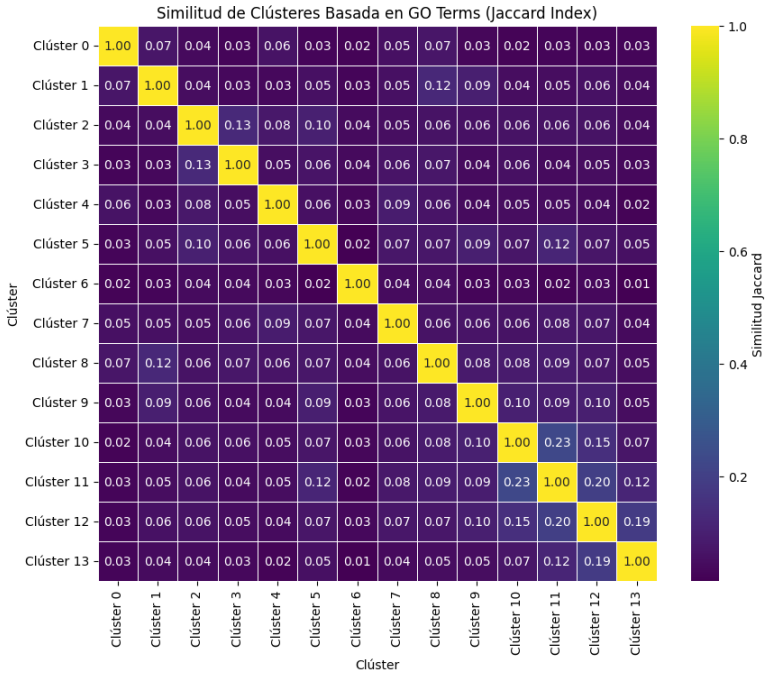


Figura 5.17: Similitud de clústeres basada en términos GO, calculada mediante índice de Jaccard.

Fuente: Elaboración propia.

Los resultados obtenidos evidencian que, a pesar de la complejidad de la red PPI original y de la alta dimensionalidad de las características de los nodos, el modelo GNN fue capaz de generar embeddings informativos.

5.4.1. Priorización de proteínas

Al restringir el análisis a las **1,083** proteínas asignadas a clústeres por HDBSCAN, el número de aristas se redujo a **50,669**. Sobre esta subred se definió, en conjunto con el laboratorio del profesor Ramírez, calcular métricas topológicas ponderadas por `interaction_Score` con el objetivo de caracterizar el rol de cada proteína dentro de su módulo y entre módulos, prorizando las proteínas T4.

La Tabla 5.6 muestra la distribución de combinaciones de *Target_group*, donde se observa predominio de T4 (blancos poco explorados), con 692 proteínas (63.9 %).

Target_group	**Frecuencia**	**Porcentaje (%)**
T4	692	63.90
T2	213	19.67
T2, T4	80	7.39
T2, T3	78	7.20
T1, T2, T3	11	1.02
T3	3	0.28
T2, T3, T4	3	0.28
T1, T2, T3, T4	1	0.09
T3, T4	1	0.09
T1, T2	1	0.09

Tabla 5.6: Distribución de *Target_group*.

Fuente: Elaboración propia.

Las métricas topológicas corresponden a: intracluster_connect, intercluster_connect y el Connectivity_Score. Las dos primeras corresponden a la conectividad interna y externa de cada módulo funcional, respectivamente, mientras que el `Connectivity_Score` constituye una métrica global provista por Ramírez Lab, únicamente calculada para los T4, definida como la relación entre las conexiones de un módulo con aquellos asociados a la EA y con los nodos vecinos de segunda generación. El análisis topológico de estas métricas, cuyas distribuciones completas se presentan en el Anexo C, permitió caracterizar el comportamiento global de la red y diferenciar roles proteicos.

Definimos, para cada proteína i con clúster $C(i)$, y pesos $w_{ij} =$ `interaction_Score`:

$$\texttt{intracluster\_connect}(i) = \sum_{j \in C(i)} w_{ij}, \qquad \texttt{intercluster\_connect}(i) = \sum_{j \notin C(i)} w_{ij}. \tag{5.1}$$

- `intracluster_connect`: suma ponderada de interacciones con vecinas del *mismo* clúster. Valores altos sugieren *hubs internos* (nodos centrales para la cohesión y función del módulo).

- `intercluster_connect`: suma ponderada de interacciones con vecinas de *otros* clústeres. Valores altos sugieren *puentes* o *conectores* entre módulos (posibles mediadores de comunicación/integación de procesos).

Como se detalla en el Anexo C, la Tabla C.1 resume los percentiles de las tres métricas y la Figura C.1 muestra sus distribuciones completas. En particular, la **conectividad intracluster** presentó una mediana de 24.92 enlaces, con un rango intercuartílico entre 15 y 60, y un subconjunto reducido de valores atípicos que supera ampliamente las 100 conexiones internas, identificados como *hubs intracluster*. La **conectividad intercluster**, en contraste, mostró valores muy bajos en la mayoría de las proteínas (mediana = 0.55), con 25 % de nodos sin conexiones externas (Q1 = 0) y 75 % con $\leq$ 4.10 enlaces, aunque unos pocos superan las 150 conexiones, cumpliendo el rol de *puentes críticos intermodulares*. Finalmente, el `Connectivity_Score` evidenció una distribución más compacta (mediana = 0.739) y rango intercuartílico = 0.64–0.86.

Con el fin de integrar estas perspectivas y priorizar de manera objetiva los blancos moleculares más relevantes, se construyó un **diagrama de Venn** definiendo tres conjuntos de proteínas a partir de un umbral en el percentil 75 % (Q3), lo que correspondió a un total de 173 proteínas con valores de conectividad superiores a la mayoría de la red. En este contexto, el término *alto* se utiliza explícitamente para referirse a aquellas proteínas cuyos valores se encuentran por sobre dicho percentil 75 %.

- A = T4 con `intracluster_connect` alto ($\geq$ percentil 75 %)

- B = T4 con `intercluster_connect` alto ($\geq$ percentil 75 %)

- C = T4 con `Connectivity_Score` alto ($\geq$ percentil 75 %)

La Figura 5.18 resume las intersecciones. Las proteínas ubicadas en la intersección $\mathbf{A} \cap \mathbf{B} \cap \mathbf{C}$ ($n = 51$) representan el subconjunto más estricto de candidatos, ya que cumplen simultáneamente con una alta conectividad intracluster, intercluster y un `Connectivity_Score` elevado. Dichas proteínas, detalladas en la Tabla 5.7, constituyen los posibles *blancos moleculares prioritarios* identificados en este trabajo.

