Trabajo Fin de Grado

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

GNN para clasificación de artículos científicos

Autor: Pablo Ruiz Valero

Tutor: Alejandro J. del Real Torres

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

GNN para clasificación de artículos científicos

Autor:
Pablo Ruiz Valero

Tutor:
Alejandro J. del Real Torres

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: GNN para clasificación de artículos científicos

Autor: Pablo Ruiz Valero

Tutor: Alejandro J. del Real Torres

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

El Secretario del Tribunal

A mi familia

A mis profesores

Agradecimientos

A lo largo mis años como estudiante universitario he vivido momentos maravillosos y enriquecedores que han contribuido de forma muy positiva a mi desarrollo personal, sin embargo, en aquellos momentos más duros o complicados siempre he podido contar con mi familia y amigos.

En especial me gustaría agradecerles a mis padres la confianza incodicional que siempre han depositado en mí, motivándome a perseguir mis sueños e inculcándome que el trabajo duro lleva a una meta valiosa. Pues no importa equivocarse de camino, sino luchar por el que de verdad queremos seguir. Infinitas gracias.

Este trabajo también va dedicado a mis amigos con los que llevo compartiendo experiencias desde que tan solo éramos unos niños y a día de hoy siguen a mi lado, compartiendo momentos juntos. Han sido un pilar fundamental durante todo este trayecto.

Durante todos estos años he hecho amigos maravillosos que me han acompañado a lo largo de la carrera, junto a los cuales he podido aprender el significado de la palabra compañerismo y esfuerzo.

Por último, me gustaría dedicarles este trabajo a los profesores que durante todos estos años se han preocupado por la formación impartida, siendo los que han contribuido a nuestro desarrollo como personas e ingenieros. En especial me gustaría agradecer la labor de mi tutor Alejandro J. del Real, pues me ha dado la oportunidad de conocer el maravilloso campo del Deep Learning motivándome a realizar mi trabajo final sobre ello. Junto a mi tutor, he contado con el apoyo y consejo de Daniel García Guirao, siendo una pieza fundamental para el desarrollo de este Trabajo Fin de Grado.

Pablo Ruiz Valero

Sevilla, 2023

Resumen

En los últimos tiempos, las aplicaciones e investigaciones relacionadas con la Inteligencia Artificial han supuesto un gran avance en nuestra sociedad, permitiéndonos desde identificar objetos en imágenes hasta generar texto o audio de forma automática. Junto a la necesidad de resolver nuevos problemas, surgen nuevas ideas y estructuras enfocadas al campo de la Inteligencia Artificial, permitiendo aplicar técnicas de Deep Learning para lograr un resultado preciso de la forma más eficiente posible.

Vivimos en un mundo donde todo está conectado de forma directa o indirecta, siendo algunos ejemplos de ello las relaciones personales que se mantienen entre amigos y familia, los enlaces entre moléculas o las uniones vistas como los trayectos de vía que unen distintas estaciones de ferrocarril.

La necesidad de gestionar datos distribuidos en forma de redes y gráficos, en los cuales existen distintas entidades (nodos) unidas en base a relaciones entre ellas (uniones), dio lugar a la aparición de las Redes Neuronales Gráficas o *Graph Neural Networks* conocidas como GNN. Este modelo de red neuronal no solo permite realizar predicciones o análisis del gráfico en función de las propiedades o *features* del elemento analizado, sino que también, permite tener en cuenta para dicha predicción las características de los nodos vecinos y los tipos de uniones que los relacionan.

En este proyecto se presenta una aplicación basada en Redes Neuronales Gráficas cuya función será clasificar distintos artículos científicos según los temas que tratan. Se dispone de un conjunto de datos conformado por nodos y uniones, siendo 3312 artículos científicos los nodos y 4732 referencias entre ellos las que determinarán las uniones del gráfico.

Basándonos en las relaciones entre ellos y propiedades de cada artículo, se planteará y desarrollará el entrenamiento de un modelo basado en GNN para agrupar artículos científicos en 6 categorías diferentes, realizando un posterior análisis de la importancia de cada uno de ellos en base al número de veces que ha sido citado.

Abstract

In recent times, applications and research related to Artificial intelligence have made great strides in our society, allowing us to identify objects in images, generate text or audio automatically and more. Along with the need to solve recent problems, new ideas and structures focused on the field of Artificial Intelligence have emerged, allowing us to apply Deep Learning techniques to achieve precise results as efficiently as possible.

We live in a world where everything is connected directly or indirectly, such as personal relationships between friends and family, links between molecules, or connections seen as the paths of railway lines connecting different stations. The need to manage distributed data in the form of networks and graphs, where there are different entities (nodes) connected based on relationships between them (edges), gave rise to the emergence of Graph Neural Networks, or GNNs. This neural network model not only allows predictions or analysis of the graph based on the properties or features of the analyzed element, but also considers the characteristics of neighboring nodes and the types of connections that link them.

In this project, a Graph Neural Network-based application is presented whose function is to classify different scientific articles according to the topics they address. A dataset is available consisting of nodes and edges, with 3312 scientific articles as nodes and 4732 references between them as edges that determine the connections in the graph. Based on the relationships between them and the properties of each article, the training of a GNN-based model will be proposed and developed to group scientific articles into 6 different categories, followed by an analysis of the importance of each article based on the number of times it has been cited.

Índice

Acrónimos y siglas

GNN	Graph Neural Networks
ANN	Artificial Neural Networks
GCN	Graph Convolutional Networks
CNN	Convolutional Neural Networks
GAT	Graph Attention Network
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
DB	Bases de Datos
IR	Recuperación de Datos
BA	Agentes Biológicos
HCI	Interacción Humano-Computador

1 INTRODUCCIÓN

1.1 Motivación

Durante los últimos años, el término Inteligencia Artificial ha estado muy presente en la sociedad, asociándose a infinidad de aplicaciones y posibilidades tales como el reconocimiento de imágenes, detección de enfermedades o dotar a sistemas de chat una capacidad de compresión y resolución de problemas nunca vistas anteriormente. Dicho potencial ha motivado la rápida expansión del uso de sistemas basados en Deep Learning en inifinidad de áreas. A pesar de su éxito y avance, aún existen multitud de retos que desafían día a día a investigadores y desarrolladores, dejando ver un mundo de posibilidades aún por explorar en el campo de la Inteligencia Artificial.

El avance científico se sustenta en gran parte gracias a la colaboración entre organizaciones y profesionales que comparten resultados experimentales y conclusiones obtenidas, permitiendo establecer puntos de referencia que posibilitan un mayor avance para lograr las metas comunes de la comunidad científica. Si bien es cierto que la divulgación de los avances tiene lugar en conferencias y congresos, son los artículos y 'papers' científicos los que conforman el medio más empleado para conocer los últimos avances referentes a un campo de estudio o tecnología, reforzando la conciencia científica colectiva y desarrollo cultural [1].

Una correcta clasificación de los diversos artículos publicados según campo de estudio, contenido y respaldo bibliográfico permite localizar aquella información de mayor interés, siendo de gran utilidad las aplicaciones y filtros aplicados a dichas publicaciones. En plena era digital y con la versatilidad que nos ofrecen las técnicas basadas en Deep Learning, resultaría de gran utilidad usar ese potencial en la gestión y clasificación de publicaciones científicas mediante estructuras y métodos innovadores en el campo de la Inteligencia Artificial.

La posibilidad de dotar a la comunidad científica con herramientas de filtrado eficientes conllevaría un ahorro de tiempo y un aumento del porcentaje de éxito, no solo para los profesionales que trabajan en el sector de investigación, sino para la sociedad en su conjunto.

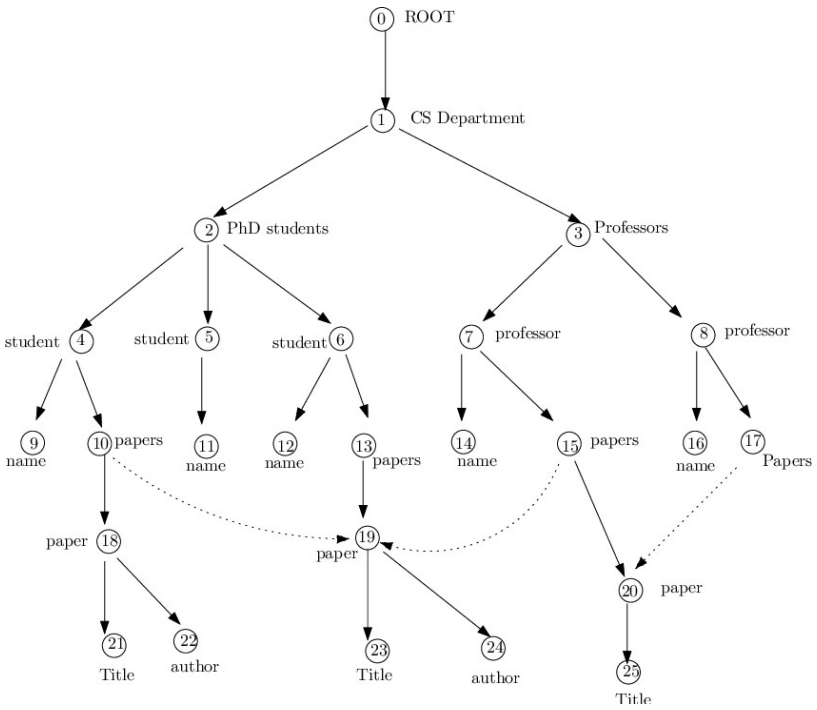


Figura 1.1. Ejemplo de estructuración gráfica de datos [2].

1.2 Objetivos

El objetivo del proyecto será desarrollar un sistema basado en técnicas de Deep Learning para clasificar artículos científicos por categorías de forma automática basándonos, no solo en los propios atributos de cada artículo, sino también en las referencias a otros artículos y al contenido de dichos artículos citados. La gestión de datos estructurados en forma de grafo o red como ocurre en el caso planteado supone un gran reto para los modelos clásicos de Inteligencia Artificial basados en redes neuronales multicapa. Es por ello por lo que se planteará un modelo basado en Redes Neuronales Gráficas con el fin de implementar un sistema de clasificación de artículos científicos según su campos de estudio.

Tras la implementación del modelo para clasificación de los artículos según temática, se procederá a analizar el set de datos que agrupa referencias y artículos con el objetivo de establecer una métrica para identificar artículos potencialemente interesantes, estableciendo una referencia de calidad entre cientos de artículos de un mismo campo.

1.3 Organización

El contenido del documento se encuentra dividido en las siguientes partes:

- **Estado del Arte**: se llevará a cabo una revisión del estado actual de las técnicas de Deep Learning, exponiendo sus fundamentos, aparición y desarrollo en el panorama científico y tecnológico.
- **Redes Neuronales**: presenta los conocimientos básicos necesarios antes de sumergirnos en el mundo de las Redes Neuronales Gráficas o GNN, es conveniente conocer las bases expuestas sobre las que se construye el modelo para gestión de datos estructurados en gráficos. Se analizarán sus principios de funcionamiento, así como los términos y métodos a seguir para su entrenamiento y ajuste de cara a conseguir un correcto rendimiento.
- **Redes Neuronales Gráficas o GNN**: detalla el principio de funcionamiento de una Red Neuronal Gráfica presentando desde un enfoque matemático la distribución de información a través de la estructura. En este capítulo se analizarán las distintas capas de las que se disponen de cara a la realización de un modelo enfocado a predicción y análisis de datos estructurados en forma de red.
- **Clasificador de artículos científicos**: Este capítulo tratará la implementación de los distintos modelos de GNN recogiendo el rendimiento conseguido en la clasificación de los artículos que conforman el dataset CiteSeer, siendo detalladas todas las características de dicho dataset en este apartado.
- **Resultados**: en este apartado se expondrán los resultados obtenidos con cada modelo implementado permitiendo realizar un análisis profundo con la finalidad de poder escoger el modelo con mayor rendimiento para nuestra aplicación de clasificación. En base a los resultados y métricas experimentales obtenidas, se cerrará el apartado desarrollando una conclusión basada en los datos de los experimentos previos y en el desempeño observado de las GNN's aplicadas para clasificación de artículos científicos.

2 ESTADO DEL ARTE

EL análisis de datos estructurados de forma gráfica ha sido un reto afrontado desde diversas perspectivas en etapas previas a la aparición de técnicas de Deep Learning aptas para modelar comportamientos y predicciones para las entidades que conforman la red. Conocer la historia y desarrollo de las primeras aplicaciones de la Inteligencia Artificial y su desarrollo hasta nuestros días nos permite ver la evolución a lo largo de su recorrido, así como la intuición necesaria para afrontar los retos actuales.

2.1 IA y Deep Learning

Como su nombre indica, la Inteligencia Artificial busca replicar las capacidades humanas, empleando algoritmos matemáticos y técnicas de optimización para simular la capacidad de aprendizaje y comprensión humana. Aunque la Inteligencia Artificial, aprendizaje máquina y aprendizaje profundo es un tema de actualidad, el inicio de su estudio y desarrollo comienzó en 1943 con la publicación del concepto de 'neurona artificial' en la revista británica 'Bulletin of Mathematical Biophysics' por parte de Warren Sturgis McCulloch y Walter Pitts [2]. Dicha neurona es la primera aproximación a las redes neuronales modernas y el principio de funcionamiento que sigue intenta imitar el proceso de transmisión y aprendizaje de las neuronas biológicas. Las entradas a dicha neurona se denominan 'dentritas' y la salida recibe el nombre de 'axón'. Para obtener una salida en función de las entradas, dentro de la unidad neuronal tiene lugar la multiplicación de dichas entradas por los pesos W, los cuales serán sumados y finalmente sometidos a una función no lineal la cual se denomina función de activación. A dicha unidad neuronal se le atribuye la capacidad de implementar operaciones lógicas o funciones muy sencillas, sin embargo, a partir de su estudio se llegó a la conclusión de que se podría implementar la funcionalidad de la máquina de Turing si en lugar de emplear una neurona, se emplease una red finita de neuronas convencionales, demostrando el potencial que ofrecía esta nueva tecnología. En la Figura 2.1 se muestra la estructura de la unidad neuronal propuesta.

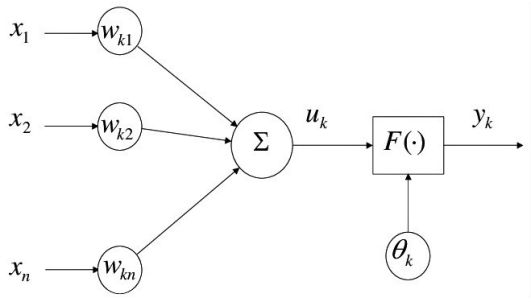


Figura 2.1. Modelo de neurona artificial básica [3].

Pero no será hasta el año 1958 cuando se presente el perceptrón, propuesta desarrollada por Frank Rosenblatt en la cual se mostraba una estructura de aprendizaje y clasificación basada en ajuste de coeficientes de ponderación. Uno de los principales problemas fue la sobreexpectación que desarrolló la posibilidad de implementar algoritmos y sistemas que reprodujeran el proceso de aprendizaje humano basado en neuronas artificiales, pues si bien es cierto que las investigaciones abrían la puerta a un gran abanico de posibilidades, los primeros resultados no cumplían las expectativas y todo ello propició la detención en las investigaciones referentes al aprendizaje basado en redes neuronales.

A pesar del escepticismo en torno al campo de la Inteligencia Artificial, múltiples aplicaciones novedosas aparecieron en la escena, pues el simple hecho de mantener una conversación que podría resultarnos tan trivial y simple para los humanos, llevada a cabo de forma casi involuntaria y sin esfuerzo, supone conocer con profundidad el lenguaje, reglas de uso y estructuras gramaticales. Tal aplicación propia de la ciencia ficción en aquella época, se materializo en 1964 con 'ELIZA', desarrollada por Joseph Weizenbaum en el Instituto de Tecnología de Massachusetts (MIT). Pocos años más tarde, en 1966, se implementaron técnicas de aprendizaje máquina en un robot móvil desarrollado en el Centro de Inteligencia Artificial del Instituto de Investigación de Stanford con Charles Rosen como director del proyecto. El robot, denominado 'Shakey', demostró una cierta capacidad para desglosar y analizar acciones propias [4].

El interés aumenta en gran medida a partir de 1980, cuando el término de retropropagación aparece en la escena, reforzando el concepto de aprendizaje automático y sus posibilidades. Su implementación permitía realizar un ajuste de los pesos, minimizando así el error del modelo propuesto. Este término, no solo será la base de los siguientes avances durante el siglo XX, sino que también será uno de los pilares fundamentales de la tecnológia de aprendizaje máquina y profundo de los que disponemos hoy día [5].Algunos de los avances más significativos han sido desarrollados por grandes compañías y organizaciones como el Instituto de Tecnología de Massachusetts, IBM o Sony. Uno de los hitos más sonados de la época fue la derrota del campeón mundial de ajedrez Garri Kasparov contra la computadora de IBM 'Deep Blue' en el año 1997. El gran rendimiento de las aplicaciones basadas en técnicas de aprendizaje máquina o 'Deep Learning' demostró la superioridad que los algoritmos desarrollados podían tener sobre la capacidad humana a la hora de plantear estrategias o predecir situaciones, permitiendo anticiparse y tomar las decisiones más adecuadas en cada momento.

De forma paralela a la investigación para afrontar nuevos retos como la visión artificial, se le comienza a dar gran importancia a un punto clave para el correcto funcionamiento de modelos de aprendizaje automático o profundo, el acceso a datos de calidad con los que entrenar los modelos y aplicaciones. En el año 2009 fue lanzada la base de datos 'ImageNet' por Fei-Fei Li, formada por 14 millones de imágenes que serían empleadas para el desarrollo de aplicaciones de visión artificial y reconocimiento de objetos. El acceso libre a datos motiva el perfeccionamiento de técnicas de visión, así como establece nuevos retos enfocados a lograr reproducir ciertas capacidades humanas tales como la atención, la comprensión o la síntests de texto o audio.

- **Visión artificial**: En 2012, Alexnet consiguió superar el rendimiento humano en la tarea de reconocimiento de imágenes usando la base de datos 'ImageNet' [5]. La introducción de redes neuronales convolucionales supuso un gran avance pues no solo tomará en cuenta la intensidad de píxeles y color, sino que podrá identificar patrones, texturas, colores y, en base a todo ello, identificar objetos de forma precisa.
- **Atención**: Al igual que el comportamiento animal y la naturaleza han sido una fuente fundamental para el desarrollo de sistemas robóticos, el estudio de las capacidades humanas ha sido muy importantes para el desarrollo de la Inteligencia Artificial. Marcar como objetivo la implementación de dichas capacidades de forma matemática ha permitido dotar a los sistemas inteligentes con nuevas y potentes herramientas para afrontar retos y mejorar su rendimiento Una de las propiedades humanas de mayor interés es la atención. Cuando analizamos un texto, lo generamos o debemos tomar una decisión basándonos en un cierto número de características, los humanos tendemos a prestar más atención a ciertos criterios o elementos los cuales nos permiten generar un texto coherente o focalizar en partes de una imagen que contiene elementos de interés. Diversas capas cuya función es simular la atención cognitiva han sido desarrolladas, siendo los 'Transformers' la estructura de autoatención aplicada a procesamiento de lenguaje con mayor impacto [6].
- **Síntesis de información organizada en redes**: Con el avance de la potencia computacional y un aumento exponencial de datos, se abre la posibilidad de realizar predicciones o análisis basándonos en relaciones o nexos comunes entre entidades en lugar de emplear solo información de la propia entidad como sucede en las estructuras clásicas de redes neuronales. Ante la problemática que se presenta al intentar entrenar estructuras clásicas con datos estruturados en grafos o redes surgen las Redes Neuronales Gráficas o GNN. Estas permiten generalizar y entablar relaciones en función de los datos del nodo analizado y de sus vecinos, teniendo en cuenta el tipo de unión que los relaciona.

2.2 Grafos y redes de datos

2.2.1 Teoría de grafos

Los grafos como estructura de datos constituyen un 'tipo abstracto de datos' formado por vértices o nodos que están unidos mediante aristas. Cada nodo representa un elemento o entidad, definiendo las relaciones que mantiene con otros nodos vecinos. Para tratar problemas desde el punto de vista de los grafos, es importante poder visualizar los datos a tratar como un conjunto de elementos relacionados entre sí donde cada unión puede ser de distinto tipo y con sus propiedades características.

El uso de datos estructurado en grafos se remonta al siglo XVIII con la aparición de la teoría de grafos. El problema planteado mostraba una distribución de siete puentes sobre el río Pregel, definiendo los distintos puntos de comienzo y fin (nodos), unidos por los puentes (aristas). El objetivo consistía en diseñar una ruta para atravesar todos los puentes, pero sin poder pasar más de una vez por ellos. La estructuración de datos en nodos y uniones permitía tener en cuenta información clave en predicciones o agrupación de nodos, siendo estas ventajas las que motivarán su uso en cientas de aplicaciones científicas y de ingeniería. En 1857, Arthur Cayley estudió el problema de enumeración de isómeros en compuestos con distinta estructura molecular [7], siendo representada la estructura de cada compuesto mediante una serie de nodos y aristas.

El uso de grafos para la representación de datos suele ser utilizado para visualizar de una forma clara y estructurada la información, así como para determinar las propiedades de los nodos, agrupar las entidades por categorías, identificar aristas o relaciones clave para nuestra aplicación. Mientras que la visualización y estructuración en tablas de los datos no requiere ningún procedimiento adicional una vez se encuentren los datos en formato de grafo, el análisis de nodos o vértices implica el uso de algoritmos tales como Dijkstra. El uso de este algoritmo está ampliamente extendido en el campo de las comunicaciones, siendo uno de los algoritmos empleados para encontrar la ruta más corta entre distintos router (nodos) unidos entre ellos (aristas), teniendo un coste de paso asociado. Esta aplicación se da sobre sistemas como el representado en la Figura 2.2.

