Universidad de La Laguna

Máster Universitario en Ciberseguridad e Inteligencia de Datos

Exploración y Evaluación de Técnicas de *Deep Learning* con Grafos

"Exploration and Evaluation of Deep Learning Techniques with Graphs"

Sergio Pitti de Armas

La Laguna, 7 de julio de 2023

D. **Marcos Colebrook Santamaría**, con N.I.F. 43.787.808-V, profesor titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Exploración y Evaluación de Técnicas de *Deep Learning* con Grafos"

ha sido realizada bajo su dirección por D. **Sergio Pitti de Armas**, con N.I.F. 51.149.697-C.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a Fecha 7 de julio de 2023

Agradecimientos

Quiero agradecer a todas las personas que me han ayudado y apoyado en esta última etapa académica. En primer lugar, a mi tutor, Marcos Colebrook. También a mi familia y amigos sin los cuales no habría llegado tan lejos. Por apoyarme y entenderme todos estos años. A mi madre, por apoyarme incondicionalmente día a día. Y en especial a mi padre, por siempre haberme apoyado y creído en mí. Siempre serás un ejemplo a seguir. Gracias.

Licencia

Resumen

Los grafos son una herramienta esencial para representar, explorar, predecir y explicar fenómenos del mundo real y digital. Con el creciente volumen y complejidad de los datos, cada vez es más importante tener la capacidad de manejar grandes grafos (Big Graphs) de manera efectiva. En este proyecto, se abordará la aplicación de los grafos en el campo de la Inteligencia Artificial (IA), específicamente en áreas como Data Science (DS), Machine Learning (ML) y Deep Learning (DL). El objetivo principal de este proyecto es llevar a cabo pruebas de concepto (Proof of Concept, PoC) para explorar el potencial de los grafos en la IA y proporcionar una base teórica sólida para futuras investigaciones en el campo.

Los grafos tienen una larga historia en la teoría de grafos y matemáticas, y en los últimos años han encontrado una amplia aplicación en la ciencia de datos y la inteligencia artificial. En particular, los grafos son utilizados en el análisis de redes sociales, la recomendación de productos y servicios, la optimización de rutas y la detección de anomalías.

En el campo del Machine Learning, los grafos son utilizados para la representación y modelado de datos estructurados, como los datos de redes sociales, los datos biológicos y los datos de sistemas de recomendación. Además, los algoritmos de grafos se utilizan en la detección de comunidades, la clasificación y la predicción de enlaces.

En el campo del Deep Learning, los grafos son utilizados en redes neuronales que pueden aprender a representar y modelar datos de grafo, como los datos de redes sociales y los datos biológicos. Estos modelos de grafos han demostrado ser efectivos en la predicción de enlaces, la clasificación de nodos y la detección de anomalías.

Es por todo ello que los grafos son una herramienta valiosa en la inteligencia artificial y la ciencia de datos, y su aplicación continúa evolucionando y expandiéndose en muchos campos. Este proyecto se centrará en explorar la aplicación de los grafos en el Data Science, Machine Learning y Deep Learning a través de un PoC.

Palabras clave: Machine Learning, Inteligencia Artificial, Redes Neuronales basadas en Grafos, GNN, Análisis de Datos

Abstract

Graphs are an essential tool for representing, exploring, predicting and explaining phenomena in the real and digital world. With the increasing volume and complexity of data, it is becoming very important to have the ability to handle Big Graphs effectively. In this project, we will address the application of graphs in the field of Artificial Intelligence (AI), specifically in areas such as Data Science (DS), Machine Learning (ML) and Deep Learning (DL). The main objective of this project is to carry out a Proof of Concept (PoC) to explore the potential of graphs in AI and to provide a solid theoretical basis for future research in the field.

Graphs have a long history in graph theory and mathematics, and in recent years have found wide applications in data science and artificial intelligence. Graphs are used in social network analysis, product and service recommendation, route optimization and anomaly detection.

In the field of Machine Learning, graphs are used for the representation and modelling of structured data, such as social network data, biological data and recommender system data. In addition, graph algorithms are used in community detection, classification and link prediction.

In the field of Deep Learning, graphs are used in neural networks that can learn to represent and model graph data, such as social network data and biological data. These graph models have proven to be effective in link prediction, node classification and anomaly detection.

This is why graphs are a valuable tool in artificial intelligence and data science, and their application continues to evolve and expand in many fields. This project will focus on exploring the application of graphs in Data Science, Machine Learning and Deep Learning through a PoC.

Keywords: Machine Learning, Artificial Intelligence, Graph Neural Networks, GNN, Data Analysis.