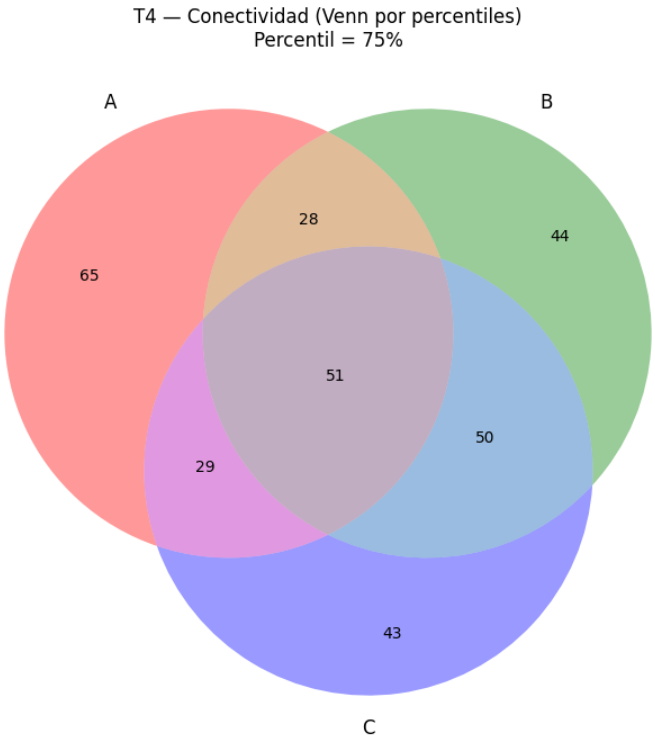


Figura 5.18: Diagrama de Venn de T4 con umbral en el percentil 75 % para `intracluster_connect` (A), `intercluster_connect` (B) y `Connectivity_Score` (C).

Fuente: Elaboración propia.

Sin embargo, al contrastar los resultados con el Ramírez Lab, se concluyó que la mayoría de las proteínas seleccionadas correspondían a proteínas del tipo complejos y, adicionalmente, se encontraban agrupadas en los mismos clústeres identificados en este trabajo, lo que genera un alto grado de solapamiento en las tres métricas y disminuye la capacidad discriminativa del enfoque interseccional. Además, los resultados se concentraron únicamente en los clústeres 5 y 11, lo que redujo aún más la representatividad del análisis al focalizarse en solo dos módulos. Por este motivo, se acordó no restringir la priorización a la coincidencia en las tres categorías, sino seleccionar directamente los valores más altos en cada métrica, independientemente de si convergen en el mismo subconjunto.

Protein Name	Clúster	Intracluster	Intercluster	Connectivity Score
CPX-2341	5	128.555	47.522	2.413216243
CPX-1009	5	123.900	47.522	1.180486790
CPX-1112	5	123.900	47.522	2.404533853
CPX-998	5	123.900	47.522	2.404533853
CALM2	5	73.676	21.028	0.895245987
CALM1	5	72.822	20.323	0.873462407
CPX-6186	5	68.876	29.714	0.867587494
CPX-6189	5	68.876	29.714	0.867587494
CPX-7524	11	109.095	40.809	0.930040156
CPX-7527	11	109.095	40.809	0.930040156
CPX-7530	11	109.095	40.809	0.930040156
CPX-7533	11	109.095	40.809	0.930040156
CPX-7519	11	108.916	40.809	0.930538915
CPX-2280	11	108.614	40.809	0.933068180
CPX-7525	11	108.614	40.809	0.933068180
CPX-7526	11	108.614	40.809	0.933068180
CPX-7531	11	108.614	40.809	0.933068180
CPX-7532	11	108.614	40.809	0.933068180
CPX-7534	11	108.614	40.809	0.933068180
CPX-7535	11	108.614	40.809	0.933068180
CPX-7554	11	108.488	40.809	0.879241702
CPX-7522	11	108.450	40.809	0.932512327
CPX-7523	11	108.450	40.809	0.932512327
CPX-2272	11	108.061	40.809	0.932916189
CPX-2292	11	106.187	52.128	0.960529468
CPX-2295	11	106.187	52.128	0.886360536
CPX-2296	11	106.187	52.128	1.031684658
CPX-2297	11	106.187	52.128	1.031684658
CPX-2298	11	106.187	52.128	0.886360536
CPX-7587	11	106.187	52.128	0.860162981
CPX-7590	11	106.187	52.128	1.031684658
CPX-7592	11	106.187	52.128	0.886360536
CPX-2627	11	104.161	40.809	0.943767907
CPX-2196	11	99.543	30.187	0.871440858
CPX-2212	11	99.543	30.187	0.945478093
CPX-2309	11	99.543	30.187	0.997427050
CPX-2310	11	99.543	30.187	0.871440858
CPX-2318	11	99.543	30.187	0.871440858
CPX-2320	11	99.543	30.187	0.871440858
CPX-2323	11	99.543	30.187	0.871440858
CPX-2330	11	99.543	30.187	0.945478093
CPX-2570	11	99.543	30.187	0.943275894
CPX-2205	11	99.301	33.347	0.897877682
CPX-2307	11	99.301	33.347	0.897877682
CPX-2317	11	99.301	33.347	1.038380372
CPX-2569	11	99.301	33.347	0.971891971
CPX-2555	11	95.921	40.809	0.919957764
CPX-2557	11	95.921	40.809	1.010693051
CPX-2571	11	95.921	40.809	0.935773392
CPX-7111	11	64.399	30.599	0.974144492
CPX-944	11	63.193	40.348	0.996385847

Tabla 5.7: Proteínas T4 seleccionadas con sus métricas de conectividad (intracluster, intercluster y conectivity).

Fuente: Elaboración propia.

De este modo, el siguiente paso respondió a la pregunta: *"¿Cuál es la proteína T4 del clúster k más conectada con otros clústeres y cuál es la más conectada consigo misma?"*. En conjunto con el Ramírez Lab, se definió seleccionar, para cada clúster, las cinco proteínas T4 con mayor `intercluster_connect`, y de forma análoga, las T4 con mayor `intracluster_connect` y `Connectivity_Score`. Dado que los clústeres 12 y 13 no presentaron proteínas T4, se reportan únicamente los resultados de los clústeres 0–11 (Tabla 5.8).