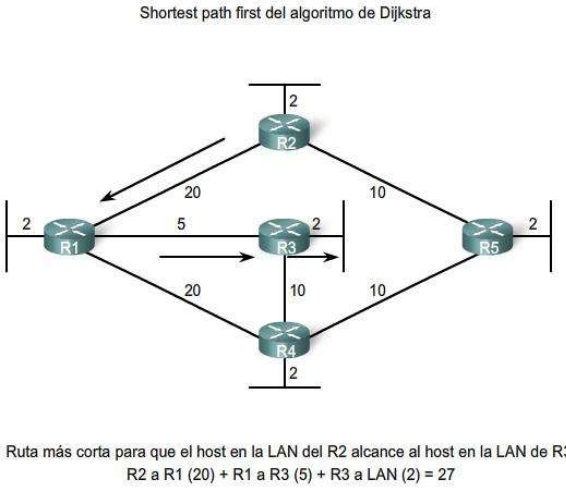


Figura 2.2. Aplicación del algoritmo Dijkstra en red de routers [7].

A partir de 1950 la teoría de grafos se introdujo como herramienta para la sociometría y análisis de relaciones e interacciones sociales, haciendo necesaria la aparición de algoritmos de clasificación o 'clustering' para agrupar los distintos nodos de nuestro grafo según el criterio más conveniente de cara al estudio de la población o elementos de nuestro grafo [9].

2.2.2 Algoritmos de clasificación en gráficos

- **K-means Clustering**: el algoritmo buscará distribuir los nodos de nuestro grafo en K clases, siendo K un valor definido por el analista o impuesto por el set de datos utilizado. La agrupación pretende ubicar nodos con mismas o similares características en una misma categoría, permitiendo así que el propio algoritmo identifique los nodos que son susceptibles de pertenecer a la misma clase. Para ello, el algoritmo K-means emplea como métrica la distancia entre el nodo clasificado y el centroide de la clase asignada, tratando de minimizar en conjunto la suma de las distancias entre los nodos y el centroide del grupo al que pertenecen. Una de sus principales ventajas es la sencillez y velocidad que nos permite conseguir este algoritmo durante su ejecución, sin embargo, presenta diversos problemas de robustez [10].
- **Hierarchical Clustering**: la clasificación de los nodos se produce partiendo desde un enfoque aglomerativo o ascendente en el que todos los nodos se encuentran inicialmente sin agrupar. La división por clases se lleva a cabo mediante un proceso iterativo en el que cada nodo se unirá a otro según el criterio de distancias elegido, conformando así un sub-grafo que será agrupado con un nuevo elemento de la red en el siguiente paso iterativo. En el instante final, todos los nodos pertenecen al mismo cluster, a partit del cual se realizará la división entre las clases correspondientes en base a un valor umbral decidido por el analista [11].
- **Distribution-based Clustering**: algoritmo empleado en casos particulares donde los nodos y datos pertenencen a una distribución como la gaussianas, por ejemplo. Se establece una probabilidad de pertenencia a la clase que disminuirá conforme aumente la distancia al centro de la distribución.

2.2.3 Últimos avances y retos

El procesamiento de datos estructurados como grafos ha suscitado gran interés por parte de empresas y organismos que encuentran en ello una oportunidad para ofrecer soluciones comerciales personalizadas, preveer movimientos sociales o identificar indicios criminales analizando relaciones entre distintas personas y sus actividades. Aunque las posibilidades son infinitas, también lo son las limitaciones, pues el procesamiento de datos distribuidos en grandes grafos con miles de nodos y decenas de miles de aristas deja de ser eficiente y robusto al usar los métodos clásicos. Como alternativa se plantea la posibilidad de emplear modelos basados en redes neuronales pero las redes multicapa convencionales presentan dificultades a la hora de recibir como entrada datos distribuidos en grafos, siendo el coste computacional necesario el principal inconveniente. El modelo de grafos requiere que la información de nodos vecinos pueda propagarse, permitiendo sintetizar y generalizar según el conjunto del grafo y no únicamente basándonos en las características del elemento ‘i’ de nuestro set de datos como si ocurre al emplear redes neuronales clásicas.

Como solución a este problema surgen las Redes Neuronales Gráficas, un modelo basado en aprendizaje máquina el cual trabaja con el conjunto completo de nodos y aristas permitiendo ser entrenada con un enfoque de conjunto. Durante las distintas etapas tiene lugar la propagación de propiedades entre vecinos con un coste computacional bajo y un rendimiento muy elevado, siendo capaces de llevar a cabo predicciones a nivel de nodos, aristas o a nivel de gráfico completo [12].

3 DEEP LEARNING Y REDES NEURONALES

La Inteligencia Artificial ha demostrado ser capaz de afrontar y resolver problemas de diversos ámbitos. A pesar de que su estudio y desarrollo comenzó en el siglo XX, el abánico de posibilidades, enfoques y técnicas asociadas a este campo han dado lugar a la aparición de metodologías y términos que hacen referencia a los algoritmos con los que se abordan los problemas desde el enfoque de la Inteligencia Artificial.

3.1 Conceptos básicos

Para la resolución o planteamiento de modelos es muy común referirse al empleo de técnicas de Machine Learning o Deep Learning, si bien es cierto que ambas pertenecen al campo de Inteligencia Artificial, son varias las características que los diferencian. De cara a la correcta comprensión de la información expuesta en este trabajo resulta muy interesante conocer dichos términos para evitar confusiones.

3.1.1 Machine Learning

El Aprendizaje máquina o *Machine Learning* es la disciplina de la Inteligencia Artificial en la que se busca el aprendizaje de las máquinas o sistemas. A través de un gran volumen de datos, se consigue realizar predicciones e identificar patrones de forma automática. Dentro de esta rama de la Inteligencia Artificial podemos distinguir distintos modelos de aprendizaje o metodologías de entrenamiento:

- **Modelos geométricos**: construidos en un espacio bidmensional o multidimensional donde se establecen umbrales de decisión en forma de funciones lineales que delimitan el espacio de decisión.
- **Modelos probabilísticos**: intentan determinar la distribución probabilística de la función que enlaza las distintas características y sus patrones asociados.
- **Modelos lógicos**: las reglas son transformadas en estructuras de árboles de decisión.

Los algoritmos de aprendizaje más comunes son los siguientes:

- **Aprendizaje no supervisado**: el conjunto de ejemplos empleado solo contiene entradas al sistema, no conociendo las categorías o etiquetas, siendo el propio algoritmo de ML (Machine Learning) el que identifique los patrones de los ejemplos para poder etiquetarlos.
- **Aprendizaje supervisado**: el algoritmo genera una expresión que relaciona las entradas y salidas del sistema. Se emplean en problemas de clasificación donde el sistema asigna el ejemplo de entrada a una de las clases conocidas en base a su vector de características o propiedades.
- **Aprendizaje semisupervisado**: este algoritmo combina ambas estrategias, supervisión y no supervisión, pudiendo así lograr un alto rendimiento al usar datos etiquetados y no etiquetados.
- **Aprendizaje por refuerzo**: en lugar de indicarle al algoritmo la acción a realizar, este debe analizar y comprender el comportamiento del sistema o medio en el que se encuentra en base al método de prueba y error, recibiendo recompensas tras llevar a cabo acciones correctamente.

3.1.2 Deep Learning

El Deep Learning o aprendizaje profundo se trata de un subconjunto de las técnicas de Machine Learning en el que se busca replicar la forma en el que el cerebro humano procesa la información, así como otras características humanas relacionadas con el campo de la visión o síntesis de contenido.

Al igual que ocurre con las técnicas anteriormente expuestas al referirnos al Machine Learning, el Deep Learning también requiere de un entrenamiento. Para el desarrollo de algoritmos de Deep Learning eficientes y su entrenamiento será necesario un gran volumen de datos y disponer de la potencia de cálculo suficiente para procesar dicha información y realizar el computo de algoritmos de entrenamiento.

El desarrollo e investigación de algoritmos de Aprendizaje Profundo promovido por el interés que suscita este campo ha permitido un rápido avance consiguiendo implementar potentes estrucuras y modelos basados en redes neuronales. Dichas redes de neuronas artificiales multicapa (con profundidad) o las redes convolucionales que permiten aproximar la visión artificial a la visión humana empleando detección de patrones y objetos son algunos de los avances más notables de este subconjunto de técnicas de la Inteligencia Artificial.

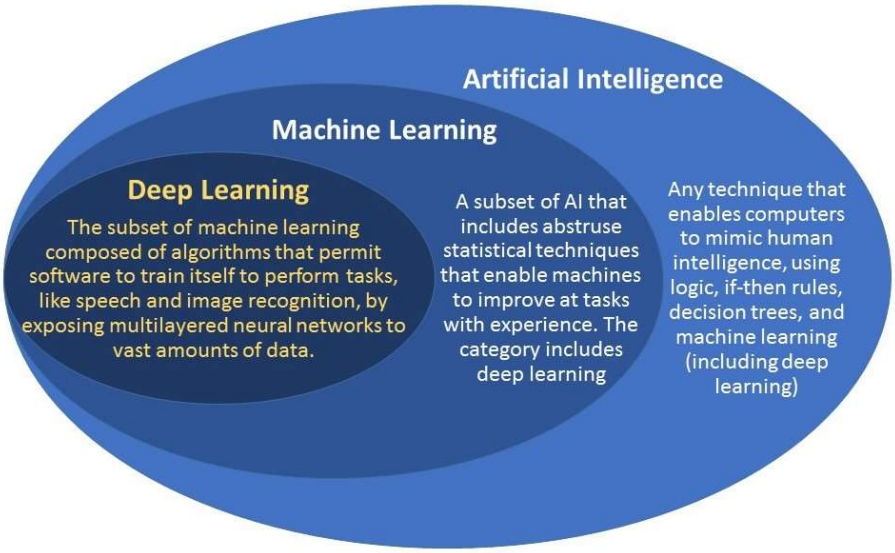


Figura 3.1. Diagrama de conceptos en el campo de la Inteligencia Artificial [13].

3.2 Redes Neuronales

Las redes neuronales son uno de los pilares de la Inteligencia Artificial, pues se basa en parten en el funcionamiento de las neuronas biológicas, siendo modeladas estas en las redes neuronales como neuronas artificiales. Dichas neuronas conforman los nodos que procesarán las distintas entradas, propagando dicha información a través de múltiples capas de neuronas que permiten identificar patrones y características de nuestros datos. Las redes neuronales requieren ser entrenadas con un gran volumen de datos, de forma análoga a como ocurre en el aprendizaje humano, pues somos capaces de identificar si un animal es un perro o no gracias al entrenamiento y estímulos recibidos a lo largo del tiempo. Ver un perro en la televisión junto a explicaciones de qué es y cómo es, observarlo en un parque o dibujarlo en la escuela entrena nuestra capacidad de identificar ese objeto con gran precisión al igual que sucede con redes neuronales enfocadas a visión artificial [14].

3.2.1 Neuronas artificiales

El funcionamiento de las neuronales artificiales está en parte ligado al comportamiento de las neuronas biológicas. Las redes de neuronas son conformadas por múltiples neuronas enlazadas entre sí, donde la salida de una neurona será la entrada de la otra, siendo esta salida dependiente de las características de cada neurona y del grado de conexión de la sinapsis. Este hecho se modela ajustando un valor de pesos que afectan a la entrada de la neurona, ponderando las entradas y permitiendo que, al final de la red, se haya aprendido mediante el ajuste de los pesos la función deseada.

Matemáticamente se modelará como la entrada 'x', multiplicada por los pesos 'w', donde las dimensiones de 'x' y 'w' dependerán del número de entradas asociadas a la neurona. Junto a la múltiplicación de los pesos con las entradas, se computa la suma del sesgo 'b' definiendo la función de la unidad neuronal artificial como:

$$f(x) = x * w + b$$

Como se verá en el siguiente apartado, las neuronas serán enlazadas entre sí consiguiendo mayor capacidad de generalización y extracción de características mediante el paso sucesivo de las entradas por distintas neuronas con distinta ponderación.

Imaginemos que nuestro propósito es desarrollar una aplicación basada en neuronas artificiales con una estructura mononeuronal para realizar la conversión de un valor de temperatura dado en grados Celsius a grados Fahrenheit. Nuestra neurona, en primera instancia, habrá sido inicializada con valores aleatorios de 'w' y 'b', haciendo que esta neurona nos de valores lejos de ser la conversión real de temperatura buscada.

El proceso de entrenamiento que será explicado con detalle en apartados siguientes permite que, a partir de datos y ejemplos de conversiones de Celsius a Fahrenheit, nuestra neurona identifique que los pesos y bias actuales no logran un resultado correcto y poco a poco se realice una corrección de los pesos hasta llegar a la siguiente función:

$$f(x) = x * 1.8 + 32$$

Siendo 'x' un valor de temperatura en grados Celsius y f(x) su correspondiente conversión a grados Fahrenheit.

Aunque este caso expuesto es de los más simple que pueden estudiarse con redes neuronales, la mayoría de las aplicaciones no pueden ser abordadas mediante una única neurona siendo necesaria la interconexión de varias de ellas. Para seguir la filosofía del comportamiento neuronal biológico, la transferencia de información entre neuronas no se realiza de forma directa, sino que se ven afectadas por no linealidades las cuales se traducen a la aplicación de funciones para definer umbrales o aplicar variaciones en el comportamiento de la salida [14].

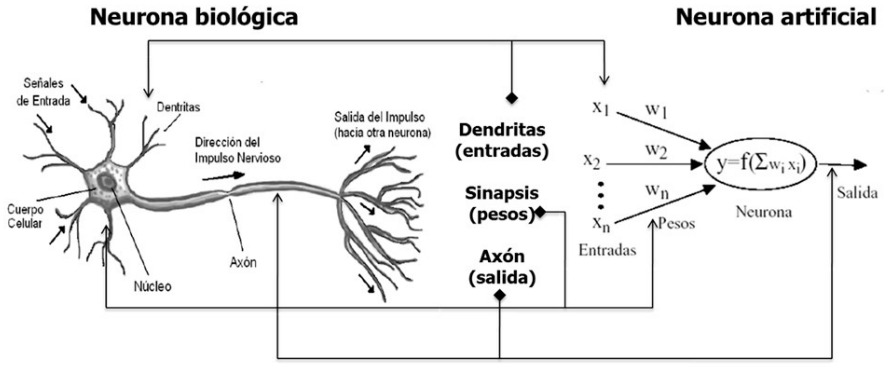


Figura 3.2. Comparación entre neurona artificial y biológica [15].

3.2.2 Arquitectura

Las unidades de procesamiento de redes neuronales se estructuran por capas, siendo la capa de entrada y salida las partes principales que conforman la estructura de las redes multicapa. Aunque el aumento del número de capas y neuronas en cada una de ellas incrementa el coste computacional del entrenamiento del algoritmo, se consiguen extraer características de más bajo nivel conforme se profundiza en la red neuronal, obteniendo mejores resultados y mayor eficiencia que si solo se dispusiera de una capa con gran cantidad de neuronas.

La conectividad entre las capas dispuestas en cascada y las dimensiones de estas puede variar según los modelos o incluso entre cada etapa del entrenamiento. Por motivos de versatilidad, la estructura general más empleada de Redes Neuronales Artificiales o Artificial Neural Network (ANN) es la estructura multicapa con conectividad total [14].

La red estará formada por un número 'L' de capas y un número de neuronas igual a 'n' particularizado en cada capa. A lo largo de la red se computará la operación de multiplicación con pesos 'w' que pasaremos a llamar 'W' al estructurar los pesos como agrupación de vectores y la suma del sesgo 'b' agrupada como vector. En el modelo expuesto en Figura 3.3 se muestra la disposición de capas denominando 'a' a la salida de cada capa y para la salida de la capa final 'y'.

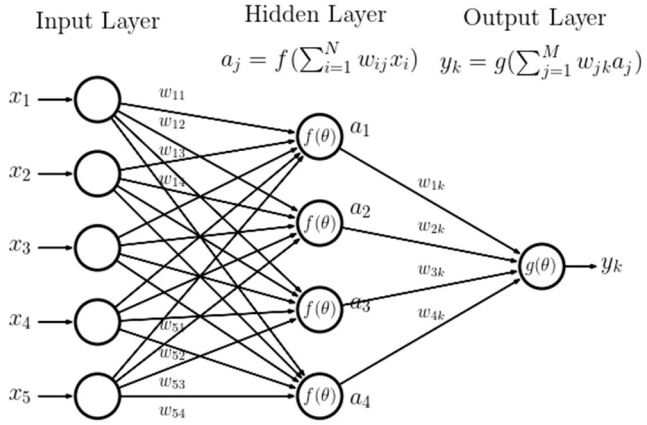


Figura 3.3. Estructura general de red neuronal [16].

El uso de funciones de activación a la salida de las neuronas será necesario para aplicar efectos de no linearidad. El uso de dichas funciones permite definir el nivel de activación de cada neurona así como su contribución e importancia para las siguientes capas. Además, estas funciones nos permiten acotar los valores de salida de las neuronas pues si no se regularan, podrían dar lugar a problemas de cálculo.

De cara a lograr el objetivo final de nuestra red, podremos ajustar la capa final para obtener como salida un valor numérico en el caso de predicción de precios o una probabilidad en el caso de buscar la existencia o no de un elemento en una imagen.

De cara a su aplicación en los modelos de redes neuronales gráficas (GNN), se presentarán las funciones de activación más utilizadas y las expresiones que las definen en la Figura 3.4.

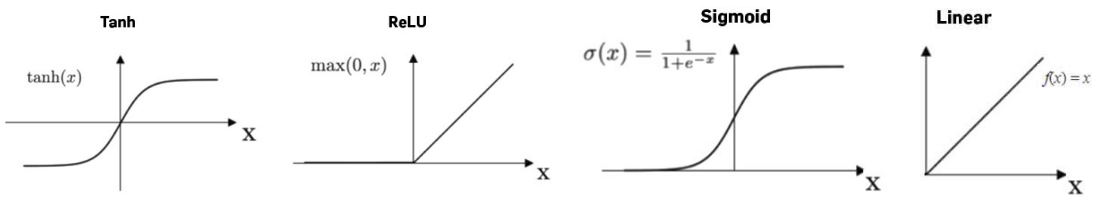


Figura 3.4. Funciones de activación [17].

3.2.3 Introducción a la operación de convolución

Si bien es cierto que las redes neuronales conformadas por neuronas artificiales clásicas tienen una gran capacidad de generalización con respecto a los datos de entrada proporcionados, dejan ver ciertas debilidades en algunas aplicaciones tales como el procesamiento de imágenes. Las entradas o 'inputs' en inglés en una red neuronal serán los valores de intensidad de cada píxel, reestructurando la matriz de valores de la imagen en un vector. Nuestra red generalizará para identificar intensidad, ubicación, densidad de un cierto color o contraste en ciertas zonas. Esto hará que los resultados sean muy pobres y poco fiables ya que la robustez del modelo depende, en el caso de querer identificar gatos, del color del gato con el que se ha entrenado, la perspectiva desde la que se tomó la foto o dimensiones del elemento a identificar.

Otro gran problema que incentivó la búsqueda de modelos más eficientes fue el alto coste computacional que supondría usar una red neuronal con entradas del orden de 3 x Hp x Wp donde Hp es el alto en píxels de la imagen y W el ancho de dicha imagen en pixels y el 3 hace referencia a los canales RGB de la imagen en color. Una imagen cuadrada de 128x128 píxels en RGB supondría una capa de entrada que recibiría 49.152 datos, los cuales debían ser procesados a su vez por las capas posteriores con el elevado coste computacional que ello supondría.

Siguiendo el modelo de la visión y procesamiento humano de las imágenes, se propuso un modelo basado en capas convolucionales para identificar patrones, texturas o formas, y mediante una estructura con mayor profundidad, detectar ojos, orejas, patas u otras partes significativas de los elementos a identificar. A pesar de que este proyecto no trata sobre identificación de objetos ni procesamiento de imágenes, resulta interesante estudiar las bases y fundamentos de la convolución, pues para el correcto funcionamiento de las Redes Neuronales Gráficas interviene una particularización del proceso convolutivo que veremos en este apartado.

La operación de convolución y su implementación como capa de los modelos de aprendizaje profundo permitieron procesar imágenes de forma muy eficiente, extrayendo características geométricas y puntos singulares a identificar. La capa de convolución se fundamenta en el uso de 'kernels' o máscaras que se desplazarán por los píxels de la imagen de forma horizontal y vertical, asignando al píxel sobre el que se encuentra la máscara un valor en función del valor de intensidad de los vecinos enmascarados y los propios valores de la mascara empleada. Cada píxel de la imagen se multiplica por el valor de la máscara asociada y se realiza la suma de esos 8 valores tal y como se representa en la Figura 3.5.

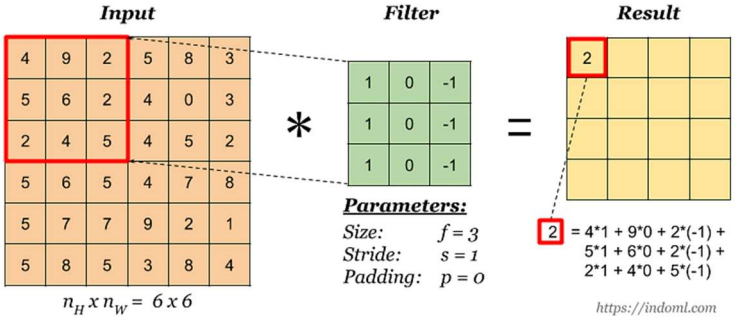


Figura 3.5. Operación de convolución [18].

Esta operación permite reducir el coste computacional, pues el ajuste de los pesos de la máscara (valores de la máscara) supone menor coste que definir los pesos para las entradas a nivel de píxel.

El verdadero valor que aporta esta capa junto con la operación de convolución que implementa reside en la capacidad de sintetizar en un único valor la contribución de los píxels vecinos siendo capaz de ajustar las ponderaciones atribuidas a cada elemento para conseguir mayor rendimiento. Gracias a la síntesis de información de píxels vecinos, en las GNN se pudo aplicar un mecanismo similar gracias al cual se tiene en cuenta la información del nodo analizado y sus vecinos directos.

3.2.4 Entrenamiento y proceso de aprendizaje

Las unidades neuronales y la combinación de estas en forma de redes permiten mediante un proceso de aprendizaje, identificar características o patrones útiles para mejorar su rendimiento. La sucesión de operaciones matemáticas y funciones de activación por las que se procesa la entrada da lugar a la salida de nuestra red neuronal, que como se comentó antes, puede ser un valor numérico o un valor probabilístico. Este proceso se llama propagación hacia delante o 'forward propagation' y es probable que al ser realizada con los pesos 'w' y sesgo 'b' inicializados de forma aleatoria, obtengamos unos resultados poco precisos y dispares.

La técnica empleada para el ajuste de los pesos y sesgo se denomina propagación hacia atrás o 'back propagation'. Este término fue expuesto en 1960 [19] en el campo de teoría de control, siendo desarrollado en 1962 por Frank Rosenblatt [20] aunque encontrando dificultades para su implementación. Será en 1985 cuando David E. Rumelhart presente su propuesta de uso de la propagación hacia atrás para ajuste de pesos en una red neuronal.

Para proceder al ajuste de los pesos, debemos encontrar una métrica que nos indique el rendimiento de nuestro modelo, y en caso de no ser óptimo, nos indique que variación en los pesos y sesgo del modelo incrementarán la precision del mismo. Para ello se empleará la función de coste, la cual indicará mediante un valor numérico la correcta o errónea predicción en nuestro modelo. Un ejemplo de función de coste es el error cuadrático computado con la predicción del modelo, habiendo obtenido puntos predichos los cuales se esperan que se ajusten sobre una recta. Si el error cuadrático es elevado, se determinará un mal funcionamiento de nuestro modelo y será necesario un ajuste de los parámetros. Como se comentó anteriormente esta función depende de los pesos, sesgo, entradas y salidas esperadas, pues se realiza una comparación entre los valores predichos y los esperados.

Una función de coste nos indica un buen funcionamiento de nuestro modelo cuando la métrica asociada a dicha función es mínima o próxima a cero, encontrándose este punto deseado en el punto mínimo de la función. Para el ajuste de pesos y sesgo, se emplea el 'algoritmo de descenso del Gradiente'. Para aproximarnos al mínimo de la función de coste mediante el ajuste de parámetros, será necesario calcular la dirección en la que el cambio de valor de los pesos o sesgo contribuyan a minimizar la función de coste, y ello consiste en el cálculo del gradiente de la función de coste con respecto a los parámetros a modificar, pesos y sesgo. Para conseguir disminuir el coste computacional requerido durante el entrenamiento, se sigue la regla de la cadena en derivación desde la última capa a la primera, permitiendo emplear las derivadas con respecto a los pesos de capas finales para calcular la derivada de las capas iniciales [21]. Si los parámetros son agrupados en una matriz 'W', siendo 'f' la función de coste, el gradiente de dicha función se definirá con la siguiente expresión:

$$dw^L = \frac{\partial f}{\partial W^l} \; para \; l = 1, \dots, L$$

El cálculo de la dirección de descenso de gradiente permitirá que aplicando el ajuste de parámetros sustrayendo o sumando un valor de forma iterativa durante cada paso del aprendizaje, nos aproximemos al valor deseado tal y como se aprecia en la Figura 3.6.

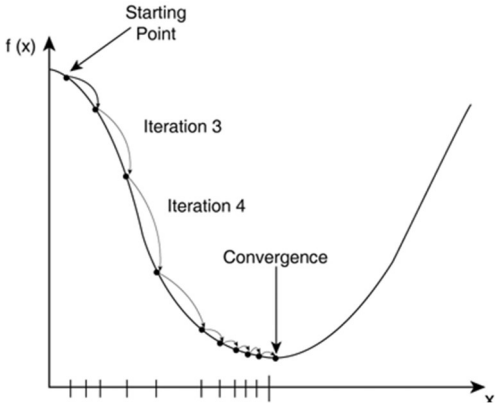


Figura 3.6. Direcciones de descenso de gradiente en función de coste. [21]

Una vez computadas las derivadas asociadas a los parámetros sujetos al ajuste, serán modificados mediante la implementación del algoritmo de Descenso del Gradiente. Siendo denominada la pendiente o dirección de descenso *dw*, los pesos de la capa *l* se modificarán sustrayendo al valor de dichos pesos una determinada cantidad dependiente de la dirección de descenso de gradiente y un hiperparámetro de aprendizaje llamado 'tasa de aprendizaje'. Una tasa de aprendizaje alta hará que el progreso sea más veloz, sin embargo, el uso de un valor elevado para este hiperparámetro podría acarrear un mal ajuste de los pesos, pues los pasos grandes en el proceso de ajuste podrían dificultar la llegada al mínimo o a valores aceptables en la función de coste. La expresión general del ajuste de pesos, una vez calculada la dirección de descenso *dw*, queda definida de la siguiente forma [22]:

$$W_{i+1}^{L} = W_{i}^{L} - dw_{i}^{L} lr_{i}$$

Siendo *lr* el valor de la tasa de aprendizaje para ajustar la magnitud de avance en la dirección del gradiente calculado. El subíndice *i* hace referencia a las etapas de entrenamiento indicando que el ajuste de dichos pesos es un proceso iterativo, pues en cada paso de propagación hacia atrás se modifica el valor de los pesos anteriores.

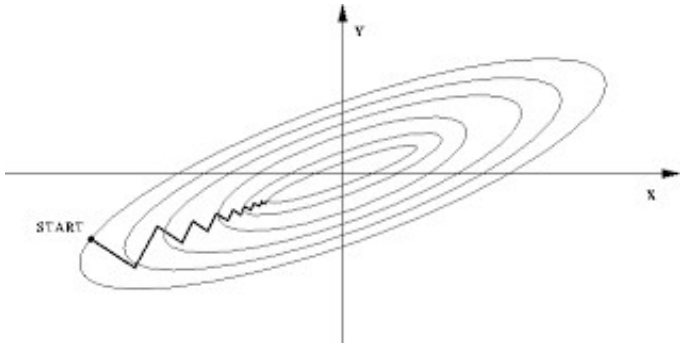


Figura 3.7. Aproximación a mínimo de la función de coste iterativamente [23].

Para mejorar el progreso de aprendizaje existen numerosas técnicas las cuales afectan a la cantidad de datos que requieren para ser procesados y evaluados por la función de coste, siendo algunos de los más empleados los presentados a continuación:

- **Descenso del Gradiente Estocástico**: usando una época de entrenamiento para cada ejemplo del set escogido para entrenar el modelo y reajustando parámetros en cada paso.
- **Descenso del Gradiente por Lotes**: usando una época de entrenamiento en un conjunto de ejemplos del set empleado para entrenar el modelo y ajustando los parámetros tras la evaluación de cada lote en la función de coste.

El algoritmo de ajuste trata de conseguir alcanzar el mínimo en la función de coste, comparando los valores predichos con los resultados esperados. Sin embargo, aunque nuestro modelo consiguiese minimizar el coste con los datos empleados para computar el algoritmo de gradiente, podría darse el caso de que no hubiese generalizado bien y su rendimiento con datos de entrada no usados para entrenamiento sea muy inferior al esperado. Para asegurar que el modelo tiene un buen desempeño con datos que nunca han sido usados para el entrenamiento se separará nuestro set de datos en set de entrenamiento, de validación y de test, siendo este último el que nos permitirá comprobar el correcto funcionamiento del modelo de cara a un uso con datos no conocidos por nuestro algoritmo de Deep Learning.

3.2.5 Principales problemas de sesgo y varianza

Cualquier modelo de Deep Learning contará con errores asociados al ruido intrínseco de los datos, al valor de la varianza y valor del sesgo del modelo. Antes de explicar los distintos errores debe asumirse que el ruido propio de los datos de nuestro set no podrá ser evitado, haciendo que el problema principal a la hora de conseguir un buen rendimiento sea encontrar un equilibrio entre el valor de sesgo o bias y varianza [24].

- **Varianza**: la varianza es un estimador que nos indica cuanto variaría la predicción según los datos que se usen durante el entrenamiento. Si la varianza es alta, significa que la función que ajusta el modelo de Deep Learning será distinta si se emplean otros datos para ser entrenada, siendo este un indicador de que el modelo no generaliza de forma correcta y presta demasiada atención a las particularidades de los datos de entrenamiento.
- **Sesgo**: el sesgo indica la diferencia entre el valor predicho por el modelo y el valor real, es decir, indicará si el modelo arroja predicciones con poca diferencia entre el valor real y el predicho o, por el contrario, el modelo entrenado es demasiado simple y no ha aprendido a generalizar ni con los rasgos y propiedades más básicos del set de datos de entrenamiento.

Los problemas de sesgo y varianza dan lugar a que se produzca sobreajuste u subajuste.

- **Subajuste o Underfitting**: este fenómeno se da cuando el sesgo es demasiado elevado. La función aproximada por el modelo entrenado no tiene la complejidad necesaria ni ha adquirido la capacidad de discernir patrones generales de los datos de entrenamiento. Un modelo que sufre de subajuste estará relacionado con altos valores de error durante el entrenamiento usando el training set y verificación al emplear el test set.
- **Sobreajuste u Overfitting**: este fenómeno está relacionado con un valor elevado de varianza, obteniendo un modelo con un error asociado a los datos de entrenamiento muy bajo, pero al ser sometido a datos no vistos previamente, el valor del error aumenta en gran media, poniendo de manifiesto el hecho de que el modelo particulariza la predicción en base a los detalles asociados a los datos de entrenamiento en lugar de generalizar para el conjunto de datos completo.

Para abordar dichos problemas debemos ajustar dentro de lo posible la complejidad del modelo evitando el sobreajuste, buscando un modelo cuya complejidad nos permita, para una cantidad de épocas de entrenamiento, conseguir un equilibrio entre varianza y bias que se apreciará cuando los errores obtenidos en las predicciones con datos de entrenamiento son similares a los errores empleando el set de validación y test. El uso de una cantidad mayor de datos de entrenamiento puede aportar información útil al modelo de cara a generalizar acerca de las características que permitirán al modelo ajustarse de forma balanceada a los datos de entrenamiento para encontrar un equilibrio como el que se representa en la Figura 3.8.

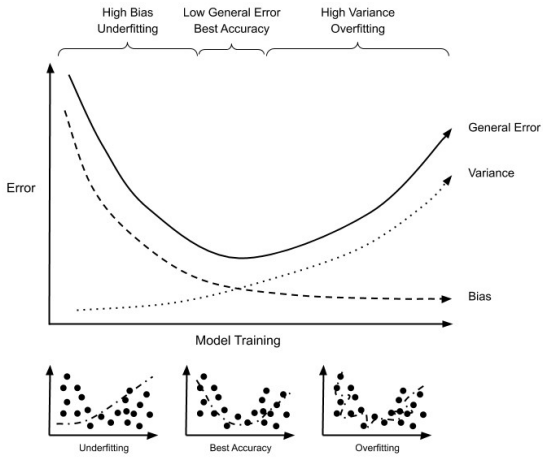


Figura 3.8. Análisis gráfico de problemas de varianza y sesgo [25].

3.2.6 Métricas de validación

Una de las métricas más empleadas en Deep Learning para validación de un modelo consiste en medir la exactitud, llamada *accuracy* en inglés, de nuestro modelo. Esta métrica consiste en identificar el porcentaje de ejemplos correctamente clasificados.

Para evaluar la información obtenida nos encontramos ante distintas casuíticas a la hora de resolver problemas mediante técnicas de Deep Learning, existiendo aplicaciones en las cuales un alto porcentaje de falsos positivos o falsos negativos invalidarían nuestro modelo a pesar de que, de forma general, puede que tenga una buena exactitud.

Junto a la exactitud se tiene en cuenta el valor de *loss* o pérdida, indicado por la función de coste, siendo un indicador del ajuste de nuestro modelo a las características del set de entrenamiento. Todas estas métricas permiten preveer el comportamiento del modelo e intuir si nuestro modelo debiera ser entrenado durante más o menos épocas, dando lugar en este último caso a la finalización de entrenamiento de forma prematura o con *early stopping* al comprobar que un entrenamiento demasiado largo empeora el rendimiento del modelo [26] como puede apreciarse en la Figura 3.9.

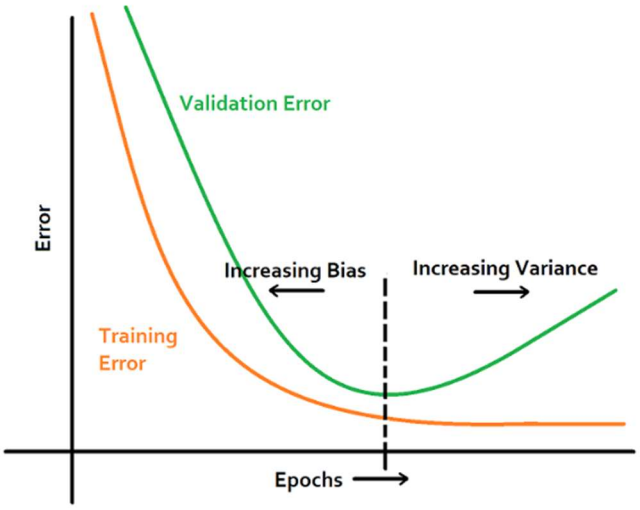


Figura 3.9. Early stopping en función del error obtenido [27].

4 REDES NEURONALES GRÁFICAS

4.1. Representación de datos en grafos

4.1.1 Datos estructurados en grafos

Tal y como se presentó en la sección introductoria del documento presente, los grafos permiten establecer conexiones entre distintos objetos o entidades, permitiendo visualizar de forma directa la relación entre los distintos nodos. Tanto los elementos dispuestos en el grafo como las relaciones entre ellos no están restringidos a un cierto campo de aplicación o una aplicación concreta, sino que gran parte de las relaciones que pueden ser identificadas entre dos objetos o elementos permitirán una adaptación a formato de grafos. Para ejemplificar el uso de grafos en diversos casos, podríamos plantear su uso a la hora de representar un conjunto de personas cuyas uniones establecen las relaciones interpersonales que mantienen. La versatilidad que nos aporta ha incentivado la investigación y el desarrollo de algoritmos para tratamiento de dichas estructuras.

La posibilidad de establecer vínculos interpersonales puede ser trasladado a la relación entre consumidores y productos, brindando a compañías y centros de gestión de datos una información de gran valor de cara al análisis y estrategias de mercado, pudiendo tener en cuenta factores derivados de los vínculos establecidos entre los elementos.

Los grafos pueden ser simples o constar de diversos parámetros y características asociadas que enriquecen la información que representan.

4.1.2 Estructura del grafo y atributos

Un grafo (G) queda definido por un conjunto de nodos o vértices (V) relacionados por aristas, o *edges* (E) en inglés. Una vez son conocidos los elementos que conforman el gráfico y las uniones entre cada uno de ellos, podrá ser representado y quedará definido por la siguiente expresión formal como:

$$G(V, E)$$

Cada grafo representa un conjunto de elementos que lo conforman, pudiendo tener asociado cada uno de ellos un conjunto de características propias. En el ejemplo expuesto en el apartado anterior, las personas que representarían a los nodos del grafo tendrán ciertos atributos tales como pudieran ser la edad, estudios o poder adquisitivo. En base a estos atributos, las uniones que representan las aristas podrán ser vistas como un vínculo dependiente, en parte o en su totalidad, de las características del sujeto en cuestión [28].

De la misma forma que los nodos pueden tener asociado un vector de características propio, las aristas pueden disponer de información asociada la cual podría ser un valor probabilístico, ejemplificando las uniones que podrían representar la probabilidad de que un consumidor compre o no un producto, o un valor numérico si dicha arista representa una transacción económica y el valor asociado es el importe del pago. En el caso de que las aristas no contengan información adicional, se denominarán aristas de tipo binario, quedando indicada la existencia de una arista por el valor ‘1’ y la ausencia de esta por el valor ‘0’.

Ciertos grafos pueden necesitar que las aristas establecidas entre los nodos tengan una dirección única, indicando que la relación en cuestión se produce del nodo A al nodo B y no al contrario. En base a esta característica, los grafos podrán ser de dos tipos [28]:

- **Grafo direccionado**: las aristas constan de una dirección única, permitiendo identificar con claridad el nodo orígen y destino del vínculo. Algunos ejemplos en los que es necesario representar la dirección del vínculo podrían ser aquellos en los que se produce una operación con destinatario como podría ser una transferencia o para indicar explícitamente que la unión en el sentido contrario no es físicamente posibles, pudiendo trasladarse a dos nodos que representen puntos urbanos y la arista una vía de sentido único que los comunica.
- **Grafo no direccionado**: en este caso la relación no precisa que sea asiganda una dirección, pues la misma arista actuará como una doble unión direccionada entre los nodos. Si la arista representa una unión entre dos moléculas dentro de una estructura química, no es relevante establecer una dirección específica.

La combinación de los distintos tipos de aristas, atributos y distribuciones permite extender el uso de grafos a casi cualquier campo, consiguiendo agrupar grandes volúmenes de información de diversa tipología y procedencia, tal y como representa en la Figura 4.1 referente a conexiones entre distintas ubicaciones.

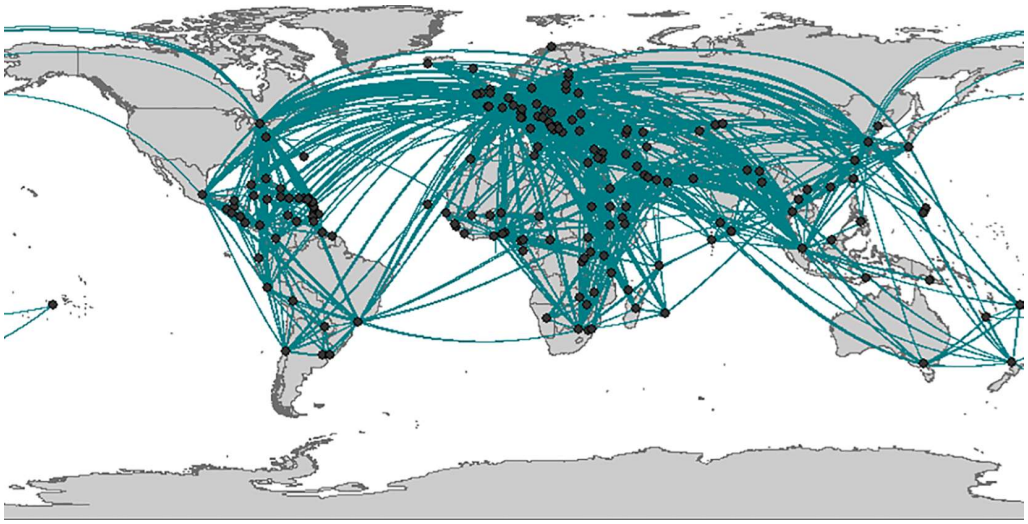


Figura 4.1. Representación de trayectos aéreos entre aeropuertos [29].

La información que puede extraerse de los grafos no se limita únicamente a las uniones o atributo de los nodos, sino que en ocasiones el interés del análisis de grafos reside en evaluar el grafo completo con el fin de determinar una propiedad o conclusión sobre el conjunto.

Dicha aproximación permite evaluar uniones entre moléculas con el fin de determinar el compuesto que esta siendo representado en el grafo.

4.1.3 Propiedades y restricciones de los grafos

Los grafos, al igual que otras distribuciones de datos, tienen características intrínsecas relacionadas con la disposición de los datos, imposibilitando que sean tratados de forma similar a como se procede con imágenes o secuencias. Para tratar los grafos será necesario emplear algoritmos teniendo en cuenta las siguientes restricciones y condiciones necesarias [30]:

- **Aplicable a espacio no-euclídeo**: los nodos y aristas del grafo no se encuentran ubicados en un espacio euclídeo, pues no existen dimensiones consistentes entre los elementos como si ocurre por ejemplo con los píxels de una imagen, por lo que el método de análisis propuesto debe ser capaz de tratar datos distribuidos en un espacio no euclídeo.
- **Tolerante a cambios de dimensión**: el número de nodos y aristas que conforman un grafo puede variar con la incorporación de nuevos datos, haciendo que los algoritmos cuyos parámetros asociados a la dimensión de entrada son fijos no permitan tratar datos estructurados en forma de grafos.
- **Invariable a permutación**: la permuta de dos nodos del gráfico de entrada no afecta a la salida. El orden de los nodos puede variar, y junto a él, el orden en el que se representarán las aristas del grafo [31].

4.2. GNN

4.2.1. Enfoque clásico mediante redes neuronales

Una vez ha sido estudiada la organización de los datos, tras ver la capacidad de establecer relaciones mediante vínculos y atributos en los grafos, surge el interés por tratar estos datos mediante modelos basados en técnicas de Deep Learning para predecir comportamientos de entidades o establecer vínculos no existentes entre los nodos. La posibilidad de tratar grafos mediante redes neuronales multicapas convencionales es descartada rápidamente atendiendo a las restricciones previamente comentadas, pues la capa de entradas se fija durante la creación del modelo, no admitiendo variaciones en las dimensiones.

Otro enfoque de gran interés consiste en la posibilidad de tratar a los nodos del grafo de forma similar a como se disponen los píxels en una imagen con el objetivo de aplicar la operación de convolución. El principal problema al aplicar este método es la distribución de los datos en el espacio no euclídeo, imposibilitando el uso de una máscara convencional de convolución.