Clúster	Protein (Inter)	Intercluster Score	Protein (Intra)	Intracluster Score	Protein (Connectivity)	Connectivity Score
0	FUCA1	1.598	GNS	10.585	FUCA1	0.749
0	IDS	1.158	NAGLU	10.574	IDUA	0.723
0	ARSB	1.031	CPX-502	9.758	NEU1	0.709
0	B4GALNT1	0.871	CPX-687	9.700	MAN2B1	0.709
0	SGSH	0.463	CPX-686	9.195	NAGLU	0.679
1	CPX-4945	4.126	CPX-6239	27.508	CPX-5675	1.548
1	CPX-6223	4.086	CPX-6165	26.476	CPX-6215	1.457
1	ADAMTS13	2.949	SERPING1	25.392	CPX-6234	1.456
1	SERPING1	1.874	CPX-6179	25.254	CPX-4945	1.376
1	CPX-5602	1.618	CPX-6240	25.052	CPX-6224	1.351
2	RPE65	17.320	CRX	37.587	NRL	0.775
2	CNGB1	11.184	RPE65	37.172	GUCY2D	0.735
2	CNGA1	8.792	PRPH2	36.793	ROM1	0.718
2	GNAT1	8.513	ABCA4	35.055	RD3	0.708
2	FSCN2	7.882	CNGB1	34.890	RDH5	0.705
3	GJB2	2.957	TMC1	19.240	CABP2	0.685
3	TPRN	2.565	CDH23	17.578	PJVK	0.637
3	OTOF	2.540	GRXCR1	17.416	SLC26A5	0.601
3	TRIOBP	2.414	PCDH15	17.190	GRXCR1	0.571
3	ESPN	2.301	GJB2	16.025	TRIOBP	0.571
4	CYB5A	3.174	SLC35A2	32.701	CYP26C1	0.997
4	AKR1D1	2.258	AKR1D1	22.561	SLC35A2	0.984
4	EPHX1	2.199	HSD3B2	19.850	AKR1D1	0.975
4	BLVRA	1.978	CYP26C1	19.080	BLVRA	0.960
4	HSD3B2	1.403	EPHX1	18.642	ABCB4	0.937
5	CPX-2453	51.462	CPX-8641	145.803	CPX-2341	2.413
5	CPX-2454	51.333	CPX-8639	143.664	CPX-998	2.405
5	CPX-2455	51.333	CPX-8642	141.107	CPX-1112	2.405
5	CPX-1009	47.522	CPX-8640	141.020	CPX-1009	1.180
5	CPX-1112	47.522	CPX-8674	139.340	SLC12A5	0.962
6	EARS2	4.096	LARS2	22.080	SARS1	0.701
6	TSFM	3.557	RARS2	21.403	NARS1	0.680
6	DARS2	2.339	YARS2	21.141	NARS2	0.674
6	GTPBP3	2.276	AARS2	20.863	TARS1	0.669
6	SARS1	1.735	EARS2	20.820	DARS2	0.668
7	OTC	7.624	CPX-6233	34.513	CPX-7142	1.543
7	CPX-6175	6.851	ECHS1	28.842	CPX-6233	0.824
7	CPX-6233	5.576	CPX-6175	27.344	PNP	0.781
7	GLUD1	5.311	HMGCL	26.665	MLYCD	0.737
7	ALDH3A2	4.658	GLUD1	25.658	HSD17B4	0.735
8	CPX-1822	39.789	CPX-1727	31.434	CPX-1766	0.792
8	CPX-1770	4.599	CPX-1728	31.434	CPX-390	0.792
8	CPX-1772	4.435	CPX-1729	31.230	CPX-1822	0.783
8	BGN	4.382	CPX-1736	30.146	CPX-1759	0.779
8	CPX-1759	4.047	CPX-1723	28.886	ADAMTS3	0.764
9	IRF1	25.248	IL7	28.994	CPX-833	0.946
9	TLR3	24.485	CPX-6050	26.019	IRF1	0.904
9	TLR7	21.474	CPX-833	25.034	TLR3	0.866
9	IL7	19.470	CPX-383	24.864	CPX-6012	0.865
9	IRF8	18.152	CPX-382	24.132	CPX-6013	0.865
10	CPX-1209	169.223	CPX-1202	39.644	CPX-1218	1.433
10	CPX-1216	169.223	CPX-1194	39.644	CPX-4223	1.433
10	CPX-1219	169.223	CPX-1225	39.644	CPX-1216	1.433
10	CPX-1226	169.223	CPX-1207	39.644	CPX-1217	1.418
10	CPX-1202	169.114	CPX-1203	39.644	CPX-1202	1.418
11	SUMO1	128.989	CPX-7527	109.095	CPX-7601	1.133
11	CPX-2292	52.128	CPX-7530	109.095	CPX-2517	1.132
11	CPX-2295	52.128	CPX-7533	109.095	CPX-2317	1.038
11	CPX-2296	52.128	CPX-7524	109.095	CPX-104	1.033
11	CPX-2297	52.128	CPX-7519	108.916	CPX-2296	1.032

Tabla 5.8: Proteínas T4 priorizadas por las tres métricas: `intercluster_connect`, `intracluster_connect` y `Connectivity_Score` (top 5 por clúster en cada métrica).

Fuente: Elaboración propia.

Estos resultados fueron revisados y aprobados en conjunto con el Ramírez Lab, tanto en la metodología de priorización como en la lista final de proteínas obtenida. Las tres tablas construidas (basadas en las métricas `intercluster_connect`, `intracluster_connect` y `Connectivity_Score`) arrojaron inicialmente 60 proteínas cada una, es decir, 180 en total.

Posteriormente, el Ramírez Lab procesó los datos para identificar las proteínas presentes en más de una de las métricas, consolidando así los resultados y eliminando redundancias.

El proceso culminó en una lista aprobada de **145 proteínas candidatas**, la cual constituye la base para los análisis posteriores y las futuras validaciones experimentales.