Si bien el uso de la operación de convolución clásica permitiría obtener un modelo más eficiente computacionalmente que si se usaran redes neuronales multicapa, nos encontramos con que los sets de datos e información necesaria para abordar problemas reales con grafos pueden llegar a constar de cientos de miles de nodos y aristas, pudiendo tener asociado cada uno de ellos un vector de características, disparando las dimensiones de la capa de entrada a valores de varios millones de *inputs*. El coste computacional necesario para el entrenamiento y ajuste de pesos de una red semejante incentiva el desarrollo de un modelo capaz de manejar y tratar los datos estructurados en grafos mediante técnicas de Deep Learning de forma más eficiente, dando lugar a las Redes Neuronales Gráficas.

4.2.2. Preparación de los datos del grafo

Hasta el momento hemos analizado los elementos que conforman el grafo, sin embargo, no se ha analizado el formato en el cual se agruparán los distintos nodos, conexiones entre ellos y atributos de forma previa a poder ser tratados como la entrada de un modelo basado en técnicas de Deep Learning. Los modelos basados en redes neuronales multicapa y redes neuronales convolucionales tratan los datos en forma de vectores y matrices puesto que estas estructuras facilitan el análisis, acceso y recorrido a través de la información. Las uniones entre nodos serán tratadas dentro de la matriz de adyacencia mientras que los atributos de los nodos se agruparán en la matriz de propiedades, siendo estas matrices la base para tratar los datos estructurados en grafos.

Tanto el número de nodos existentes en nuestro set de datos (N), como el número de aristas (M) y la dimensión del vector de propiedades de cada elemento (D) quedarán definidas por las dimensiones de la matriz de propiedades (X) y la matriz de adyacencia (A).

Para explicar de forma clara y concisa en que consiste la matriz de adyacencia y la matriz propiedades, estas serán definidas en base al siguiente grafo no direccionado con aristas binarias que consta de 5 nodos, 6 aristas y cada nodo queda definido por un vector de propiedades de 3 componentes.

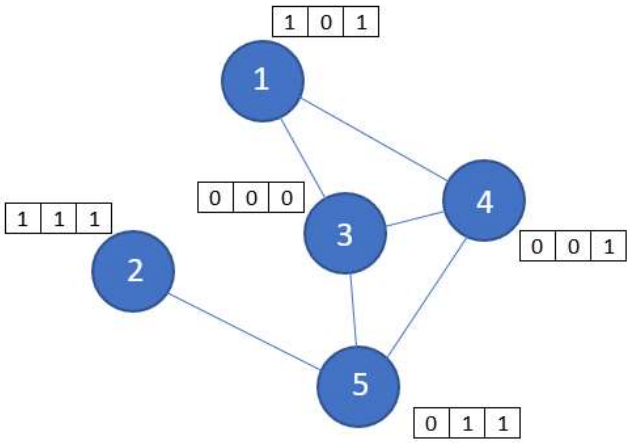


Figura 4.2. Estructura de grafo simple.

Antes de comenzar con la explicación de cada matriz, podemos asignar los valores a las dimensiones anteriormente mencionadas en base al grafo dado:

$$N = 5 \qquad M = 6 \qquad D = 3$$

- **Matriz de Adyacencia (A):** las uniones entre los distintos nodos del grafo quedan recogidas en la matriz de adyacencia, consiguiendo así estructurar las aristas que, aparentemente, en la representación del grafo no constan de orden alguno. La dimensión de la matriz A es de (NxN), siendo N el número de nodos del grafo, por lo que A será de dimensiones 5x5. El elemento A_{ij} de la matriz, siendo *'i'* y *'j'* índices referidos a los nodos numerados en la imagen del grafo superior, contendrá el valor '1' en el caso de que exista una arista desde el nodo *'i'* hasta el nodo *'j'*. Al tratarse de un grafo no direccionado, la matriz A será simétrica pues se supone un vínculo bidireccional entre los nodos conectados [32].

$A =$

	1	2	3	4	5
1	0	0	1	1	0
2	0	0	0	0	1
3	1	0	0	1	1
4	1	0	1	0	1
5	0	1	1	1	0

La diagonal de nuestra matriz de adyacencia estará compuesta por ceros, indicando que en los nodos no hay uniones con ellos mismos, caso que podría darse para indicar relaciones o interacciones consigo mismo.

- **Matriz de propiedades (X):** los distintos nodos del grafo suelen estar asociados a unos atributos que los caracterizan, pudiendo ser estos por ejemplo la edad, altura y peso de un individuo. En el grafo analizado podemos ver que el vector de propiedades de cada nodo está compuesto por variables binarias. Para formar la matriz de propiedades se agruparán los vectores de propiedades consiguiendo así una matriz de dimensiones (NxD), por lo que la dimensión de X será 5x3 para el grafo tratado. Se agruparán según el orden indicado por la numeración de los nodos.

$$X = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

La matriz de propiedades podrá variar su dimensión horizontal en el caso de que el número de propiedades de los nodos sea alterado durante el proceso de aprendizaje. Se denominará como 'Z' a la matriz de propiedades con un número de propiedades por nodo que podrá ser diferente al valor inicial.

4.2.3. Redes Neuronales Gráficas

Las Redes Neuronales Gráficas, o Graph Neural Networks (GNN) en inglés, es un enfoque del Deep Learning que permite ser aplicado directamente sobre gráficos. Las GNN aprovechan la dependencia de los nodos del grafo con sus vecinos en base a las aristas que los relacionan, consiguiendo inferir así un vector de características computando la información obtenida de los nodos vinculados a él.

Las Redes Neuronales Gráficas emplean las matrices de propiedades y de adyacencia para identificar los nodos vecinos de aquel elemento del que se desea obtener el vector de características aprendido, en el cual influirán las características y tipo de unión de los nodos conectados a él de forma directa. En base a las características inferidas en cada nodo y a la identificación de patrones que relacionan los vínculos entre los elementos del grafo, las GNN podrán aplicarse con distintas finalidades:

- **Clasificación del grafo**: la finalidad de dicha aplicación es clasificar el conjunto completo del grafo entre distintas categorías, siendo una de las aplicaciones más empleadas en el sector químico y biomédico con el objetivo de clasificar estructuras moleculares.
- **Clasificación de nodos**: el objetivo será determinar el vector de propiedades inferido en cada nodo en base a la información obtenida de las entidades vecinas. Se entrena empleando parte del grafo sin etiquetar, consiguiendo así que nuestro modelo aprenda las características dependientes de los nodos vecinos que determinarán la pertenencia del nodo a una cierta categoría. En el sector comercial se emplea por ejemplo con el objetivo de determinar si un usuario de nuestra red comprará o no un cierto producto, siendo etiquetado como potencial comprador o no.
- **Predicción de vínculos**: el algoritmo buscará relaciones entre los nodos vinculados entre sí con el objetivo de predecir nuevas aristas en el grafo. Su aplicación más extendida tiene lugar en el campo de las redes sociales en el cual en base a las características y amistades de un usuario, se realizarán recomendaciones de otros perfiles.

El mecanismo de intercambio de información entre nodos se denomina 'Paso de Mensajes', gracias al cual la información de los nodos vecinos es compartida con el fin de ser integrada junto al vector de características de nuestro nodo objetivo.

Para conseguir el vector de características del nodo serán necesarios dos pasos gracias a los cuales se podrá combinar la información de los nodos vecinos, mientras que, con el segundo paso, se actualizará el vector teniendo en cuenta el vector actual y la información aportada por los nodos en el primer paso. Los pasos a seguir serán los siguientes [33]:

- **Agregación**: durante el proceso de agregación se producirá la síntesis de información aportada por los nodos vecinos. Una vez hayan sido identificados los nodos vecinos y extraído el vector de características de cada uno, será necesario combinarlos para generar una representación adecuada de dichos valores. La fase de agregación permite emplear diversas estrategias tales como la media, suma, máximo o mínimo de los vectores de características de los vecinos, debiendo ser escogida la estrategia que permita obtener mayor rendimiento según la aplicación específica a tratar. Para observar la interacción e influencia de los nodos vecinos con respecto al nodo objetivo se marcará el nodo '2' de nuestro grafo como nodo objetivo y se agregará la información de la vecindad asociada [34].

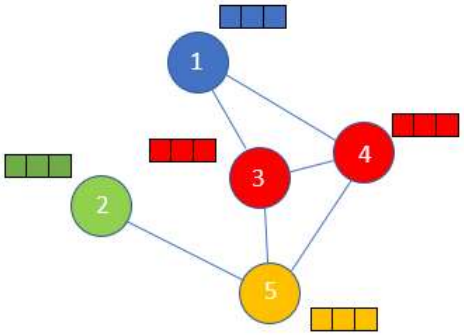


Figura 4.3. Estructura de grafo y vecindad.

En la Figura 4.3 se ha marcado el nodo objetivo en color verde mientras que los vecinos directos aparecen en color amarillo. El siguiente círculo de vecindad aparece representado en color rojo, permitiendo visualizar la inferencia del vector de características del nodo '2' en base a vecinos más alejados de este. Durante la fase de agregación, todos los nodos toman información de los nodos vecinos de forma simultánea, por lo que, tras una etapa de agregación, cada nodo tendrá un vector de características relacionado con la información de sus vecinos directos. Tras una segunda etapa de agregación, los nodos comenzarán a recibir información de sus nodos vecinos nuevamente, que, a su vez, ya recibieron durante la etapa anterior información de los nodos directamente conectados a ellos, haciendo que el nodo objetivo agregue información no solo de sus vecinos directos, sino de la vecindad de sus vecinos directos. Se representará en la Figura 4.4 la variación en los vectores de características de cada nodo según los vecinos que aportan información a este.

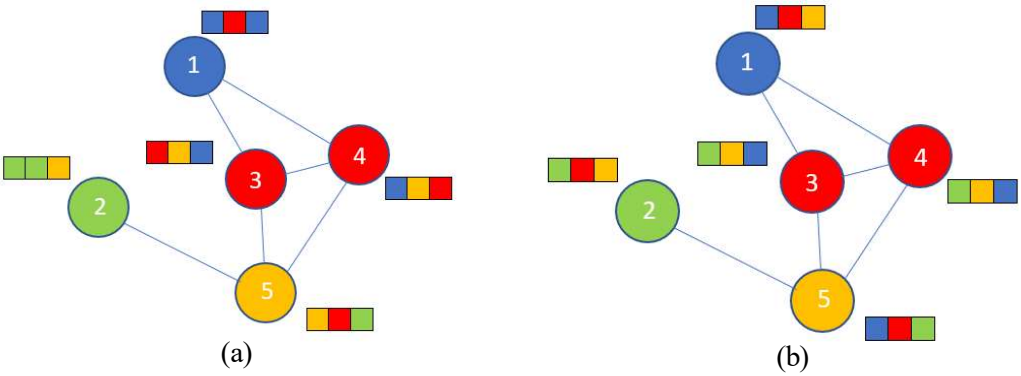


Figura 4.4 Grafo tras una fase de agregación (a) y grafo tras dos fases de agregación (b).

La influencia que ejerce cada elemento sobre el nodo objetivo puede verse gráficamente en la Figura 4.5 gracias al grafo de computación, representando la dependencia entre nodos en función de la cantidad de etapas de agregación empleadas.

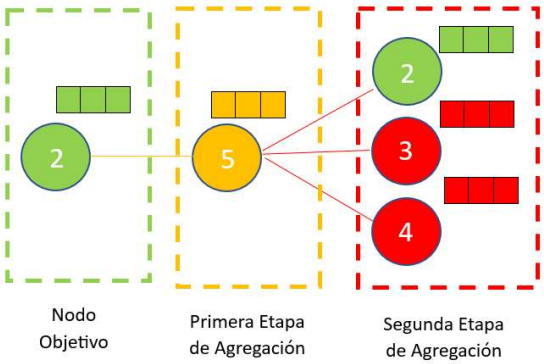


Figura 4.5. Grafo de computación en el nodo 2.

Cada uno de los nodos tiene asociado un grafo de computación, en base al cual se determinará el valor de su vector de características. La fase de generación del vector de propiedades final está relacionada con la actualización, siendo esta la etapa necesaria tras la agregación. La defínción matemática de esta fase en su forma general puede ser desarrollada en base a los siguientes elementos e índices que intervienen en la agregación:

$$Vagg^{(k)} = Fagg^{(k)}(\{h_v^{(k)}, \forall v \in N(u)\})$$

Donde:

- $h_v^{(k)}$: vector de características con índice 'i' de un nodo determinado en la etapa k de agregación.
- u: índice del nodo objetivo.
- k: etapa de agregación actual.
- $N(u)$: conjunto de índices asociados a los nodos de la vecindad directa del nodo u.
- $Fagg^{(k)}$: función aplicada en la etapa k de agregación.
- $Vagg^{(k)}$: vector de propiedades en la etapa k de agregación.

Algunas de las funciones de agregación más comunes (Fagg) aplicadas a los vectores de propiedades del conjunto de vecinos son las siguientes:

- Media: $$Vagg^{(k)} = \frac{\sum_{v \in N(u)} h_v^{(k)}}{|N(u)|}$$
- Suma: $$Vagg^{(k)} = \sum_{v \in N(u)} h_v^{(k)}$$
- Máximo: $$Vagg^{(k)} = \max_{v \in N(u)} (\{h_v^{(k)}\})$$
- Mínimo: $$Vagg^{(k)} = \min_{v \in N(u)} (\{h_v^{(k)}\})$$

- **Actualización**: en la fase de agregación se ha obtenido la representación vectorial de propiedades en base a la información de los elementos vinculados directamente al nodo objetivo. Se dispone de un vector de características inferido en base a la vecindad y un vector de características propio del nodo objetivo, por lo que será necesario combinar la información de ambos para conseguir el vector de propiedades final en el nodo escogido.

  Siguiendo los principios de funcionamiento de las redes neuronales clásicas, las GNN deben emplear una matriz de pesos W y un sesgo B que deberán ser ajustados durante el entrenamiento para poder emplear el modelo correctamente. El modelo de red neuronal gráfica puede constar de más de una capa, siendo implementadas en cada etapa tanto una fase de agregación como una fase de actualización. Cada capa tendrá su propia matriz de pesos y sesgo dependiente de la capa en la que sean aplicadas. La dimensión del vector de características resultante podrá ser diferente en cada capa.

  El vector de características que será utilizado como entrada en capas posteriores será el vector resultante $h_u^{(k+1)}$. Este vector dependerá de la ponderación establecida por el peso $W^{(k)}$ sobre el vector resultante de la fase de agregación $Vagg^{(k)}$ y de la ponderación del sesgo $B^{(k)}$ sobre el vector de características actual del nodo analizado $h_u^{(k)}$. De forma análoga al funcionamiento de las redes neuronales simples o multicapa, será necesario aplicar una función de activación σ que aporte no linealidad al modelo diseñado, pudiendo emplearse las mismas funciones de activación que fueron presentadas junto a las redes neuronales clásicas.

  Suponiendo que en la fase de agregación se emplea la función de media, obtendremos la siguiente expresión como vector de características generado en la capa 'k' [33]:

$$h_u^{(k+1)} = \sigma(W^{(k)} \frac{\sum_{v \in N(u)} h_v^{(k)}}{|N(u)|} + B^{(k)} h_u^{(k)})$$

  El uso de múltiples capas en cascada permite generar un vector de características dependiente de un mayor número de nodos pues la información se propaga a través de varios círculos de vecindad. El uso de un número elevado de capas puede dar lugar lugar a una disminución en la precisión de nuestro algoritmo tomando en cuenta parámetros de nodos que, en ciertas aplicaciones, no deberían ser tomados en cuenta. El fenómeno que puede dar lugar a un rendimiento nulo es conocido como oversmoothing en inglés, debido al cual la información de todos los nodos se ha propagado con una profundidad tal que el modelo no es capaz de identificar características propias de cada nodo, sino que todos serán tratados de forma similar.

  Este hecho puede ser apreciado en el caso de tener únicamente dos nodos conectados entre sí mediante una arista bidireccional. Si se produce un flujo de información a través de múltiples capas, y por tanto se producirán múltiples fases de agregación, los nodos perderán en cierto modo la identidad que los caracteriza. Es por ello por lo que deberá adecuarse la profundidad del modelo a la información tratada en la aplicación, basándonos en la dependencia con respecto a la información del resto de elementos que conforman el grafo. En la Figura 4.6 se muestra este fenómeno en un grafo binodal.

(a) (b)

Figura 4.6. Grafo de dos nodos inicial (a) y grafo tras ser procesado en red GNN muy profunda (b).

4.2.4. Capa de convolución gráfica

Para conseguir inferir los vectores de propiedades de cada nodo del grafo integrando las fases de agregación y actualización se empleará sobre cada nodo del grafo la operación de convolución gráfica. Esta operación se basa en las capas de convolución comúnmente aplicadas para detección de imágenes en las cuales una máscara, o kernel como se conoce en inglés, es aplicada sobre los distintos píxeles de la imagen combinando la información del píxel sobre el que se coloca la máscara y de su vecindad acotada por el tamaño de la máscara.

El motivo principal por el que la operación de convolución clásica no puede ser aplicada sobre los grafos es el espacio sobre el que se distribuye la información. Los nodos y la información de los grafos se encuentran en el espacio no euclídeo mientras que las imágenes se estructuran en el espacio euclídeo como matrices de píxels. El hecho de que el número de vínculos directos asociados a cada nodo varíe, no permitirá la aplicación de una máscara de dimensiones fijas sobre la totalidad del grafo. Es por ello que la convolución en grafos adapta la fusión de información independientemente de la cantidad de vectores de propiedades a tomar en cuenta [35].

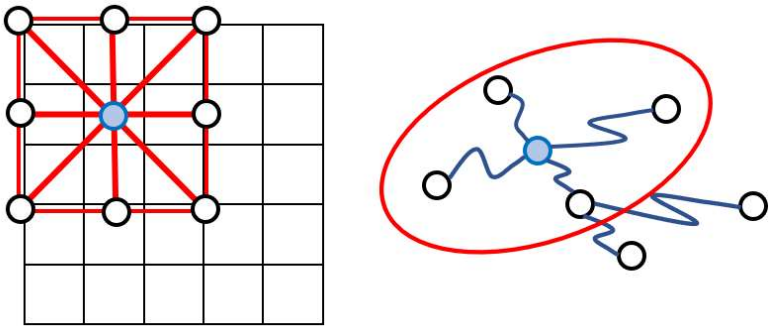
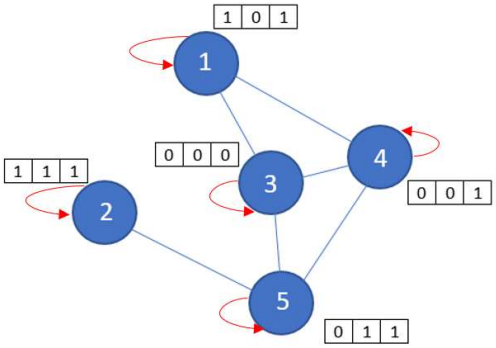


Figura 4.7. Comparación gráfica entre operación de convolución en imágenes y grafos.

Para mejorar el coste computacional del entrenamiento de los modelos basados en capas de convolución gráfica se empleará la vectorización. La representación del gráfico completo queda definida por la matriz de adyacencia A y la matriz de propiedades del grafo original X. De aquí en adelante la matriz de propiedades en la capa k será denominada como $H^{(k)}$, siendo la matriz de propiedades inicial $H^{(0)}$ equivalente a la matriz de propiedades del grafo original conocida como X.

La matriz de adyacencia permite obtener los vínculos directos de un nodo determinado, pudiendo identificar en base a la matriz A que vectores de propiedades dentro de la matriz deben ser tomados en cuenta para la determinación del nuevo vector de propiedades del nodo escogido. La capa de convolución gráfica permite obtener la matriz de características de todos los nodos del grafo simultáneamente, teniendo en cuenta para ello el vector del propio nodo del cual se busca obtener la nueva representación. El método seguido para ello consiste en incluir bucles propios en cada nodo siendo esto equivalente a sumar la matriz identidad de dimensiones NxN a la matriz A. Al agrupar la aportación del nodo objetivo y de los nodos vecinos en la misma matriz no será necesario emplear dos variables de peso separadas, siendo utilizada únicamente una matriz de peso $W^{(k)}$.



	1	2	3	4	5
1	1	0	1	1	0
2	0	1	0	0	1
3	1	0	1	1	1
4	1	0	1	1	1
5	0	1	1	1	1

Figura 4.8. Grafo y matriz de adyacencia tras aplicar bucles propios.

En la capa de convolución gráfica se emplea de forma general la función equivalente a la agregación basada en la media, pues se aplica un factor de normalización entre el número de vectores empleados durante la agregación y actualización. Una vez aplicada la media, el resultado será ponderado usando la matriz de pesos $W^{(k)}$. La dimensión de la matriz de peso puede variar entre distintas capas pues la dimensión del vector de características puede comprimirse a una menor dimensión o, por el contrario, se podrá expandir a una dimensión mayor.

Para expresar matemáticamente la operación computada por la capa de convolución gráfica, deberá ser preparada la matriz de adyacencia con los bucles propios, pasando a denominar dicha matriz como matriz de adyacencia aumentada $\hat{A}$.

$$\hat{A} = \mathrm{A} + \mathrm{I}$$

Para evitar problemas tales como el desvanecimiento o aumento exponencial del gradiente, se propone la normalización de la matriz $\hat{A}$ usando la matriz de grado D. Dicha matriz será diagonal y en cada elemento de la diagonal D_{ii} contendrá el número de vínculos directos que posee el nodo con índice 'i', siendo este valor la suma de cada fila de elementos de la matriz $\hat{A}$. La normalización dará lugar a la matriz de adyancencia normalizada $\tilde{A}$.

$$\tilde{A} = D^{-1}\hat{A}$$

La normalización simétrica permitirá controlar en mayor medida el valor de los gradientes, permitiendo mantener el valor de los autovalores de la matriz de adyacencia en un rango inferior a 1.

$$\tilde{A} = D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}$$

Las posiciones donde los valores de $\tilde{A}$ son distintos de cero indican el índice de los nodos cuyos vectores de propiedades deben usarse para el cálculo del nuevo vector. Una vez sean agregados los vectores de propiedades tras el producto entre $\tilde{A}$ y $H^{(l)}$ se multiplicarán por el vector de pesos $W^{(k)}$. Dicho vector será de dimensión (YxZ) donde Z indica la dimesión del vector de salida generado para cada nodo mientras que Y es la dimensión de los vectores de características que están siendo agregados. Es por ello que cuanto mayor sea el vector de propiedades resultante, mayor será el coste computacional de ajuste de pesos $W^{(k)}$.

La función de activación presentada en la fase de actualización debe mantenerse para aplicar no linealidad

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

De forma análoga al efecto presentado al agrupar distintas fases de agregación y actualización, los modelos formados por varias capas de convolución gráfica permiten inferir los vectores de propiedades a partir de la información de nodos a mayor distancia.

Aplicando esta tecnología al problema de clasificación de nodos sobre el cual se desarrollará la aplicación final, tendremos como salida de la última capa de convolución un vector para cada nodo a partir del cual se podrá predecir su clase o categoría. Para ello se emplearán funciones como Softmax o Linear.

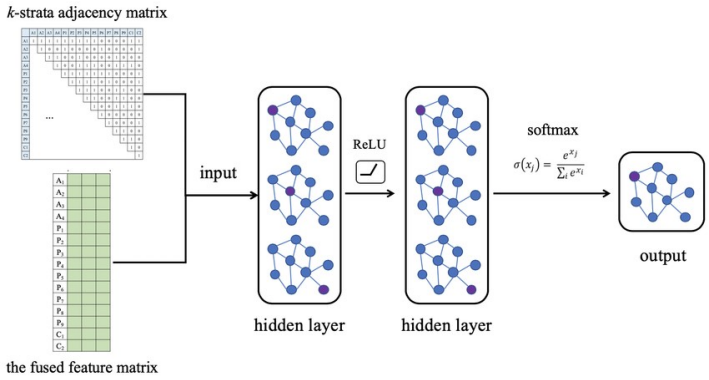


Figura 4.9. Clasificación de nodos basado en GNN multicapa [32].

4.2.5. Capa de atención en gráficos

De forma general, los vecinos de un nodo son tratados igualitariamente durante la fase de agregación, en la cual puede ser aplicada la función de media, tratándose de igual forma a todos los nodos a la hora de realizar la suma y la división entre el número de elementos. Sin embargo, al tratar con grafos cuyos nodos representan entidades con distintas propiedades dentro del conjunto, es común que no todos los nodos sean igual de influyentes por lo que algunos de ellos deberían ser ponderados en mayor medida, prestando así más "atención" a los nodos identificados como influyentes.

Cada arista tendrá asociada un coeficiente de atención α_{ij} indicando la importancia del nodo 'j' para el vector de características inferido del nodo 'i'. Los coeficientes α asociados a las aristas unidas directamente a un nodo sumarán uno.

El mensaje agregado dependerá de los distintos coeficientes haciendo que los nodos cuya influencia se determine como baja o nula tendrán una importancia proporcional a dicha categoría reduciendo su contribución. En el caso de que un nodo tenga contribución de dos vecinos, a pesar de que se aplique la operación de media entre los vectores de propiedades, el valor del vector agregado tenderá a estar mucho más próximo al vector asociado al nodo cuya arista indica que es necesaria una mayor atención. El vector de agregación al aplicar la media quedará definido bajo la siguiente expresión:

$$Vagg^{(k)} = \frac{\sum_{v \in N(u)} h_v^{(k)} \alpha_{uv}}{|N(u)|}$$

Suponiendo que se aplica la fase de agregación a los vecinos del nodo 1, cuyas aristas tendrán asociadas coeficientes de atención, quedarán representados de forma gráfica como se observa en la Figura 4.10.

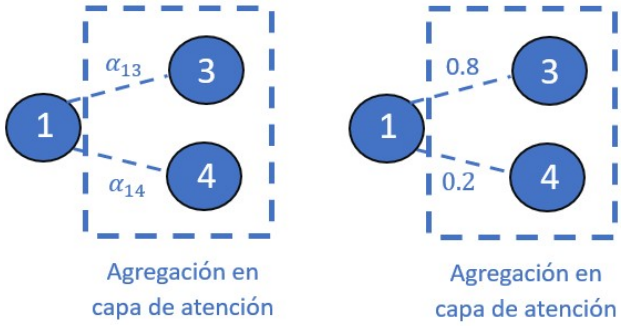


Figura 4.10. Agregación usando coeficientes de atención.

Para poder aplicar el mecanismo de atención a las predicciones del grafo completo será necesario determinar la aportación de cada nodo sobre sus vecinos, es decir, será necesario determinar los coeficientes de atención de cada arista. Se concatenarán los vectores de propiedades transformados del nodo actual y del nodo vecino, obtenidos tras multiplicar la matriz de pesos $W^{(k)}$ por el vector de propiedades de la capa actual $h_i^{(k)}$. Una vez concatenados los vectores transformados, se emplearán como entrada de una neurona simple cuyo peso se denomina $W_a^{(k)}$ y la función de activación asociada será una función LeakyReLU. Finalmente se aplicará la función Softmax para normalizar los coeficientes de atención, obteniendo así un coeficiente por cada arista existente entre cada pareja de nodos. La matriz de pesos $W_a^{(k)}$ se ajustará durante el entrenamiento, permitiendo obtener los coeficientes de atención apropiados [37].

$$\alpha_{uv} = \frac{\exp\left(LeakyReLU(W_a^{(k)}[W^{(k)}h_u^{(k)} \,||W^{(k)}h_v^{(k)}]\right))}{\sum_{v \in N(u)} \exp\left(LeakyReLU(W_a^{(k)}[W^{(k)}h_u^{(k)} \,||W^{(k)}h_v^{(k)}]\right))}$$

Estos coeficientes se integrarán en la matriz de adyacencia $\tilde{A}$, donde los valores normalizados serán sustituidos por los coeficientes de atención calculados para cada arista indicada en la matriz.

4.2.6. Entrenamiento de GNN para clasificación de nodos

El ajuste de los pesos empleados en cada capa de la Red Neuronal Gráfica se produce mediante un proceso de entrenamiento análogo al que tenía lugar durante el ajuste de los pesos en redes neuronales multicapa clásicas, pues se basa en una función de coste y en la apliación de el método de descenso de gradiente.

Las redes neuronales pueden ser entrenadas para predicción de vínculos entre nodos, clasificación del grafo completo o clasificación de nodos. Para la correcta implementación y entrenamiento de un modelo de clasificación de nodos será necesario que dichos elementos del grafo estén etiquetados correctamente con el fin de identificar predicciones incorrectas como métrica de error, permitiendo reajustar los pesos en cada época del entrenamiento.

Tal y como sucede con la red neuronal multicapa clásica, se deberá realizar el entrenamiento con una subdivisión del set completo de datos y, posteriormente, ser evaluado con el set de evaluación o *test set* como se conoce en inglés. El principal problema que se presenta durante el entrenamiento de las redes neuronales gráficas consiste en la fuerte relación y cohesión existente entre el conjunto de nodos, existiendo vínculos que complican la separación directa en set de entrenamiento y evaluación. Dado un grafo, la subdivisión entre los distintos sets pude realizarse de dos formas distintas, dando lugar a dos enfoques [38]:

- **Enfoque transductivo**: el grafo mantiene la totalidad de los vínculos y nodos unidos entre sí, conservando la información que cada nodo recibe de los nodos directamente conectados a él, sin embargo, se aplicarán máscaras binarias de entrenamiento y evaluación, marcando aquellos nodos que serán empleados para el entrenamiento del modelo. Un cierto porcentaje de los nodos del grafo no son visibles por el modelo durante el entrenamiento, por lo que no son tenidos en cuenta a la hora de ser clasificados ni evaluados por la función de coste durante el ajuste de los pesos al ser entrenada. A pesar de que no son tratados durante la etapa de entrenamiento, la información de sus vectores de propiedades sigue siendo visibles para los nodos vecinos, permitiendo así que las fases de agregación y actualización generen vectores de propiedades lo más fieles posibles a los que se obtendrían empleando la red del grafo completa.

  Una vez finalizado el entrenamiento, la máscara de evaluación enmascarará el grafo permitiendo que únicamente sean evaluados los nodos de test. Al igual que ocurre durante el entrenamiento, los vectores de propiedades de los nodos vinculados directamente a ellos siguen siendo visibles.

- **Enfoque inductivo**: el grafo es dividido en dos subgrafos aislados el uno del otro, tal y como ocurre con la subdivisión de set de datos para el entrenamiento y evaluación de arquitecturas clásicas multicapa. Las máscaras empleadas con el enfoque anterior no serán necesarias pues los grafos son totalmente independientes. Este método de entrenamiento es muy empleado con el objetivo de conseguir un modelo de clasificación de grafos completos. La creación de dos grafos independientes ayuda a que el modelo basado en Redes Neuronales Gráficas generalice en mayor medida, sin embargo, al realizar una subdivisión a partir de un grafo único podemos perder información de gran valor al eliminar vínculos o nodos clave. Los nodos de entrenamiento serán marcados en color rojo en los grafos.

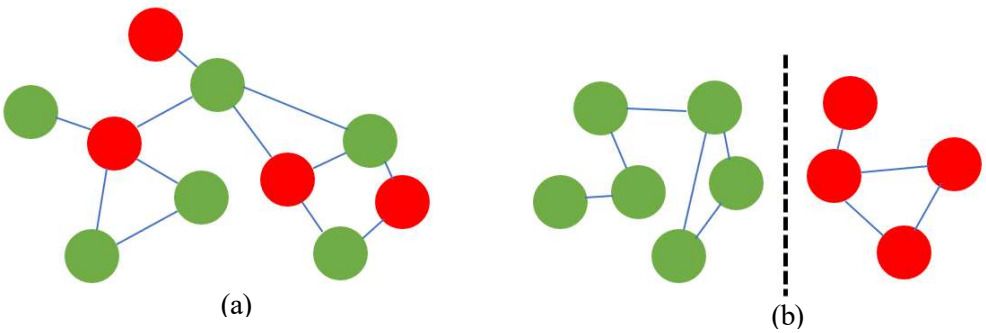


Figura 4.11. Entrenamiento transductivo (a) y entrenamiento inductivo (b).

La metodología de entrenamiento empleada para aplicaciones de clasificación de nodos es la trasductiva. Una vez definido el modelo de GNN e inicializados los pesos, serán ajustados empleando el algoritmo de descenso de gradiente basado en la derivada de la función de coste con respecto a los pesos de entrenamiento.

5 CLASIFICADOR DE ARTÍCULOS CIENTÍFICOS

5.1. Aplicación a desarrollar

La evaluación de la calidad de los artículos, así como la clasificación de estos según sus campos de estudio se ha convertido en una de las necesidades fundamentales en la comunidad científica. Los artículos o papers, como son denominados en inglés, se han convertido en uno de los principales recursos empleados por profesionales de cara al desarrollo de investigaciones, siendo de igual forma una herramienta fundamental para la divulgación científica.

La calidad de un artículo puede ser determinada tras un análisis exhaustivo de los contenidos de este y una posterior evaluación por un comité especializado en la materia. Las revistas clasifican previamente los artículos que son publicados, sin embargo, no es esta la única forma de determinar la calidad de un artículo. Una vez publicados los artículos, pueden ser citados y referenciados por otras publicaciones, siendo este hecho un indicador de que el contenido de ese artículo ha resultado útil para otro autor o investigador. Si un cierto artículo ha sido citado un gran número de veces, será indicador de que parte de la comunidad interesada en el campo de estudio en el que se encuadra el artículo utiliza las ideas recogidas en este, por lo que es posible que también sea útil para otros investigadores que buscan información relacionada con el ámbito de estudio del artículo. La aplicación propuesta tratará la generación de una métrica basada en el número de veces que ha sido citado un artículo, así como en la importancia de los artículos citados por el mismo y clasificarlos usando GNN.

El tipo de artículo o campo de estudio al que pertenece debe ser correctamente señalado, pues será uno de los datos, junto a la relevancia asociada, que más influirán a la hora de escoger o no un artículo. En ciertas ocasiones los artículos pueden tratar temas de forma transversal dificultando su asociación directa a un campo de estudio concreto, pudiendo influir en esta clasificación el vocabulario o palabras clave empleadas, así como las citas que se realizan en el mismo. Dada la complejidad del problema de clasificación de nodos y la dependencia de dicho problema con características propias y relaciones entre artículos, se propone el uso de una Red Neuronal Gráfica en la cual los nodos serán los distintos artículos y las aristas las citaciones entre ellos. Este modelo permite clasificar nuevos artículos dadas las palabras clave contenidas y la información de las veces que ha sido referenciado.

5.2 Entorno de Desarrollo

5.2.1 Google Colab

El desarrollo de la aplicación se realizará en la plataforma Google Colab, entorno de trabajo libre de Google. Dentro del entorno de desarrollo que nos brinda Google Colab podremos disponer de una GPU para el entrenamiento de los modelos. La posibilidad de ejecutar celdas de código de forma independiente convierte al entorno de Google Colab en el entorno de desarrollo y evaluación de modelos más versátil, pues la ejecución de una sola celda permite la modificación de los parámetros de forma directa.

Esta herramienta permite la programación en diversos lenguajes, incluyendo Python 3 que será el lenguaje empleado para el desarrollo del modelo basado en GNN para clasificación de los artículos científicos. A lo largo del código se emplean funciones de diversas librerías y frameworks que nos permiten estructurar y visualizar los datos además de poder realizar el entrenamiento del modelo basado en Redes Neuronales Gráficas.

5.2.2 PyTorch

PyTorch es una libería de código abierto enfocada al desarrollo de modelos de Machine Learning. Este marco de computación es empleado tanto para modelado como para entrenamiento de sistemas de visión artificial, procesamiento natural del lenguaje o modelos gráficos como las GNN. La vesión de Pytorch empleada para el desarrollo de la aplicación será la 2.0.0.

Este marco de programación fue lanzado en septiembre de 2016, siendo desarrollado por Facebook's AI Research lab (FAIR). El respaldo y actualización constante de sus características han convertido a PyTorch en uno de los *Frameworks* más importantes para el desarrollo de aplicaciones de Deep Learning. Conforme los avances tecnológicos y nuevos enfoques para distintas aplicaciones de Deep Learning surgían, PyTorch integraba los elementos, capas y optimizadores necesarios para el desarrollo de modelos innovadores. El gran interés por las redes neuronales gráficas dió lugar a la integración de capas y operaciones aplicables a grafos junto a distintas herramientas de gestión de información distribuida en gráficos. PyTorch Geometric ofrece las herramientas necesarias para tratar datos estructurados de forma irregular como ocurre con la información de los grafos tratados a lo largo de este proyecto.

5.2.3 Librerías complementarias

La visualización de los datos estructurados en grafos permite comprender con mayor claridad el problema tratado. Las operaciones con vectores y tensores requerirán librerías específicas que faciliten dicha tarea por lo que las funcionalidades necesarias serán incluidas importando las siguientes librerías:

- **Networkx**: librería que permite analizar los datos estructurados en grafos y representarlos gráficamente.
- **Matplotlib**: librería diseñada para la generación de gráficos de dos dimensiones a partir de datos en listas o arrays. Será de gran utilidad a la hora de representar histogramas de cara a analizar el set de datos empleado.
- **Scikit-learn**: librería empleada para el desarrollo de sistemas basados en Deep Learning y representación de resultados gráficamente.

5.3. Set de datos

La aplicación se desarrollará haciendo uso del set de datos ofrecido por PyTorch Geometric llamado CiteSeer el cual ofrece una serie de artículos científicos de 6 categorías diferentes. Dichos artículos son clasificados dentro del campo de los Agentes biológicos o químicos (BA), Inteligencia Artificial (AI), bases de datos (DB), recuperación de datos (IR), Machine Learning (ML) o Interacción humano-computador (HCI). El número total de artículos asciende hasta 3312. Gran parte de los artículos referencian otras publicaciones que les ayudan a transmitir o defender su idea o línea de trabajo, constando en este set de datos de un total de 4732 citaciones entre los artículos del set.

La clasificación de artículos en campos de estudio está estrechamente relacionada con el contenido tratado, así como del vocabulario empleado. La información de palabras clave se recoge en un diccionario formado por 3703 palabras para cada artículo, siendo indicado mediante unos o ceros si una determinada palabra es usada o no en la publicación.

Dado que las citas expresan el hecho de que un artículo hace referencia a otro, el sentido en el cual se ha producido esa citación queda indicado. A pesar de la estrecha relación que guardan las clases en las que son enmarcados los artículos, será muy común que algunos de ellos no sean referenciados o, que se aprecien secciones del grafo aisladas indicando la afinidad entre un cierto grupo de nodos que no es compartida con el resto.

La red formada por artículos y citaciones convierte este problema en un candidato idóneo para ser tratado desde el enfoque gráfico, permitiendo así aplicar modelos de Deep Learning basados en Redes Neuronales Gráficas. Los artículos científicos serán los nodos, etiquetados con valores entre cero y cinco según el campo de estudio al que pertenecen. Las aristas del grafo estarán direccionadas y serán representadas por las citaciones. Tal y como se ha expuesto, cada artículo tiene asociado un diccionario con palabras empleadas durante su desarrollo, permitiendo agrupar estas características como vector de propiedades de cada nodo [39].

Características set de datos CiteSeer	
Nodos	3312
Aristas	4732
Características	3703
Número de clases	6

La distribución de nodos y aristas permite representar de forma gráfica nuestro dataset, pudiendo visualizar en él los artículos y la relación existente entre ellos tal y como se presenta en la Figura 5.1.

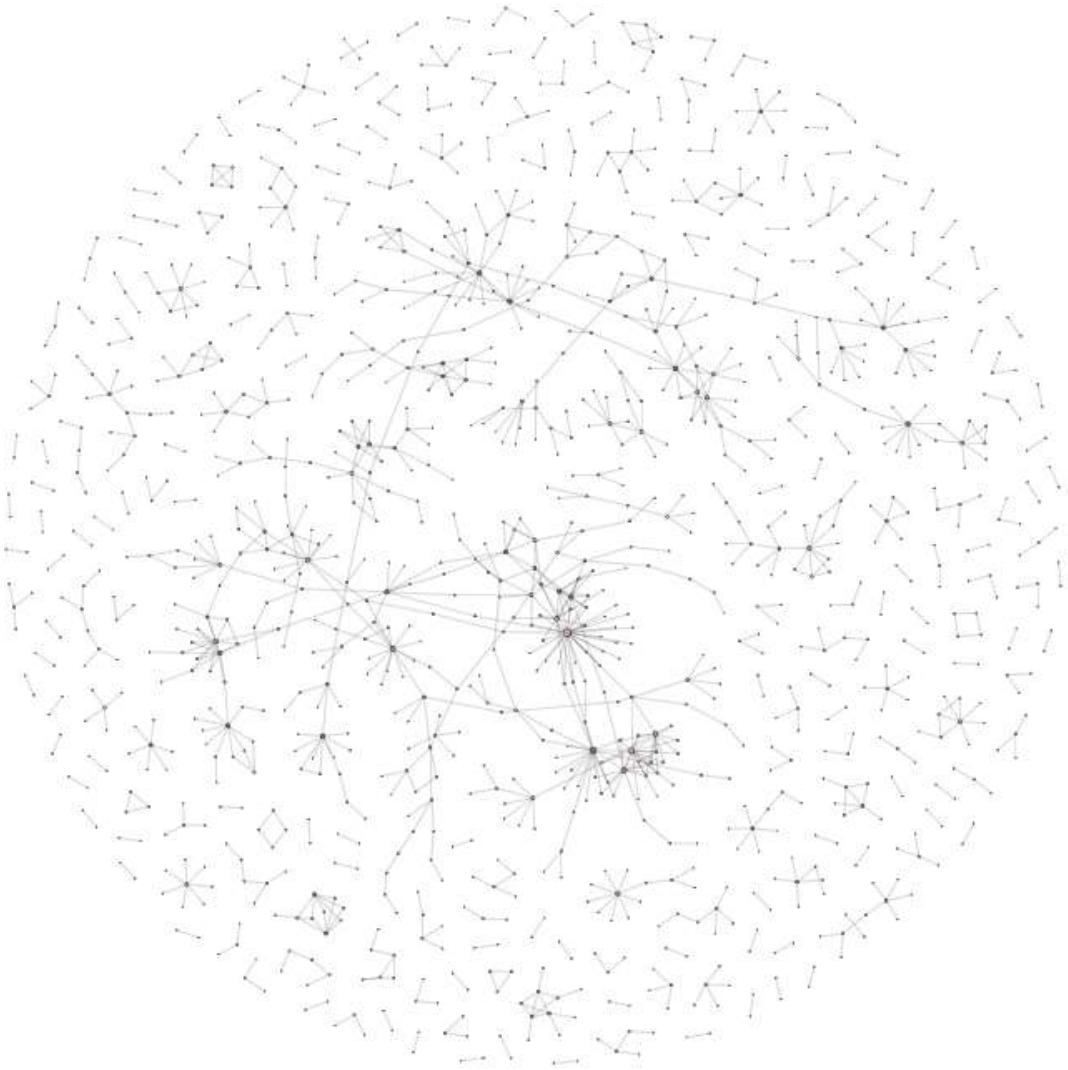


Figura 5.1. Representación gráfica del set de datos CiteSeer [33]

Para poder realizar un análisis de los artículos es necesario comprender la distribución seguida por los elementos del grafo pudiendo detectar problemas con la distribución entre las distintas clases, uso de palabras clave muy dispares entre los artículos del set o grandes diferencias en el número de veces que son citados los artículos en el grafo.