Cabe destacar que dicho trabajo de profundización y validación biológica queda en manos del *Ramírez Lab*, responsables de continuar con el estudio detallado de estas proteínas en el contexto de la investigación experimental. Con el fin de favorecer la transparencia y la reproducibilidad, se resalta que la carpeta gnngo en el repositorio de GitHub del proyecto[1] contiene los scripts y notebooks que permiten la replicación completa del pipeline descrito en esta memoria

5.5. Discusión con otros enfoques de la literatura

La metodología propuesta se enmarca dentro de una línea de trabajos que buscan combinar propiedades topológicas de las redes PPI con información funcional proveniente de GO. En esta sección se discuten los resultados obtenidos en relación con algoritmos representativos reportados en la literatura, destacando tanto similitudes como diferencias metodológicas y prácticas.

Modelos como **DC-GC-ANN** [47] alcanzan altos niveles de precisión (96 %) en la clasificación de genes candidatos, integrando centralidad de grado y coloreado de grafos en una red neuronal clásica. No obstante, el propósito de dicho enfoque difiere del presente trabajo. Mientras DC-GC-ANN se centra en la predicción supervisada de genes asociados a enfermedades, el pipeline desarrollado en esta memoria busca detectar módulos funcionales en redes PPI y priorizar proteínas candidatas mediante métricas de conectividad y enriquecimiento funcional. En consecuencia, más que maximizar una métrica de clasificación, la contribución principal radica en la generación de representaciones interpretables y en la priorización de blancos moleculares específicos para Alzheimer.

El enfoque desarrollado en este trabajo se inspira en propuestas como **MTGO** [10] y **LCDA-GO** [38], que combinan información topológica de las redes PPI con anotaciones funcionales de GO para la detección de módulos. Si bien la lógica de integración es compartida, la principal diferencia radica en que el presente pipeline se extiende hacia un marco de Deep Learning mediante GNNs, lo que permite capturar patrones más complejos y, al mismo tiempo, optimizar la eficiencia computacional.

La Tabla 5.9 muestra los tiempos de ejecución y complejidades computacionales reportados para MTGO y LCDA-GO sobre la red Krogan, considerada como punto de referencia en la literatura. Esta red incluye 2,590 proteínas y 7,079 interacciones [38]. En comparación, aunque el presente trabajo no se entrenó sobre dicha red, los experimentos se realizaron en una red PPI construida a partir de datos que alcanzan 5,390 nodos y 430,206 aristas, es decir, aproximadamente el doble de proteínas y más setenta veces la densidad de interacciones respecto de Krogan. A pesar de esta mayor escala, los tiempos de ejecución fueron considerablemente menores: el tiempo máximo registrado durante la búsqueda de hiperparámetros con Optuna fue de 2h 19m por trial (Tabla 5.1), lo cual resulta muy inferior a las 15 horas reportadas para MTGO. Esta diferencia puede atribuirse, en parte, a la mayor eficiencia de las arquitecturas de GNNs, que permiten un aprendizaje directo sobre la estructura del grafo, así como al uso de librerías de Deep Learning que facilitan una ejecución más rápida y escalable. En consecuencia, la propuesta no solo mantiene la integración topológico-funcional, sino que también asegura una escalabilidad superior frente a métodos tradicionales, aspecto crucial para su aplicabilidad en redes PPI de gran tamaño.

[1] https://github.com/macamadridb/alzheimer-target-prediction

Algoritmo	Tiempo (seg)	Complejidad
MTGO [10]	54,000	$O(Kn^3)$
LCDA-GO [38]	47.05	$O(n \log n)$

Tabla 5.9: Comparación de complejidad y tiempo de ejecución entre algoritmos que integran información de GO en la detección de módulos sobre la red Krogan.

Fuente: Adaptado de [38], donde se describe la red PPI de Krogan compuesta por 2,590 proteínas y 7,079 interacciones.

En conclusión, este trabajo se diferencia de enfoques supervisados como DC-GC-ANN al priorizar la detección e interpretación de módulos funcionales en lugar de maximizar métricas de clasificación. Asimismo, frente a métodos híbridos como MTGO y LCDA-GO, la incorporación de GNNs permite abordar redes de mayor escala con tiempos de ejecución reducidos. De este modo, la propuesta combina integración topológico-funcional con aprendizaje profundo, ofreciendo un marco escalable y útil para la priorización de proteínas candidatas en Alzheimer.

Capítulo 6. Conclusiones y Trabajo Futuro

El presente trabajo abordó el problema de identificar módulos funcionales y posibles blancos moleculares asociados a la EA, integrando información topológica de redes PPI con atributos funcionales provenientes de GO. Para ello, se propuso un pipeline no supervisado basado en una GNN entrenada mediante predicción de enlaces, generación de embeddings de proteínas y detección de módulos por clustering, seguido de un análisis de enriquecimiento GO por módulo.

Se logró integrar información topológica y funcional. El uso conjunto de la conectividad de la red PPI y las anotaciones GO permitió obtener representaciones de nodos más informativas, superando la limitación de enfoques puramente topológicos que suelen ignorar funciones biológicas de baja conectividad.

Se implementó un modelo de deep learning basado en GNN con capas GATConv, capaz de incorporar atributos de aristas en el mecanismo de atención. El modelo alcanzó un desempeño robusto, con un AUC de 0.9596, demostrando su capacidad para capturar patrones relevantes de la red y generar embeddings biológicamente significativos. Dichas representaciones fueron la base para la detección de 14 módulos funcionales.

El enfoque propuesto fue validado mediante métricas cuantitativas y análisis funcionales. Se evaluaron métricas de conectividad intraclúster e intercluster junto con el connectivity score para priorizar proteínas con alta relevancia estructural y funcional. Tras la consolidación de resultados (evitando redundancias entre métricas), el proceso culminó en una lista aprobada de **145 proteínas candidatas,** que constituye la base para análisis posteriores y validaciones experimentales; la profundización y validación biológica queda a cargo del Ramírez Lab. A partir de los resultados obtenidos, se identifican varias líneas de investigación y mejora:

- **Validación experimental**: los candidatos priorizados deben ser contrastados con datos experimentales *in vitro* o *in vivo*, para confirmar su rol en mecanismos patológicos de EA.
- **Generalización a otras patologías y/o perfiles biológicos**: el *pipeline* propuesto es replicable en otras enfermedades neurodegenerativas o condiciones biomédicas que cuenten con redes PPI y anotaciones GO.
- **Optimización de modelos**: explorar arquitecturas más avanzadas de GNN y técnicas de auto-supervisión podría mejorar la calidad de los *embeddings*.

En conclusión, el trabajo desarrollado establece una estrategia computacional para integrar información topológica y funcional en redes PPI asociadas a la enfermedad de Alzheimer, generando resultados consistentes y ofreciendo una base metodológica extensible a futuros estudios de descubrimiento de blancos moleculares.

Bibliografía

[1] World Health Organization. Dementia. https://www.who.int/news-room/fact-sheets/detail/dementia, 2025. Accessed: 2025-04-13.

[2] Mayo Clinic. Enfermedad de alzheimer - síntomas y causas, 2023. Accedido el 14 de abril de 2025.

[3] Dementia Statistics. The economic impact of dementia. https://dementiastatistics.org/statistics/the-economic-impact-of-dementia/, 2024. Accessed: July 31, 2025.

[4] S. Chen et al. The global macroeconomic burden of alzheimer's disease and other dementias: estimates and projections for 152 countries or territories. *Lancet Global Health*, 12(9):e1534–e1543, 2024.

[5] Y. Cao, F. Yu, Y. Lyu, and X. Lu. Promising candidates from drug clinical trials: Implications for clinical treatment of alzheimer's disease in china. *Frontiers in Neurology*, 13, 2022.

[6] R. Haussmann, F. Noppes, M. D. Brandt, M. Bauer, and M. Donix. Minireview: Lithium: a therapeutic option in alzheimer's disease and its prodromal stages? *Neuroscience Letters*, 760:136044, Aug 2021.

[7] Dementia Researcher. Who dementia statistics. https://www.dementiaresearcher.nihr.ac.uk/who-dementia-statistics/, 2019. Accessed: July 31, 2025.

[8] Juanita-Dawne R. Bacsu, Raymond J. Spiteri, Kate Nanson, et al. Understanding stigma of dementia during covid-19: a scoping review. *Frontiers in Psychiatry*, 15, 2024.

[9] Roberto Mosca, Tirso Pons, Arnaud Céol, Alfonso Valencia, and Patrick Aloy. Towards a detailed atlas of protein–protein interactions. *Current opinion in structural biology*, 23(6):929–940, 2013.

[10] Danila Vella, Simone Marini, Francesca Vitali, Dario Di Silvestre, Giancarlo Mauri, and Riccardo Bellazzi. Mtgo: Ppi network analysis via topological and functional module identification. *Scientific reports*, 8(1):5499, 2018.

[11] Rita Santos, Oleg Ursu, Anna Gaulton, A Patrícia Bento, Ramesh S Donadi, Cristian G Bologa, Anneli Karlsson, Bissan Al-Lazikani, Anne Hersey, Tudor I Oprea, et al. A comprehensive map of molecular drug targets. *Nature reviews Drug discovery*, 16(1):19–34, 2017.

[12] Justin M Long and David M Holtzman. Alzheimer disease: an update on pathobiology and treatment strategies. *Cell*, 179(2):312–339, 2019.

[13] Albert-László Barabási and Zoltán N. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[14] Eduardo Moreno and Héctor Ramírez. *Grafos: Fundamentos y Algoritmos*. Universidad de Chile, 2011. Accedido el 10 de julio de 2025.

[15] Bruce Alberts, Alexander Johnson, Julian Lewis, David Morgan, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, 6th edition, 2014.

[16] Leland H. Hartwell, Leroy Hood, Michael L. Goldberg, Ann E. Reynolds, Lee M. Silver, and Randy H. Simon. *Genetics: From Genes to Genomes*. McGraw-Hill, 1999.

[17] Michael I. Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome Biology*, 15(12):550, 2014.

[18] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[19] Christopher M. Bishop and Hugh. *Deep Learning: Foundations and Concepts*. Springer, 2024.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–48. Springer, 2012.

[22] Abid Ali Awan. Comprehensive introduction to graph neural networks (gnns) tutorial. https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial, 2023. DataCamp.

[23] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017. arXiv:1609.02907.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

[25] Z. Wang, Y. Zhou, L. Hong, Y. Zou, H. Su, and S. Chen. Pairwise learning for neural link prediction. *arXiv preprint arXiv:2112.02936*, 2022.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. arXiv:1412.6980.

[27] 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 and Data Mining*, pages 2623–2631, 2019.

[28] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In John Shawe-Taylor, Richard Zemel, Peter Bartlett, Fernando Pereira, and Kilian Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[29] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance, 2023.

[30] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023.

[31] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[32] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[33] Mabel Angélica Vidal Miranda. *Machine learning classification of single cell RNA-seq across different types of cáncer*. PhD thesis, Universidad de Concepción, 2022.

[34] Isabelle Rivals, Loïc Personnaz, Laurent Taing, and Marie-Claude Potier. Enrichment or depletion of a go category within a class of genes: which test? *Bioinformatics*, 23(4):401–407, 2007.

[35] G. D. Bader and C. W. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1):1–27, 2003.

[36] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

[37] Edward Y Chen, Christopher M Tan, Yan Kou, Qiaonan Duan, Zichen Wang, Gabriel V Meirelles, Neil R Clark, and Avi Ma'ayan. Enrichr: interactive and collaborative html5 gene list enrichment analysis tool. *BMC Bioinformatics*, 14(1):128, 2013.

[38] C. H. C. Ribeiro E. Kieffer G. Danoy S. Dilmaghani, M. R. Brust and P. Bouvry. From communities to protein complexes: A local community detection algorithm on ppi networks. *PLoS ONE*, 17(1):e0260484, 2022.

[39] S. van Dongen. A cluster algorithm for graphs. *Information Systems*, 2000. (R 0010).

[40] N. Prz˘ulj A. D. King and I. Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics*, 20(17):3013–3020, 2004.

[41] S. H. Tan X. L. Li, C. S. Foo and S. K. Ng. Interaction graph mining for protein complexes using local clique merging. *Genome Informatics*, 16(2):260–269, 2005.

[42] I. Farkas G. Palla, I. Dere´ nyi and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[43] C. K. Kwoh M. Wu, X. Li and S. K. Ng. A core-attachment based method to detect protein complexes in ppi networks. *BMC Bioinformatics*, 10(1):1–16, 2009.

[44] H. Yu T. Nepusz and A. Paccanaro. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9(5):471, 2012.

[45] L. Ou-Yang X. F. Zhang, D. Q. Dai and H. Yan. Detecting overlapping protein complexes based on a generative model with functional and topological properties. *BMC Bioinformatics*, 15(1):1–15, 2014.

[46] L. Hu and K. C. Chan. A density-based clustering approach for identifying overlapping protein complexes with functional preferences. *BMC Bioinformatics*, 16(1):1–16, 2015.