5.3.1 Distribución por clases

Realizando un análisis de los datos se puede evaluar la cantidad de artículos pertenecientes a cada clase, siendo en este ejemplo la categoría minoritaria la asociada a los artículos relacionados con agentes biológicos (BA). El resto de las clases guardan un cierto equilibrio, presentando sin embargo, una cierta dominancia el sub-set de artículos de recuperación de datos (IR). El hecho de entrenar la red para clasificación de artículos empleando un set con una categoría con un volumen notablemente inferior al resto de categorías podría afectar a la robustez del modelo de cara a la clasificación de artículos relativos a agente biológicos (BA) en este caso. La distribución queda recogida en la siguiente tabla.

Categoría	Número de artículos
BA (Cat. 0)	249
AI (Cat. 1)	590
DB (Cat. 2)	668
IR (Cat. 3)	701
ML (Cat. 4)	596
HCI (Cat. 5)	508

Junto a ello, aporta un gran interés de cara al desarrollo de la aplicación conocer la distribución de citaciones y palabras clave empleadas por el conjunto de publicaciones. Estos dos últimos análisis permiten una mejor visualización del histograma asociado y serán presentados a continuación tras mostrar la distribución de los artículos en cada categoría en la Figura 5.2.

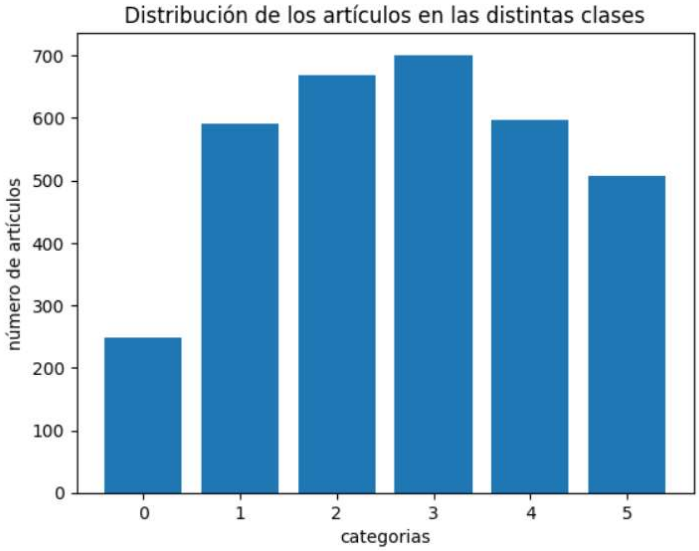
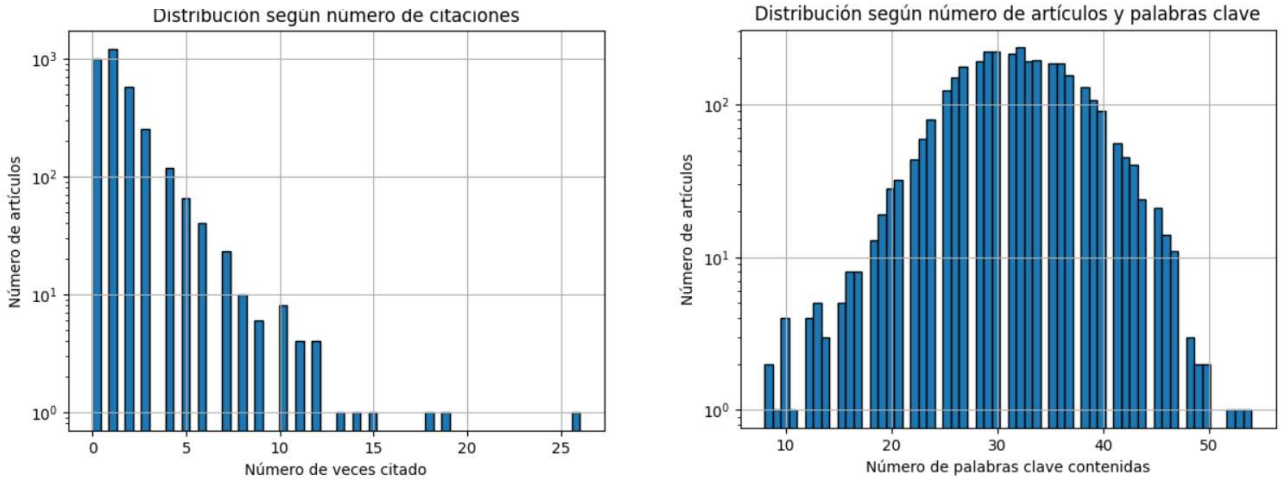


Figura 5.2. Distribución de publicaciones por categorías.



(a) (b)

Figura 5.3. Histograma asociado a número de citaciones (a) palabras clave asociadas (b).

De los histogramas presentados anteriormente en la Figura 5.3, se puede concluir que 2561 artículos, es decir, más de un 77% del conjunto, contiene entre 25 y 40 palabras clave. Entorno al 2% del set consta de un vector de características con 20 o menos palabras clave, permitiendo conocer así una aproximación del número de artículos que pueden presentar una caracterización menor, pues el diccionario empleado de 3703 palabras no engloba ciertos términos empleados por dichos artículos que añadirían valor a la hora de clasificar la publicación.

Con respecto al histograma de citaciones recibidas, se determina tras el análisis de los datos que las publicaciones que han sido citadas 10 o más veces solo conforman el 0.6% del set, es decir, un total de 22 artículos. Para determinar la métrica de calidad basada en citas se tendrán en cuenta el orden de los valores con los que trata el set de datos, así como valores máximos, mínimos y umbrales basados en el porcentaje de elementos cuyo valor es superior a la cota indicada. Dentro de la colección de publicaciones de la comunidad científica podemos encontrar artículos que han sido citados miles de veces, sin embargo, para la aplicación desarrollada se tendrán en cuenta los valores superiores de citas recogidos según el histograma. El set de datos CiteSeer tan solo engloba un número limitado y específico de artículos, pudiendo ser empleada la aplicación propuesta con distintos sets de publicaciones variando los nodos (artículos) y aristas (citas) que conforman el grafo, así como el vector de características asociado.

De cara a poder llevar a cabo el entrenamiento del modelo con un enfoque transductivo, será necesario definir las máscaras de entrenamiento que indicarán los nodos destinados a evaluación y aquellos destinados a entrenamiento. Los conjuntos de datos de PyTorch Geometric suelen integrar las máscaras de entrenamiento y test, sin embargo, el set de datos CiteSeer no incluye tales máscaras. El subconjunto de entrenamiento estará formado por el 75% de los nodos mientras que la evaluación se realizará con el 25% restante. Con el fin de poder replicar los resultados obtenidos por el modelo implementado, se ha realizado la división del set de forma alterna y siendo usada esta distribución en la totalidad de las pruebas y modelos testados. Las siguientes tablas recogen la distribución en cada conjunto según las categorías.

Distribución en subconjunto de entrenamiento por categorías	
BA (Cat. 0)	184 (7.41%)
AI (Cat. 1)	454 (18.28%)
DB (Cat. 2)	498 (20.05%)
IR (Cat. 3)	517 (20.81%)
ML (Cat. 4)	441 (17.75%)
HCI (Cat. 5)	390 (15.70%)

Distribución en subconjunto de evaluación por categorías	
BA (Cat. 0)	65 (7.85%)
AI (Cat. 1)	136 (16.42%)
DB (Cat. 2)	170 (20.53%)
IR (Cat. 3)	184 (22.22%)
ML (Cat. 4)	155 (18.72%)
HCI (Cat. 5)	118 (14.25%)

5.4. Diseño del modelo basado en GNN

5.4.1. Modelo 1: GCN

El primer modelo implementado para la clasificación de los artículos estará basado en capas de convolución gráfica. Lo modelos de convolución gráfica constituyen la base de las Redes Neuronales Gráficas, permitiendo lograr una alta precisión con un coste computacional y tiempo de ejecución mínimo. Tanto el coste computacional del modelo como su rendimiento estarán directamente relacionados con la elección de los hiperparámetros, siendo algunos de ellos la dimensión de los vectores de propiedades generados por cada capa, la profundidad del modelo, la tasa de aprendizaje o la función de coste empleada. El número de parámetros que deben ser ajustados durante el entrenamiento afectará a la duración del entrenamiento y a la precisión final obtenida al ser evaluada para clasificar los artículos con los que no ha sido entrenada. La elección de los hiperparámetros debe realizarse de forma minuciosa, basándonos en las reglas matemáticas y de comportamiento que modelan y representan cada uno de estos parámetros.

Una vez planteado el modelo final para la clasificación de artículos, analizaremos el comportamiento de otros modelos, apreciando la influencia que tiene la variación de cada hiperparámetro sobre el resultado final del modelo.

A pesar del gran número de dimensiones que posee el vector de características y aportación recibida por parte de los elementos vecinos, podremos representar gráficamente la distribución de los nodos empleando el modelo sin entrenar y apreciando la agrupación de los distintos nodos tras el entrenamiento.

Haciendo uso de las herramientas de visualización que nos ofrece Matplotlib podemos observar la Figura 5.4 generada de forma previa al entrenamiento.

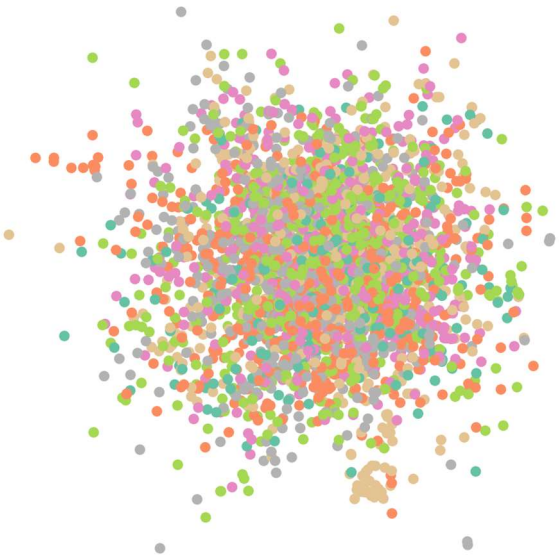


Figura 5.4. representación de la agrupación de nodos en el espacio bidimensional

En base a los vectores de propiedades de cada nodo, se representarán en la misma ubicación del espacio bidimensional aquellos nodos cuyas características inferidas indiquen relación entre ellos. Al disponer en este momento de un modelo sin entrenar, la representación obtenida no agrupa de forma correcta las publicaciones que representan los nodos pues los pesos inicializados de forma previa al entrenamiento no permiten inferir los vectores de características adecuados de cada nodo.

Los hiperparámetros seleccionados se presentan a continuación junto a la justificación que nos indicaría que los resultados logrados con dicha selección de hiperparámetros serían buenos. Además, la calidad del modelo podrá ser comparada con resultados recogidos en la literatura donde se emplean algoritmos de clasificación con el set de datos de CiteSeer.

- **Profundidad del modelo**: el número de capas de convolución empleadas conlleva el mismo número de fases de agregación y actualización que capas sean implementadas. La información de nodos más lejanos será empleada para inferir el vector de características, pudiendo incurrir en fallos o pérdidas de precisión si empleamos un modelo muy profundo en una aplicación en la que la dependencia real se establece con los vínculos directos. Dado que los artículos científicos contienen información referente a un cierto campo, conocer los artículos que estos citan también aportará valor añadido a la hora de su categorización. Para conseguir incluir en la predicción la información de los vínculos directos, tan solo será necesario incluir una capa de convolución que implemente una fase de agregación y actualización.
- **Dimensiones del vector de características generado**: cada capa de convolución gráfica generará un vector de características inferido el cual podrá mantener la dimensión del vector de características previo, aumentar o disminuir dicha dimensión. El hecho de mantener la dimensión del vector de características supone realizar la actualización sobre las mismas variables, haciendo que el riesgo de sobreajuste sea mayor, pues el modelo tenderá a ajustarse a las propiedades exactas de los nodos usados para el entrenamiento. El aumento de dimensión de este vector favorecerá en mayor medida el efecto explicado con anterioridad al mantener la dimensión del vector de propiedades. En el caso de disminuir las dimensiones de este en exceso, el modelo tenderá a generalizar de forma excesiva, pues la clasificación se basará en un menor número de características. El principal problema de implementar una transformación del vector de propiedades para reducir el número de elementos que contiene reside en el riesgo de que se produzca subajuste. El tamaño de vector de propiedades generado por nuestra capa de convolución será definido como 500. Dicha elección está relacionada con la regla aplicable a modelos de clasificación con menos de 1000 categorías posibles, siendo esta dimensión un valor estándar el cual permite generalizar evitando el subajuste que se produciría con vectores de menores dimensiones.
- **Duración del entrenamiento**: el número de épocas debe ser seleccionado teniendo en cuenta los resultados de la función de coste sobre el conjunto de entrenamiento, pudiendo detener el entrenamiento al lograr valores de precisión en el set de entrenamiento de entorno al 95%, pues a partir de ese punto las redes neuronales gráficas podrían verse afectadas por el sobreajuste al acercarse al 100%. Es por ello por lo que para el modelo gráfico de convolución unicapa se obtiene un valor de precisión aceptable en el set de evaluación entre las 30 y 40 épocas de entrenamiento. El modelo escogido será entrenado durante 35 épocas.
- **Función de coste**: la función de coste más comunmente empleada es la función de entropía cruzada ya que se corresponde estadísticamente con el logaritmo negativo de la función de verosimilitud asociada a la función Softmax. Se relaciona con la probabilidad asociada a la pertenencia a una cierta distribución, siendo una de las funciones más empleadas para aplicaciones de clasificación. [34].
- **Optimizador:** Para facilitar el entrenamiento se aplicarán técnicas de optimización conocidas como optimizador Adam. El optimizador empleado modificará durante el entrenamiento el paso de aprendizaje ajustándose según la dirección de descenso de gradiente facilitando alcanzar el mínimo de la función de coste de forma más eficiente. Uno de los principales cambios aportados por el optimizador de Adam consiste en el ajuste de pesos tras cada prueba realizada a cada elemento del conjunto de entrenamiento. El valor inicializado de paso de entrenamiento será de 0.001.

La capa de salida del modelo no será una capa de convolución gráfica, sino una capa lineal con vector de salida de 6 dimensiones para implementar la clasificación entre las diferentes categorías.

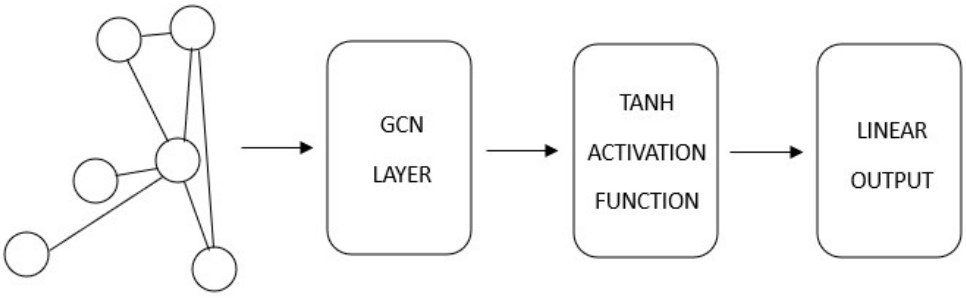


Figura 5.5. Modelo basado en GCN para clasificación de artículos.

Tras el entrenamiento del modelo planteado se obtienen las siguientes curvas y gráficas referentes a la función de coste y precisión alcanzada por el modelo.

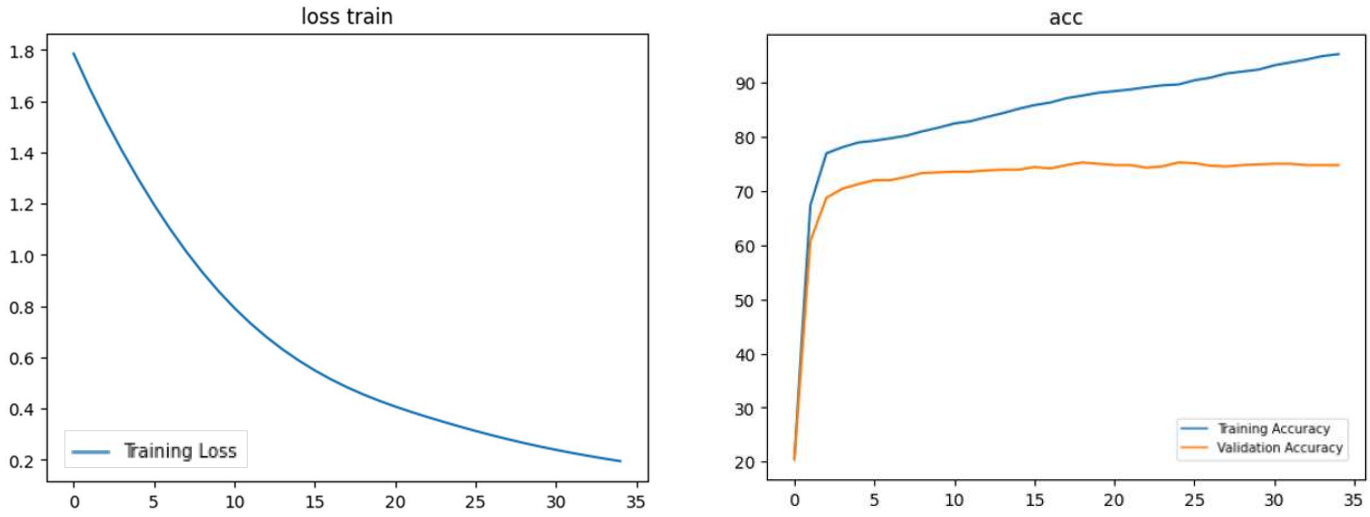


Figura 5.6. Resultados del entrenamiento de modelo basado en GCN.

Modelo evaluado	Precisión entrenamiento	Precisión evaluación
GCN (1 capa)	95.20%	74.75%

Se comprueba que con el modelo basado en redes convolucionales gráficas con una sola capa de convolución se alcanza un valor de precisión en la prueba de evaluación superior al 74%. El rendimiento obtenido alcanza valores de precisión muy próximos a los recogidos en la literatura relacionados con clasificación empleando el set de datos CiteSeer, los cuales oscilan entorno al 74% de precisión empleando modelos de convolución gráfica [35]. Una vez entrenado el modelo puede representarse de nuevo la distribución de los nodos en el espacio bidimensional, siendo situados en base a la similitud entre ellos atendiendo a los vectores de características inferidos por el modelo de Red Neuronal Gráfica.

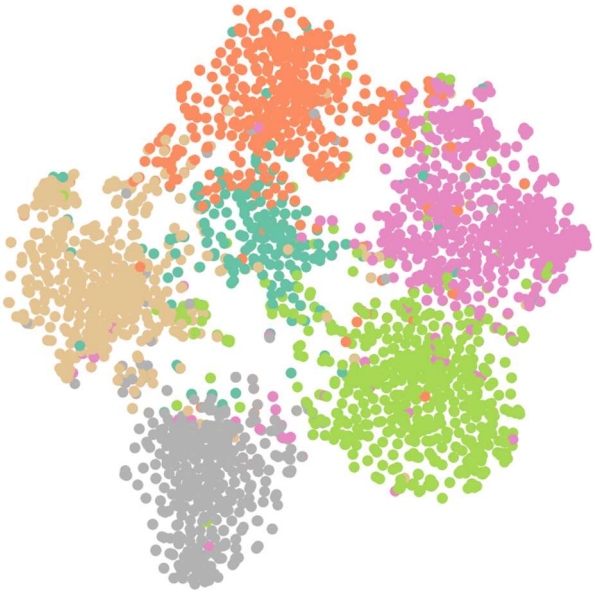


Figura 5.7. agrupación de nodos en el espacio bidimensional tras entrenamiento de GCN.

De cara al análisis de posibles anomalías con ciertas categorías se agruparán los 209 artículos mal clasificados por categoría:

Relación de artículos mal clasificados (GCN)	
BA (Cat. 0)	42 (20.01%)
AI (Cat. 1)	26 (12.44%)
DB (Cat. 2)	37 (17.70%)
IR (Cat. 3)	44 (21.05%)
ML (Cat. 4)	39 (18.66%)
HCI (Cat. 5)	21(10.04%)

El hecho de que los artículos de la categoría 0 representen el 20% de las publicaciones mal clasificadas cuando tan solo representan el 7.85% dentro del set de evaluación nos indica una posible anomalía con respecto a los elementos de otras categorías.

Tras analizar el conjunto completo de elementos de la categoría 0 (BA), se identifica el número medio de veces que son citados, siendo este valor inferior a 1 cita de media, en concreto, la media de citaciones es de 0.9036 referencias. Este hecho acompañado del desbalance existente entre la categoría cero y el resto motivará que parte de los artículos no puedan ser caracterizados correctamente en base a la información de publicaciones vecinas en el grafo. El número medio de palabras clave en los artículos de la categoría 0 (BA) es de 31.76, encontrándose dentro de la media del resto de categorías por lo que la falta de información referente al vocabulario empleado no se identifica como un potencial problema.

5.4.2. Análisis de error ante variación de hiperparámetros

Resulta de gran interés analizar si la profundidad de la red influye en el resultado, pudiendo analizar de igual forma la influencia del tamaño de los vectores de características generados por la capa de convolución gráfica, haciendo especial incidencia en el aumento de coste computaciónal que acarrea.

Se entrenarán 3 modelos adicionales con 2, 4 y 6 capas de convolución permitiendo analizar la evolución del rendimiento conforme la red adquiere mayor profundidad. Se presentará la precisión obtenida con el modelo, así como el error relativo cometido con respecto a la precisión del modelo principal formado por una sola capa de convolución gráfica. Serán entrenadas durante el mismo número de épocas y con el mismo paso de aprendizaje que el modelo anterior para poder realizar una comparación clara entre los distintos modelos.

Modelo evaluado	Precisión evaluación	Error relativo precisión de evaluación
GCN (1 capas)	**74.75%**	**0%**
GCN (2 capas)	71.73%	4.04%
GCN (4 capas)	71.65%	4.14%
GCN (6 capas)	71.01%	5.00%

Se puede representar como afectaría a la distribución espacial en el plano bidimesional el uso de un modelo compuesto por 12 capas convolucionales, pudiéndose apreciar en la Figura 5.8 una separación más difusa en algunas clases incluso produciéndose fragmentación en varias sub áreas.

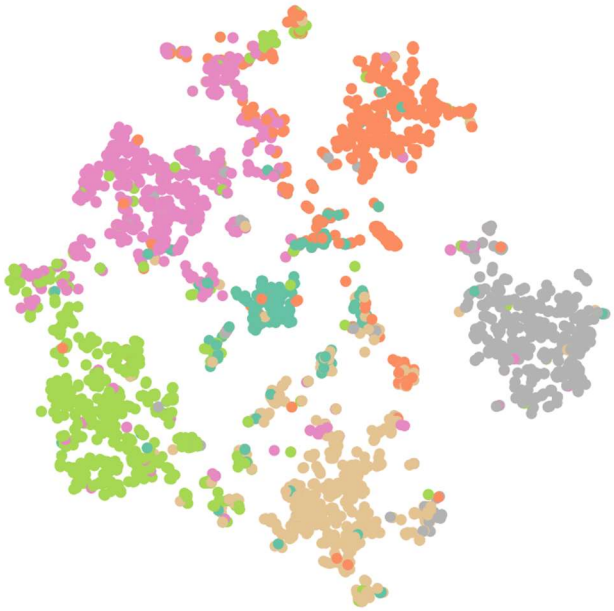


Figura 5.8. agrupación de nodos en el espacio bidimensional con modelo multicapa.

La dimensión de los vectores de propiedades inferidos o embeddings como se denominan en inglés es un hiperparámetro que permitirá definir si se mantendrán, se aumentarán o se disminuirán el número de propiedades asociadas a cada nodo. La elección de esta dimensión como un valor elevado puede llevar al modelo a sufrir sobreajuste mientras que un valor reducido causaría el subajuste del modelo. Para comprobar la variación producida se comparará el modelo principal entrenado con dimensión de vectores generados igual a 500 con otros dos modelos unicapa de las mismas características e idéntico entrenamiento que el modelo principal cuyas dimensiones de embedding serán 50 y 5000, siendo denominados en la tabla inferior como GCNV50 y GCNV5000 respectivamente.

Modelo evaluado	Precisión entrenamiento	Precisión evaluación	Error relativo precisión de evaluación
GCNV50	85.54%	73.55%	1.60%
GCNV500	**95.20%**	**74.75%**	**0%**
GCNV5000	99.79%	72.58%	2.90%

Observando los datos obtenidos tras realizar el entrenamiento de los modelos basados en capas convolucionales gráficas durante 35 etapas, puede verse como afectarán al modelo los problemas de sobreajuste y subajuste. Tanto el modelo cuyos vectores de características inferidos tienen dimensión 50 como el modelo cuyos embeddings son de dimensión 5000, logran una precisión menor a la obtenida previamente con el modelo que emplea vectores de características de 500 componentes.

Mientras que el modelo GCNV50 llega entorno al 85% de precisión en el set de entrenamiento, el modelo GCNV5000 consigue alcanzar una precisión cercana al 100% en tan solo 35 epocas de entrenamiento. A pesar del alto valor de precisión en el set de entrenamiento, es el modelo con peor desempeño al ser empleado para clasificación de artículos pertenecientes al conjunto de evaluación.

En cuanto a tiempo de ejecución y coste computacional, cuanto menor sea el vector de características generado, menor será la cantidad de pesos que deben ser ajustados durante el entrenamiento. El entrenamiento del modelo GCNV500 supone aproximadamente 1.5 veces el tiempo necesario para el entrenmeiento del modelo GCNV50.

La generación del vector de características se realiza en base al vector inicial asociado a cada nodo, sin embargo, la capacidad de determinar la clase a la que pertenece un cierto artículo está asociada con la importancia de las palabras clave empleadas por el artículo a clasificar y por las palabras empleadas en las publicaciones que este cita. Al tratar imágenes como entradas de un modelo de detección o clasificación, resultará complicado determinar la contribución o importancia de un píxel determinado a la clasificación, sin embargo, los métodos de explicabilidad permiten determinar la importancia de las palabras clave en la predicción realizada por el modelo. En nuestro caso podremos determinar que palabras clave empleadas por los artículos que intervienen en la clasificación aportan más información. La importancia se representará en el rango de 0 a 1, indicando el valor cero una nula importancia en la clasificación del artículo seleccionado o que la información de dicha palabra no aporta información. Si por el contrario una palabra clave tiene asociado el valor uno o cercano a este, será indicativo de que tendrá un peso importante en la clasificación. Emplearemos el análisis de importancia de las palabras para apreciar la pérdida de generalización y dependencia del modelo con un menor número de parámetros, acarreando problemas similares a los producidos al emplear una dimesión de embedding pequeña. Se analizará la importancia de las palabras clave a la hora de clasificar el nodo 962. Dicho nodo cita a dos publicaciones más, siendo estas la publicación número 2 y número 3151, por lo que las palabras evaluadas tendrán que ser aquellas que se empleen en alguno de los tres artículos mencionados. El modelo unicapa basado en capas de convolución gráfica entrenado durante 35 etapas muestra los siguientes datos de importancia:

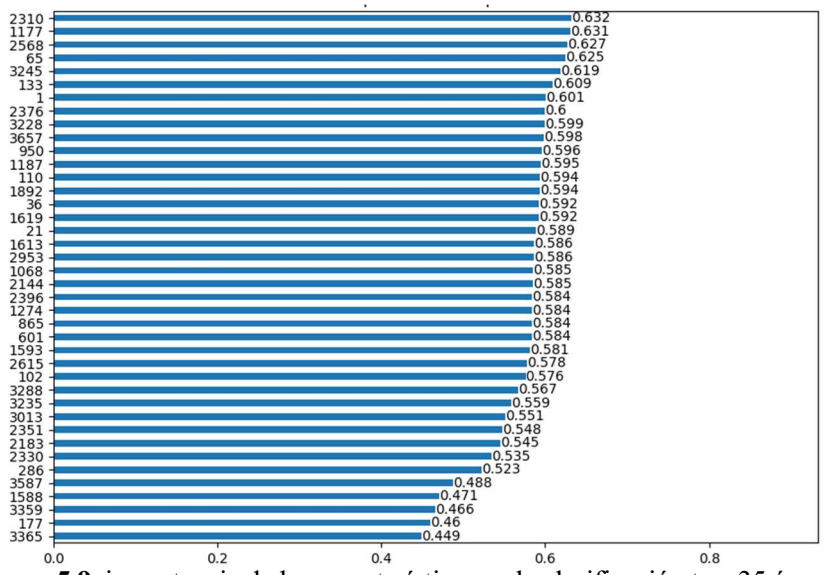


Figura 5.9. importancia de las características en la clasificación tras 35 épocas.

Mientras que, si se entrena el modelo durante un número muy elevado de épocas como pueden ser 200, apreciaremos como el modelo genera una fuerte dependencia de ciertas palabras a las que se les atribuye valores de importancia superiores a 0.8, afectando a la capacidad de generalización del modelo.

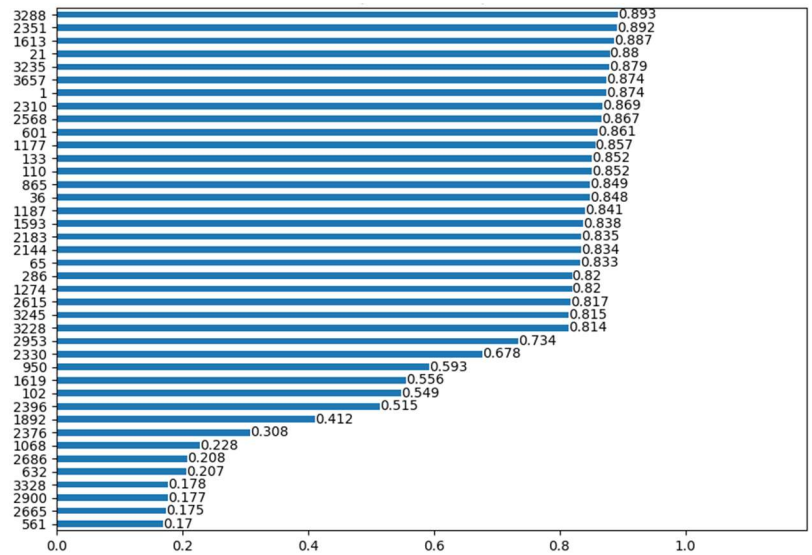


Figura 5.10. importancia de las características en la clasificación tras 200 épocas.

5.4.3. Modelo 2: GAT

Dado que las aristas en el grafo no están ponderadas de ninguna forma, es decir, solamente son caracterizadas por una variable booleana que indica su existencia o no, no podrá identificarse que vínculos son más importantes para la clasificación del nodo. Para conseguir que nuestro modelo tenga en cuenta la importancia de un artículo vecino frente a otro, implementaremos técnicas de atención, obteniendo lo que se conoce como Red Gráfica de Atención.

En un principio la sustitución de las capas de convolución gráfica por las capas de atención no modificará las operaciones básicas relacionadas con la agregación y actualización, sino que, durante la fase de agregación, la contribución de cada vector de propiedades dependerá del coeficiente de atención generado por la capa de atención gráfica.

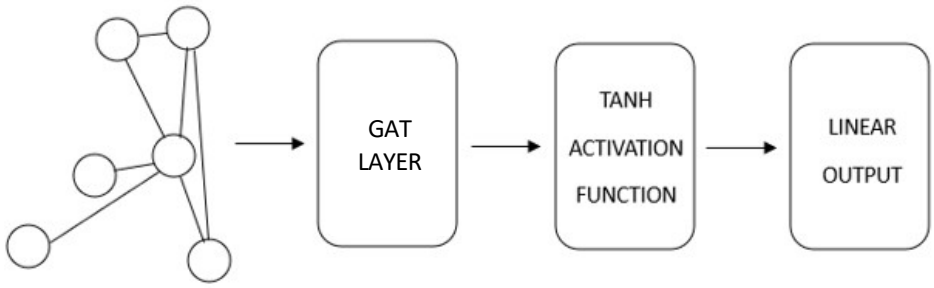


Figura 5.11. Modelo basado en GAT para clasificación de artículos.

Los resultados obtenidos tras el entrenamiento del modelo en 60 épocas se ven reflejados en las siguientes gráficas recogidas por la figura 5.12:

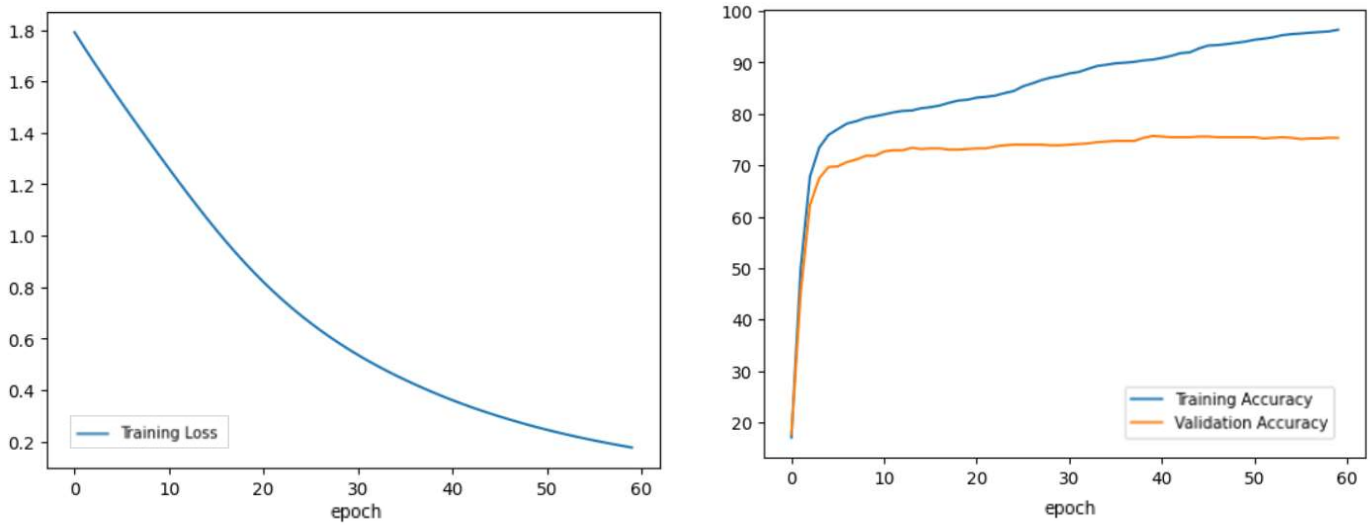


Figura 5.12. Resultados del entrenamiento de modelo basado en GAT.

Modelo evaluado	Precisión entrenamiento	Precisión evaluación
GAT (1 capa)	96.41%	75.60%

Consiguiendo con el modelo entrenado un rendimiento similar a los resultados recogidos en la literatura en aplicaciones de clasificación de nodos con el set de datos empleados.

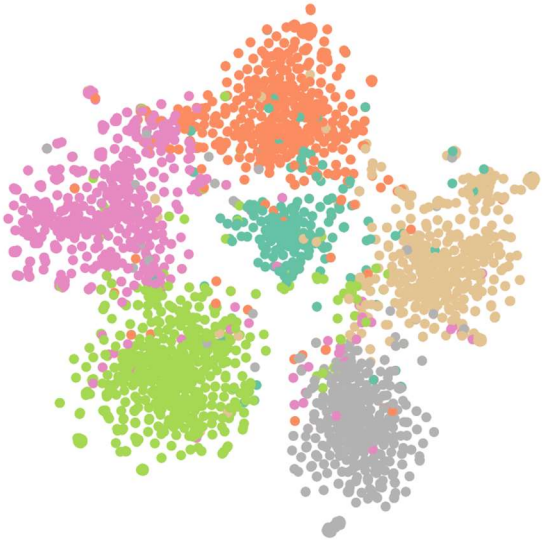


Figura 5.13. agrupación de nodos en el espacio bidimensional tras entrenamiento de GAT

La precisión lograda sobre el conjunto de evaluación con el modelo basado en capas de atención supera la precisión lograda con el modelo de convolución gráfica.

La relación de artículos mal clasificados por categoría se distribuye de la siguiente forma:

Relación de artículos mal clasificados (GAT)	
BA (Cat. 0)	42 (20.19%)
AI (Cat. 1)	29 (13.94%)
DB (Cat. 2)	40 (19.23%)
IR (Cat. 3)	38 (18.27%)
ML (Cat. 4)	33 (15.87%)
HCI (Cat. 5)	26 (12.50%)

De forma análoga al análisis realizado con el modelo basado en capas convolucionales, se buscará comprobar la variación en la precisión que se produce al implementar un modelo con mayor número de capas de atención. A pesar de que durante el entrenamiento irá determinando los pesos asociados a cada arista, ponderando en menor medida aquellos cuya información resulte más irrelevante. Si se cumple la hipótesis de que de cara a la clasificación basándonos en citaciones solo importan las referencias directas, apreciaremos un descenso en la precisión conforme se entrene un modelo con mayor profundidad, analizando los casos de 2,4 y 6 capas.

Modelo evaluado	Precisión evaluación	Error relativo precisión
GAT (1 capas)	**75.60%**	**0%**
GAT (2 capas)	73.67%	2.55%
GAT (4 capas)	71.98%	4.78%
GAT (6 capas)	71.49%	5.43%

5.5. Clasificación relevancia científica

Una vez categorizados en las distintas clases, se tendrán en cuenta los datos asociados al número de citaciones para determinar la relevancia de dicho artículo. Para etiquetar el set de datos, que hasta el momento consta de etiquetas referidas a la categoría de dicha publicación, se generará una métrica dependiente de la cantidad de veces que un artículo ha sido citado, así como de la cantidad de veces que han sido citados los artículos que se referencían en él.

Se determina el número de veces que ha sido citado y la suma de citaciones de los artículos que referencia, ponderando un 70% el primer valor y un 30% el segundo. Una vez obtenidos los datos de la métrica, se normalizarán para obtener valores entre 0 y 1, para a continuación, establecer un umbral a partir del cual se consideren de alta relevancia o de baja relevancia. El vector de características será sustituido por un tensor que almacene la métrica ya que en esta aplicación el uso del diccionario con palabras clave no será empleado.

La métrica se define como:

$$Q_u = 0.7R_u + 0.3\sum_{v\in N(u)} R_v$$

Donde:

- $Q_u = métrica\ de\ calidad\ asociada\ al\ nodo\ analizado\ u$
- $R_u = número\ de\ veces\ que\ ha\ sido\ referenciado\ el\ nodo\ analizado\ u$
- $R_v = número\ de\ veces\ que\ ha\ sido\ referenciado\ el\ nodo\ vecino\ v$
- $N(u) = conjunto\ de\ vecinos\ de\ u$

Para aplicar la normalización a cada valor de la métrica aplicando la siguiente fórmula:

$$Q'_u = \frac{Q_u - \max(Q)}{\max(Q) - \min(Q)}$$

Una vez normalizados podemos representar el histograma que recoge la distribución relativa a la métrica generada en la Figura 5.14:

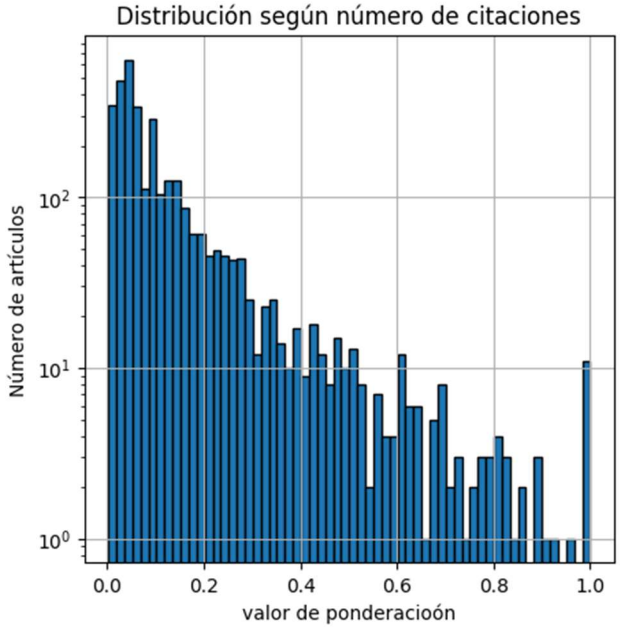
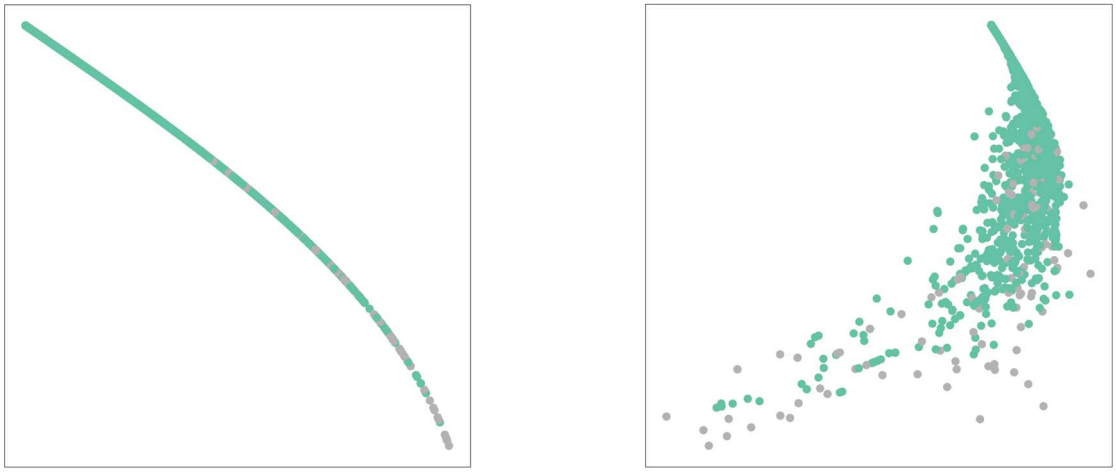


Figura 5.14. distribución valores de métrica de relevancia

Dada la baja cantidad de citaciones que se recogen para la gran mayoría de publicaciones en el set, es previsible obtener un porcentaje muy bajo de publicaciones clasificadas como 'contenido de alto interés'. Tomando el valor umbral en 0.15 agrupando aproximadamente un cuarto de los artículos como artículos de relevancia obtenemos la siguiente distribución en el set de datos:

Distribución de artículos por relevancia	
Artículos de alta relevancia	762 (23%)
Artículos de baja relevancia	2550 (77%)

El objetivo será probar una red neuronal gráfica con un set en el que las clases están desbalanceadas en gran medida. Se representa a continuación en la Figura 5.15 la distribución inicial en el espacio bidimensional al emplear una red neuronal sin entrenar compuesta por una sola capa de convolución y junto a ella la representación obtenida al tomar un modelo de dos capas de convolución gráfica.



(a) (b)

Figura 5.15. agrupación de nodos en el espacio bidimensional con 1 (a) y 2 capas (b) de convolución.