[47] Naveen Sundar Gnanadesigan, Narmadha Dhanasegar, Manjula Devi Ramasamy, Suresh Muthusamy, Om Prava Mishra, Ganesh Kumar Pugalendhi, Suma Christal Mary Sundararajan, and Ashokkumar Ravindaran. An integrated network topology and deep learning model for prediction of alzheimer disease candidate genes. *Soft Computing*, 27(19):14189–14203, 2023.

[48] Daniel F Zhang, Timothy Penwell, Yan-Hua Chen, Addison Koehler, Rui Wu, Shayan Nik Akhtar, and Qun Lu. G-protein signaling in alzheimer's disease: Spatial expression validation of semi-supervised deep learning-based computational framework. *Journal of Neuroscience*, 44(45), 2024.

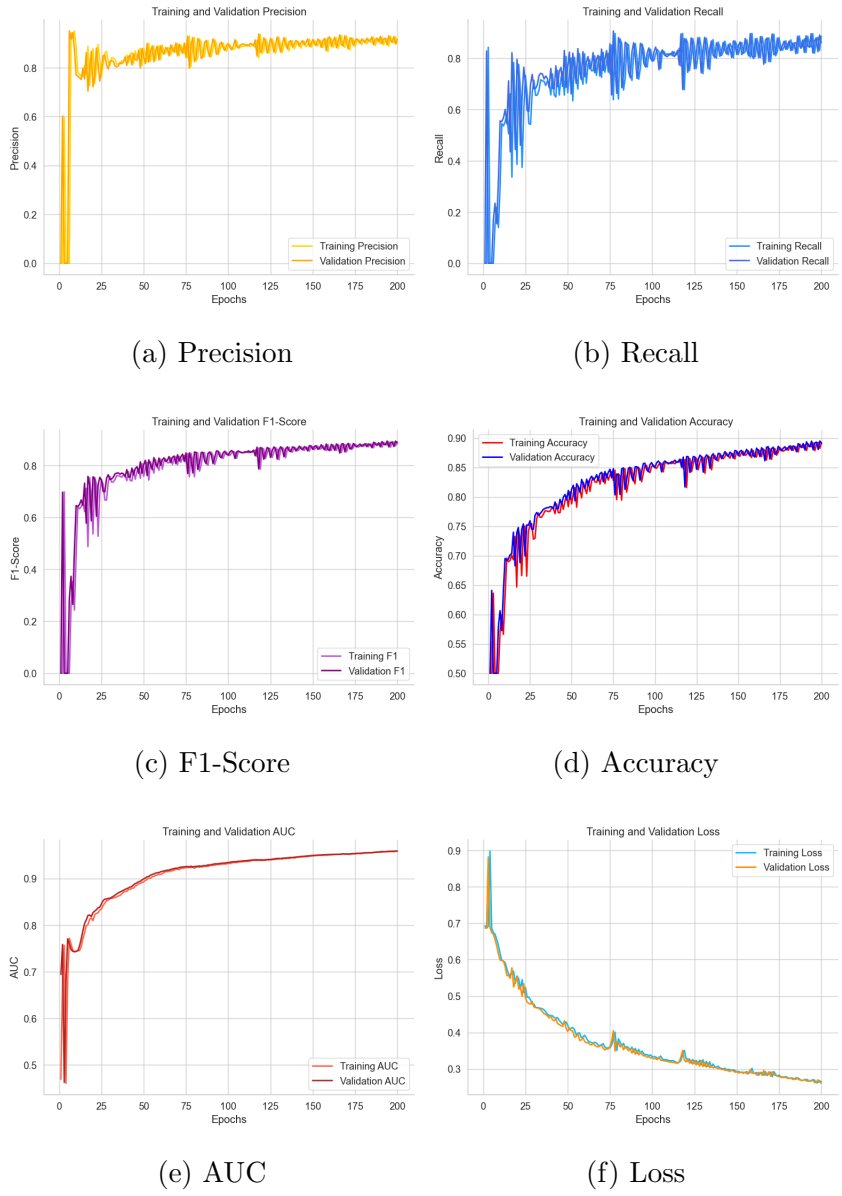
[49] C. K. Kwoh X. Li, M. Wu and S. K. Ng. Computational approaches for detecting protein complexes from protein interaction networks: a survey. *BMC Genomics*, 11(1):1–19, 2010.

[50] X. Li M. Wu and C. K. Kwoh. Algorithms for detecting protein complexes in ppi networks: an evaluation study. pages 15–17, 2008.

[51] Alexander Y Lan and M Ryan Corces. Deep learning approaches for noncoding variant prioritization in neurodegenerative diseases. *Frontiers in Aging Neuroscience*, 14:1027224, 2022.

[52] Elena Galea, Laura D. Weinstock, Raquel Larramona-Arcas, Alyssa F. Pybus, Lydia Giménez-Llort, Carole Escartin, and Levi B. Wood. Multi-transcriptomic analysis points to early organelle dysfunction in human astrocytes in alzheimer's disease. *Neurobiology of Disease*, 166:105655, 2022.

Anexo A. Curvas de entrenamiento del mejor modelo GNN



(a) Precision

(b) Recall

(c) F1-Score

(d) Accuracy

(e) AUC

(f) Loss

Figura A.1: Evolución de métricas por época para el modelo óptimo (Trial 2, Epoch 199) en entrenamiento y validación. En particular, el modelo alcanzó un valor de AUC de 0.9596 en el conjunto de prueba.

Anexo B. Búsqueda de mejor clustering

Criterio	K	Silhouette Score	Davies-Bouldin Index	Calinski-Harabasz Index
Silhouette Score (máximo)	**5**	**0.2806**	1.2480	1676.70
Davies-Bouldin Index (mínimo)	**19**	0.2497	**1.2070**	1078.70
Calinski-Harabasz Index (máximo)	**5**	0.2806	1.2480	**1676.70**

Tabla B.1: Mejores combinaciones de número de clusters K evaluadas con K-Means. Se destacan los mejores valores.

Fuente: Elaboración propia.