Distribución en subconjunto de entrenamiento relevancia	
Baja relevancia	1922 (77.38%)
Alta relevancia	562 (22.62%)

Distribución en subconjunto de evaluación por relevancia	
Baja relevancia	628 (75.85%)
Alta relevancia	200 (24.15%)

Dado que el vector de características para esta aplicación solo contendrá el valor de la métrica y esta solo está relacionada con el valor de los vecinos directos, apreciamos como al emplear un mayor número de capas, o lo que es equivalente, realizar la predicción tomando un mayor número de nodos, se pierde la linealidad que caracteriza este problema y los vectores de características asociados.

El enfoque del problema utiliza información agrupada en grafos cuyo vector de características está formado por la métrica de calidad obtenida anteriormente, indicando con un valor entre 0 o 1 la relevancia del artículo. Al emplear un modelo basado en GNN los nodos podrán agregar la información de los nodos vecinos, identificando la importancia de los nodos que son citados por él, y siendo esta información muy útil para realizar la fase de agregación en la cual interviene la métrica asociada a las publicaciones citadas y al propio vector de características del nodo, es decir, a su métrica de calidad.

5.5.1 Modelo basado GCN

El primer modelo que se explorará será entrenado empleando únicamente una capa de convolución, lo que significará que tan solo la información (valor asociado a la métrica de relevancia) de las publicaciones directamente vinculadas al artículo analizado será tomada en cuenta. A priori, ante una métrica desconocida, podríamos dudar sobre si esta se ha generado en base a los artículos citados en este o en base a otras características. Es por ello por lo que, al igual que se hizo en el problema de categorización, analizaremos los resultados obtenidos al emplear profundidades distintas.

El primer modelo constará de una única capa de convolución gráfica seguida de una capa lineal y la función de activación implementada es la tangente hiperbólica. Tras entrenar el modelo durante 35 épocas con un valor de embedding igual a 500 dado que las clases posibles tan solo son 2, se obtienen los resultados presentados a continuación en la Figura 5.16.

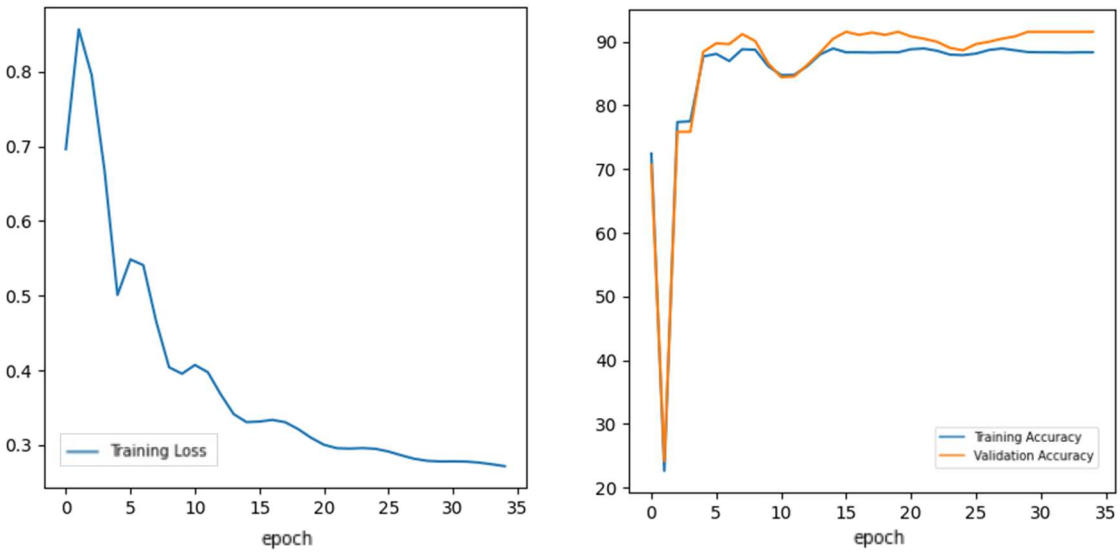


Figura 5.16. entrenamiento del modelo clasificador por relevancia basado en GCN.

Modelo evaluado	Precisión entrenamiento	Precisión evaluación
GCN (1 capa)	**88.28%**	**90.82%**

Desglosando los errores obtenidos, se determina que el 9.18% del set de evaluación no ha sido correctamente clasificado siguiendo la siguiente distribución entre artículos con relevancia académica o sin relevancia académica:

Relación de artículos mal clasificados(GCN)	
Artículos de baja relevancia	25 (32.89%)
Artículos de alta relevancia	51 (67.11%)

Partiendo del hecho de que los artículos de alta relevancia ascienden a un total de 762 en el set completo, conformando un 23% de este. Obtener un error del 67.11% en la clasificación de artículos de alta relevancia podría deberse a la gran cantidad de artículos de baja relevancia que conforman el set con respecto a aquellos de alta relevancia. Es por ello que se observan pequeñas deficiencias al evaluar los nodos que representan publicaciones de elevada calidad.

La representación en el espacio bidimensional se presenta en la siguiente imagen, viendo como claramente la parte inferior agrupa los nodos marcados en gris relacionados con los artículos de elevada calidad e interés.

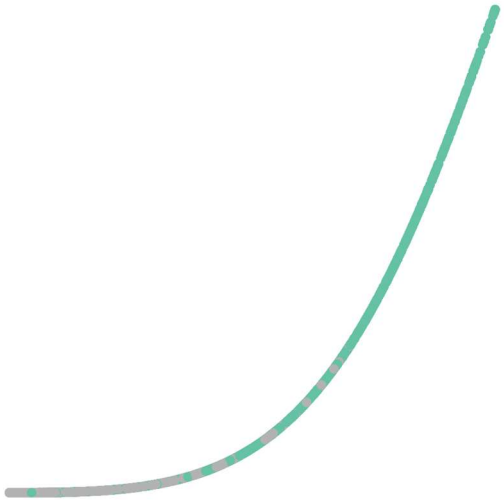


Figura 5.17. agrupación de nodos en el espacio bidimensional con 1 capa tras entrenamiento

Los modelos con múltiples capas inferirán información creando vectores de propiedades que no representan fielmente a la métrica aplicada.

Modelo evaluado	Precisión evaluación	Error relativo precisión
GCN (1 capas)	**90.82%**	**0%**
GCN (2 capas)	86.95%	4.26%
GCN (4 capas)	84.42%	7.04%
GCN (6 capas)	84.17%	7.32%

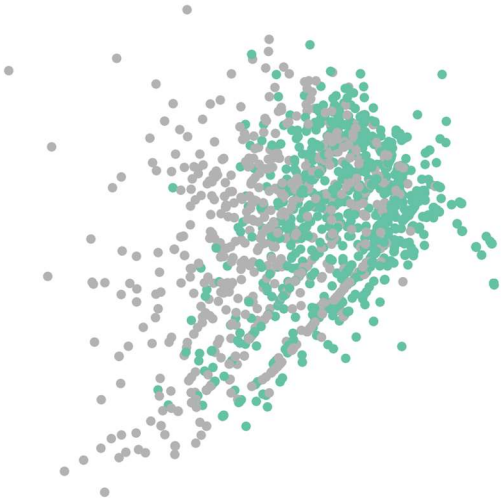


Figura 5.18. agrupación de nodos en el espacio bidimensional con 6 capas.

6 CONCLUSIÓN Y LÍNEA DE DESARROLLO

La implementación basada en redes neuronales gráficas ha demostrado un gran rendimiento arrojando resultados de precisión altos. La investigación en torno a los modelos de redes neuronales busca afrontar nuevos retos estructurando los datos en redes de grafos, logrando abordar problemas tales como la clasificación de papers según su categoría. La precisión lograda con estos modelos, al igual que ocurre con las redes neuronales clásicas, dependerá en gran medida de los datos empleados pues las relaciones establecidas entre los nodos condicionan el comportamiento de nuestro modelo, encontrando limitaciones en el rendimiento de estos. Una de las limitaciones observadas en el conjunto de datos está relacionada con la falta de referencias existentes entre los artículos, encontrando dificultades para realizar una correcta clasificación por relevancia. Ambos modelos, tanto el enfocado a categorización como a clasificación por importancia permiten añadir nuevos nodos a la red de artículos formada mediante un tratamiento para obtener su vector de propiedades y así poder incluirlos. Las limitaciones encontradas con el set de datos motivan la búsqueda de una red de artículos masiva, en la cual se reflejen el número de citaciones real basadas en un mayor número de artículos de cada campo. El entrenamiento de las redes neuronales gráficas con modelos sencillos cuya dimensión de embedding no es excesivamente alta pueden tardar tan solo unos minutos, permitiendo abordar redes de dimensiones mucho mayores de forma muy eficiente, pudiendo analizar las predicciones tomadas mediante las técnicas de explicabilidad, las cuales además de aportar información sobre las características del vector de propiedades permite visualizar la influencia de cada arista.

Las líneas de trabajo futuras que permiten ser abordadas a partir de los modelos básicos de Redes Neuronales Gráficas son, entre otras, la predicción de aristas, pues permite determinar en base a las palabras clave del diccionario si se debe referenciar un determinado artículo del grafo. Cambiando el enfoque seguido hasta el momento, se presenta otra aplicación en la cual las GNN permiten analizar pequeños grafos al completo que, en este caso, podrían ser utilizados para analizar el conjunto de publicaciones de un mismo autor e identificar sus respectivas líneas de trabajo en base a pequeños subgrafos de sus papers.

ANEXO A
CÓDIGO DESARROLLADO CLASIFICADOR

Código A.1 Descarga del set de datos.

```
from torch_geometric.datasets import AttributedGraphDataset
dataset_Polblogs = AttributedGraphDataset(root='C:/Users/PABLO/Desktop/TFG_Dat',name = 'CiteSeer')
data = dataset_Polblogs[0]
```

Código A.2 Definición de máscaras de entrenamiento.

```
train_dat = torch.ones(data.y.shape[0])
test_dat = torch.ones(data.y.shape[0])
for i in range(0,data.y.shape[0]):
    if i%4 == 0:
        train_dat[i] = 0
    else:
        test_dat[i] = 0
train_dat_Bool = train_dat.bool()
test_dat_Bool =  test_dat.bool()
True_index_train =[data.y[i] for i, x in enumerate(train_dat) if x]
True_index_test = [data.y[i] for i, x in enumerate(test_dat) if x]
```

Código A.3 Definición del modelo de GNN unicapa basado en GCN y GAT.

```
emb1 = 500
emb2 = 500
emb3 = 500


class GNN_GCN(torch.nn.Module):
    def __init__(self):
        super().__init__()
        torch.manual_seed(1)
        self.lay1 = GCNConv(dataset_Polblogs.num_features, emb1)
        #self.lay1 = GATConv(dataset_Polblogs.num_features, emb1)
        #self.lay2 = GCNConv(emb1, emb2)
```

```
        #self.lay2 = GATConv(emb1, emb2)
        #self.lay3 = GCNConv(emb2, emb3)
        #self.lay3 = GATConv(emb2, emb3)
        self.out = Linear(emb3, dataset_Polblogs.num_classes)   #dataset_Polblogs.num_classes

    def forward(self, x, edge_index):
        h = self.lay1(x, edge_index)
        h = h.tanh()
        out = self.out(h)
        return out, h

Modelo_GNN = GNN_GCN()
print(Modelo_GNN)
```

Código A.4 Entrenamiento del modelo.

```
Modelo_GNN = GNN_GCN()
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(Modelo_GNN.parameters(), lr=0.0005)

def train(data):
    optimizer.zero_grad()
    out, h = Modelo_GNN(data.x, data.edge_index)
    loss = criterion(out[train_dat_Bool], data.y[train_dat_Bool])
    loss.backward()
    optimizer.step()
    return loss, h ,out



total_loss = []
steps = []
trn_a = []
tst_a = []
for step in range(60):
    loss,value1,out_sal = train(data)
    actr,actes = acc(out_sal)
    trn_a.append(float(actr))
```

```
    tst_a.append(float(actes))
    total_loss.append(float(loss))
    steps.append(float(step))
    if step % 5 == 0:
      print(f'Epoch:' + str(step) +', Loss: '+str(float(loss))+', acct: '+str(float(actr))+', act: '+str(float(actes)))


import matplotlib.pyplot as plt
plt.figure(figsize=(10,15))
fig = plt.figure()
ax0 = fig.add_subplot( title="loss train")
ax0.plot(steps, total_loss, '-', label='train')
```

Código A.5 Gráficas de precisión.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,15))
fig = plt.figure()
ax1 = fig.add_subplot( title="acc")
ax1.plot(steps, trn_a,steps,tst_a, '-', label='train')
```

Código A.6 Visualización en espacio bidimensional.

```
def visualizacion(h, color):
    z = TSNE(n_components=2).fit_transform(h.detach().cpu().numpy())


    plt.figure(figsize=(10,10))
    plt.xticks([])
    plt.yticks([])


    plt.scatter(z[:, 0], z[:, 1], s=70, c=color, cmap="Set2")
    plt.show()


visualizacion(value1, color=data.y)
```

ANEXO B
CÓDIGO DESARROLLADO CALIFICADOR

Código B.1 Obtención del número de citaciones en el set.

```
g = torch_geometric.utils.to_networkx(data,to_undirected=False)
Adj = nx.adjacency_matrix(g)
ind = []
uniones1 = []
uniones = []
peso = []
s = Adj.todense()
for publ in range(0,data.num_nodes):
 con = 0
 for g in range(0,len(s[publ])):
  if s[g][publ]==1:
   con = con+1
   ind.append(g)
  if g ==len(s[publ])-1:
   uniones1.append(con)
for publ in range(0,data.num_nodes):
 con = 0
 for g in range(0,len(s[publ])):
  if s[publ][g]==1:
   con = con+1
   ind.append(g)
  if g ==len(s[publ])-1:
   uniones.append(con)
for publ in range(0,data.num_nodes):
 con = 0
 for g in range(0,len(s[publ])):
  if s[g][publ]==1:
   con = con + uniones[g]
   ind.append(g)
  if g ==len(s[publ])-1:
   peso.append(con)
```

Código B.2 Generador de métrica de calidad.

```
ponder = []
for publ in range(0,data.num_nodes):
 val = uniones[publ]*0.7+peso[publ]*0.3
 ponder.append(val)
for i in range(0,len(ponder)):
 ponder[i] = (ponder[i]-min(ponder))/(np.max(ponder)-np.min(ponder))
```

Código B.3 Etiquetado de artículos por calidad.

```
for publ in range(0,data.num_nodes):
 data.x[publ] = torch.tensor([ponder[publ]])
 if ponder[publ] >= 0.15:
  data.y[publ] = 1
 if ponder[publ] < 0.15:
  data.y[publ] = 0
```

Código B.4 Modelo y entrenamiento para clasificación.

```
emb1 = 500
emb2 = 500
emb3 = 500
class GNN_GCN(torch.nn.Module):
  def __init__(self):
    super().__init__()
    torch.manual_seed(21)
    self.lay1 = GCNConv(dataset_Polblogs.num_features, emb1)
    self.lay2 = GCNConv(emb1, emb2)
    self.lay3 = GCNConv(emb2, emb3)
    self.out = Linear(emb3, 2)
  def forward(self, x, edge_index):
    h = self.lay1(x, edge_index)
    h = h.tanh()
    out = self.out(h)
    return out, h

Modelo_GNN = GNN_GCN()
```

```
Modelo_GNN = GNN_GCN()
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(Modelo_GNN.parameters(), lr=0.0005)


def train(data):
   optimizer.zero_grad()
   out, h = Modelo_GNN(data.x, data.edge_index)
   loss = criterion(out[train_dat_Bool], data.y[train_dat_Bool])
   loss.backward()
   optimizer.step()
   return loss, h ,out
```

REFERENCIAS

[1] G. Cáceres Castellanos, «La importancia de publicar los resultados de Investigación,» *Revista Facultad de Ingeniería,* vol. 23, nº 37, pp. 7-8, 2014.

[2] C. Yáñez Márquez, L. O. López Leyva y M. Aldape Pérez, «Warren Sturgis McCulloch y el joven matemático Walter Pitts,» Instituto Politécnico Nacional. Centro de Investigación en Computación, México, 2007.

[3] M. Ramis, «IDIS,» [En línea]. Available: https://proyectoidis.org/neurona-artificial/. [Último acceso: 23 4 2023].

[4] G. A. Demetriou, « Mobile robotics in education and research.,» *Mobile robots-current trends,* vol. 27, p. 48, 2011.

[5] A. Abeliuk y C. Gutiérrez, «Historia y evolución de la inteligencia artificial,» *Revista Bits de Ciencia,* nº 21, pp. 14-21, 2021.

[6] A. Vaswani y e. al, «Attention is all you need,» *Advances in neural information processing systems,* vol. 30, 2017.

[7] D. H. Rouvray, «The pioneering contributions of Cayley and Sylvester to the mathematical description of chemical structure,» *Journal of Molecular Structure: THEOCHEM,* vol. 185, pp. 1-14, 1989.

[8] J. P. Cequeda Olago, «Protocolos de enrutamiento de estado de enlace,» Cisco Public, 2007.

[9] F. Harary y R. Z. Norman, «Graph Theory as a Mathematical Model in Social Science,» University of Michigan, Institute for Social Research, Ann Arbor, 1953.

[10] J. A. Hartigan y M. A. Wong, «A k-means clustering algorithm,» *Applied statistics,* vol. 28, nº 1, pp. 100-108, 1979.

[11] D. Garavito, «Agrupamiento (clustering)».

[12] S. Deoras, «Why Graph Neural Networks Are Gaining Popularity In 2021,» *Endless Origins,* 2021.

[13] «devAcademy,» Computer World España, [En línea]. Available: https://www.devacademy.es/machine-learning-deep-learning-e-inteligencia-artificial-mundo-habla-ello.

[14] X. Basogain Olabe, «Redes neuronales artificiales y sus aplicaciones,» UPV-EHU.

[15] Y. Orlando Lao-León, «Procedimiento para el pronóstico de la demanda mediante redesneuronales artificiales,» *Ciencias Holguín,* vol. 23, nº 1, pp. 43-59, 2017.

[16] A. Connolly, «Astro ML Interactive Book,» University of Washington, 2020.

[17] «AI Wiki,» 2020. [En línea]. Available: https://machine-learning.paperspace.com/wiki/activation-function. [Último acceso: 25 4 2023].

[18] «Convolutional Neural Networks Explained,» Project Pro, 24 4 2023. [En línea]. Available: https://www.projectpro.io/article/introduction-to-convolutional-neural-networks-algorithm-architecture/560.

[19] H. J. Kelley, «Gradient theory of optimal flight paths,» *ARS Journal,* vol. 10, nº 30, p. 947–954, 1960.

[20] F. Rosenblatt, «Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms Cornell Aeronautical Laboratory,» *Spartan,* pp. 292-301, 1962.

[21] F. Sancho Caparrini, «Entrenamiento de Redes Neuronales: mejorando el Gradiente Descendente,» US, Sevilla, 2020.

[22] S. Ruder, «An overview of gradient descent optimization algorithms,» *arXiv,* 2016.

[23] J. Lorraine, «A Brief Look at Optimization,» 2015.

[24] B. S., «Neural networks and the bias/variance dilemma,» *Neural computation,* vol. 4, nº 1, pp. 1-58, 1992.

[25] B.-V. Tradeoff, «Cowan,Don,» California, 2020.

[26] L. Prechelt, «Early stopping-but when?,» *Neural Networks: Tricks of the trade.,* pp. 55-69, 2002.

[27] R. Jain, «Why "early-stopping" works as Regularization?,» Medium, 2020.

[28] Y. Bai, Y. Wang, Y. Tong, Y. Yang, Q. Liu y J. & Liu, «Boundary content graph neural network for temporal action proposal generation,» *Computer Vision–ECCV 2020,* vol. 16, pp. 121-137, 2020.

[29] P.-F. Kuo, «Airline transportation and arrival time of international disease spread: A case study of Covid-19,» *PLoS ONE,* vol. 16, nº 8, 2021.

[30] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam y P. Vandergheynst, «Geometric deep learning:going beyond Euclidean data,» *IEEE SIG PROC MAG,* 2017.

[31] N. Keriven y G. Peyré, «Universal Invariant and Equivariant Graph Neural Networks,» 2019.

[32] A. V. Nowak y B. S., «A note on learning algorithms for quadratic assignment with graph neural networks,» vol. 1050, p. 22, 2017.

[33] F. Scarselli, «The graph neural network model,» *transactions on neural networks,* vol. 20, nº 1, pp. 61-80, 2008.

[34] T. N. Kipf y M. Welling, «SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS,» 2017.

[35] H. Pie, B. Wei, B. Yang, Y. Lei y K. Chen-Chuan Chang, «GEOM-GCN: GEOMETRIC GRAPH CONVOLUTIONAL NETWORKS,» *ICLR,* p. 8, 2020.

[36] J. Zhang, J. Ding, S. Liu y H. Wu, «Meta-Path-Free Representation Learning on Heterogeneous Networks,» 2019.

[37] P. Veličković, «Graph attention networks,» *arXiv preprint,* 2017.

[38] A. Rossi, M. Tiezzi, G. M. Dimitri, M. Bianchini y M. Maggini, «Inductive–transductive learning with graph neural networks,» *Artificial Neural Networks in Pattern Recognition,* vol. 8, pp. 19-21, 2018.

[39] C. L. Giles, «paperswithcode,» [En línea]. Available: https://paperswithcode.com/dataset/citeseer.

[40] R. A y K. Ahmed, «The Network Data Repository with Interactive Graph Analytics and Visualization,» 2015. [En línea]. Available: https://networkrepository.com/citeseer.php.

[41] G. J. da S. Duarte, T. Arruda Pereira y E. Jhones F. Nascimento, «How do loss functions impact the performance of graph neural networks?,» pp. 1-7, 2021.

[42] A. Kiss y V. L. Ahn, «Efficient Processing SAPE Queries Using the Dynamic Labelling Structural Indexes,» de *Advances in Databases and Information Systems*, Berlin , Springer , 2006, pp. 232-247.

ÍNDICE DE FIGURAS