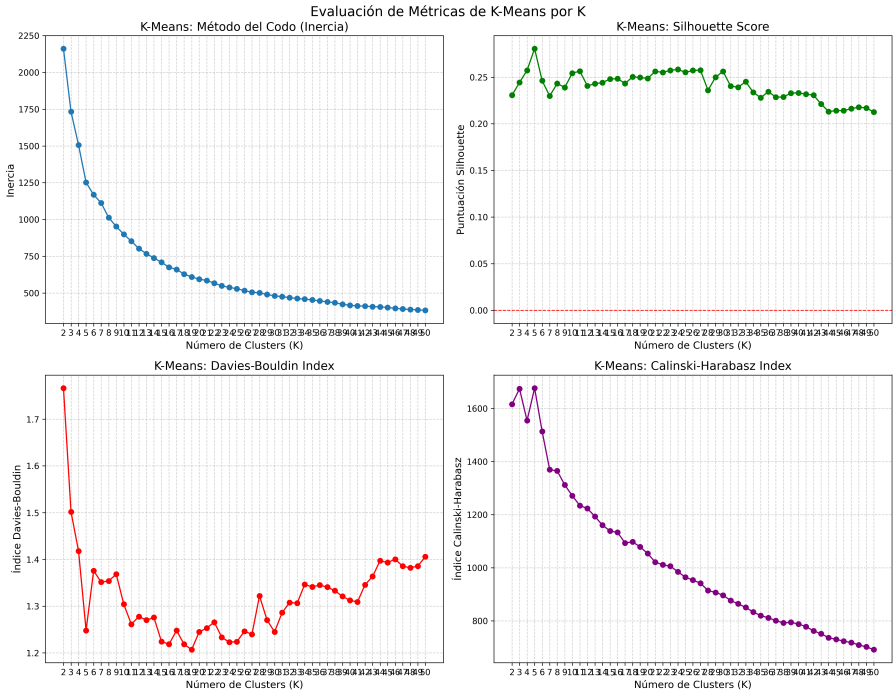
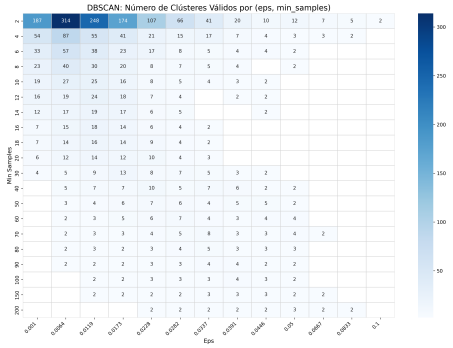


Figura B.1: Evaluación de las métricas de K-Means en función del número de clusters K. Se presentan cuatro criterios: Inercia (método del codo), Silhouette Score, Davies-Bouldin Index e Índice de Calinski-Harabasz.

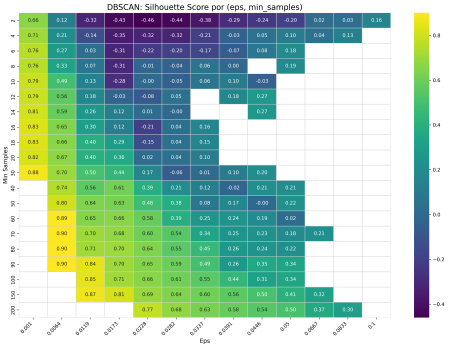
Criterio	Eps	Min Samples	Clusters Válidos	Ruido	% Ruido	Silhouette	Davies-Bouldin	Calinski-Harabasz
Silhouette Score (máximo)	**0.0064**	**90**	2	5116	94.92 %	**0.9006**	0.1465	10792.87
Davies-Bouldin Index (mínimo)	**0.0064**	**90**	2	5116	94.92 %	0.9006	**0.1465**	10792.87
Calinski-Harabasz Index (máximo)	0.0010	16	**7**	5149	95.53 %	0.8275	0.2258	**13183.86**

Tabla B.2: Mejores combinaciones de parámetros `eps` y `min_samples` evaluadas con DBSCAN. Se destacan los mejores valores.

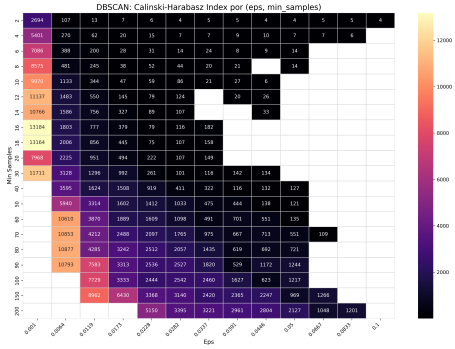
Fuente: Elaboración propia.



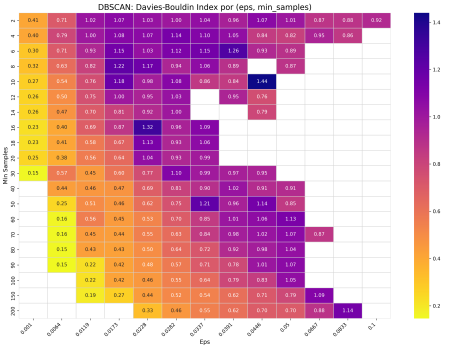
(a) Número de Clusters Válidos



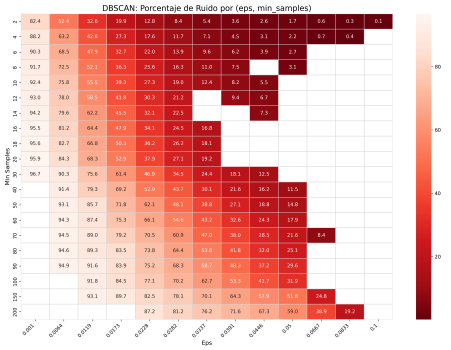
(b) Silhouette Score



(c) Calinski-Harabasz Index



(d) Davies-Bouldin Index



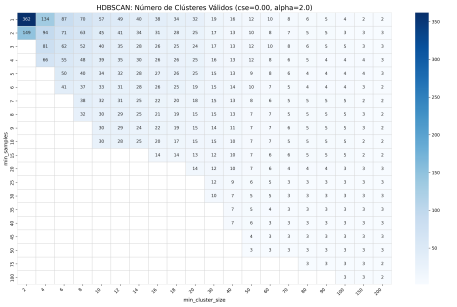
(e) Porcentaje de Ruido

Figura B.2: Heatmaps generados durante la búsqueda de hiperparámetros de DBSCAN, evaluando distintas combinaciones de `eps` y `min_samples`.

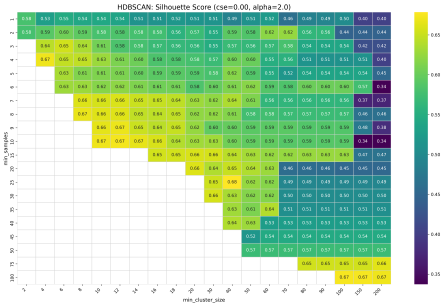
Criterio	Min Cluster Size	Min Samples	Alpha	Clusters Válidos	Ruido	% Ruido	Silhouette	Davies-Bouldin	Calinski-Harabasz
Silhouette Score (máximo)	**40**	**40**	**1.5**	8	4357	80.83 %	**0.6787**	0.4281	1706.83
Davies-Bouldin Index (mínimo)	**20**	**20**	**2.0**	14	4307	79.91 %	0.6605	**0.3897**	1370.82
Calinski-Harabasz Index (máximo)	**200**	**100**	**2.0**	**2**	4422	82.04 %	0.6701	0.4667	**2974.02**

Tabla B.3: Mejores combinaciones de parámetros de `HDBSCAN` evaluadas según tres métricas. El parámetro `cluster_selection_epsilon` se mantuvo en 0.0 (valor por defecto). Se destacan los mejores valores.

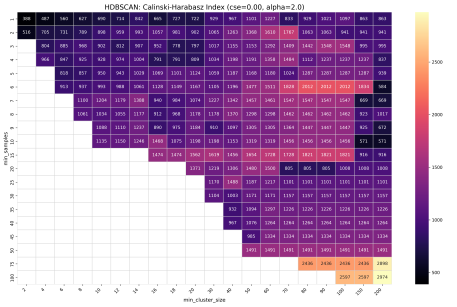
Fuente: Elaboración propia.



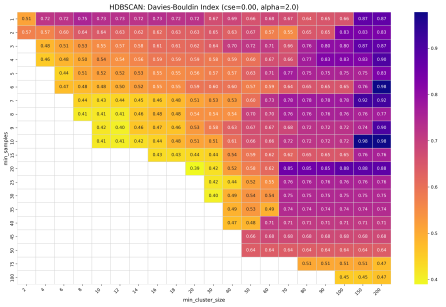
(a) Número de Clusters Válidos



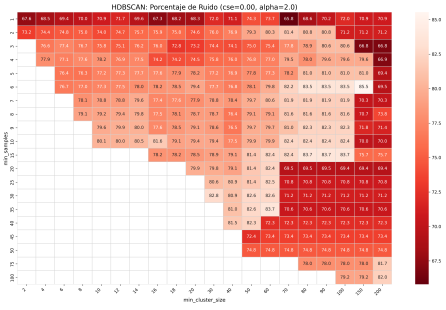
(b) Silhouette Score



(c) Calinski-Harabasz Index



(d) Davies-Bouldin Index



(e) Porcentaje de Ruido

Figura B.3: Heatmaps generados durante la búsqueda de hiperparámetros de HDBSCAN, evaluando distintas combinaciones de `min_cluster_size` y `min_samples`, con `cluster_selection_epsilon = 0.0` y `alpha = 2.0` fijos.

Anexo C. Distribución de métricas topológicas

Percentil	Intracluster	Intercluster	Connectivity_Score
0.25	14.39	0.00	0.638
0.50	24.92	0.55	0.739
0.75	61.42	4.10	0.859

Tabla C.1: Percentiles de las métricas topológicas de conectividad.

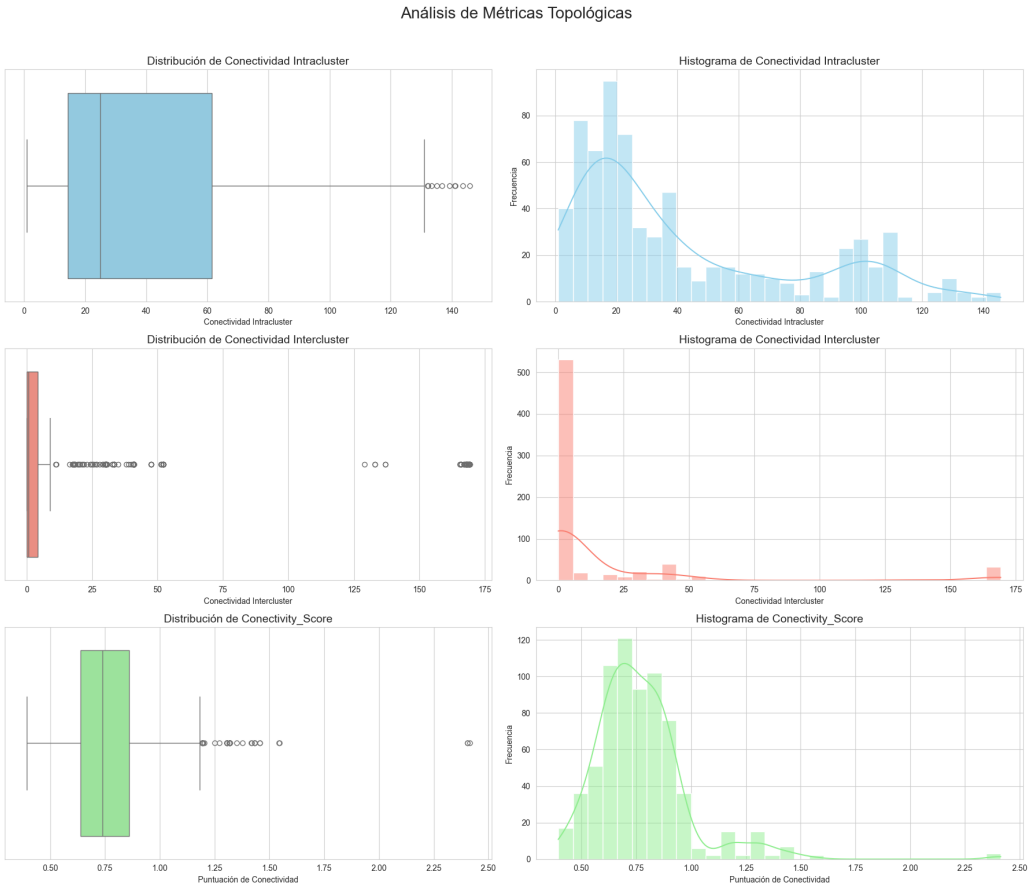


Figura C.1: Distribución de conectividad intracluster, intercluster y `Connectivity_Score`. Se muestran diagramas de caja y histogramas con densidad ajustada, evidenciando la presencia de hubs internos y conectores intermodulares.