Índice general

Capítulo 1 Introducción 1

Capítulo 2 Antecedentes 4

2.1 Inteligencia Artificial 4

2.2 Machine Learning 5

Capítulo 3 Estado del Arte 8

3.1 Redes Neuronales y Deep Learning 8

3.2 Grafos en Inteligencia Artificial 11

3.3 Redes Neuronales Basadas en Grafos 13

Capítulo 4 Diseño e Implementación 23

4.1 Conceptos Tecnológicos, CPU vs GPU 23

4.2 Librerías de Aprendizaje Profundo 26

Capítulo 5 PoC y Resultados 31

5.1 Proof of Concept 33

Capítulo 6 Conclusiones y Líneas Futuras 38

Capítulo 7 Summary and Conclusions 41

Capítulo 8 Presupuesto 43

8.1 Presupuesto 43

Apéndice: Código Destacable 44

A.1 Definición de la clase GCN 44

Bibliografía 45

Índice de figuras

Figura 1.1 - Agregación de Nodos en una GNN…… 2

Figura 3.1 - Comparativa entre neuronas reales y artificiales…….. 8

Figura 3.2 - Representación de las capas de una GNN…… 9

Figura 3.3 - Ejemplo de Convolución …….. 16

Figura 5.1 - Ejemplo de propagación hacia delante y hacia atrás…….. 35

Figura 5.2 - Extracto de epochs durante el entrenamiento …… 36

Figura 5.3 - Resultados de la evaluación del modelo…….. 36

Capítulo 1
Introducción

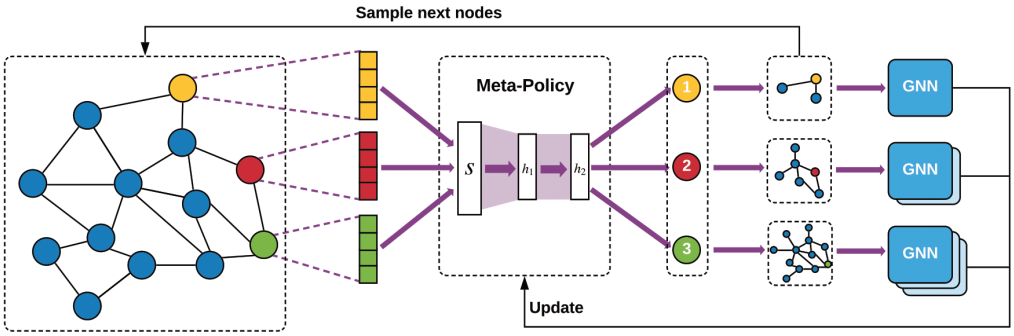
Los métodos de *Deep Learning* [1] desempeñan un papel cada vez más importante en las aplicaciones de los sistemas de recomendación. Estos se utilizan para aprender patrones o relaciones a bajo nivel de imágenes, texto e incluso redes de usuarios. Las representaciones aprendidas mediante modelos pueden utilizarse para complementar, o incluso sustituir, a los algoritmos de recomendación tradicionales. Estas representaciones aprendidas tienen una gran utilidad ya que es posible su reutilización en diversas tareas de recomendación. Por ejemplo, las incrustaciones de ítems [2] aprendidas mediante un modelo profundo pueden utilizarse para la recomendación ítem-ítem o para recomendar colecciones temáticas, como listas de reproducción o contenido tipo "*feed*".

En los últimos años, se han producido avances significativos en este ámbito, especialmente el desarrollo de nuevos métodos de *Deep Learning* capaces de aprender sobre datos estructurados en grafos, lo que resulta fundamental para las aplicaciones de recomendación. Los grafos aparecen de forma natural en numerosos ámbitos de aplicación, desde el análisis social y la bioinformática hasta la visión por ordenador. La capacidad única de los grafos permite captar las relaciones estructurales entre los datos y, por tanto, obtener más información que analizando los datos de forma aislada.

Sin embargo, a menudo es muy difícil resolver los problemas de aprendizaje en grafos. Esto se debe a que muchos tipos de datos no están estructurados originalmente como grafos, como las imágenes y los datos de texto, y para los datos estructurados en grafos, los patrones de conectividad subyacentes son a menudo complejos y diversos. Aunque se han realizado enormes esfuerzos para abordar el problema del aprendizaje de la representación de grafos, muchos de ellos siguen adoleciendo de mecanismos de aprendizaje poco profundos. Los modelos de aprendizaje profundo sobre grafos, como las redes neuronales de grafos, han surgido recientemente en el aprendizaje automático y otras áreas relacionadas, y han demostrado un rendimiento superior en diversos problemas.
Entre estos avances recientes destaca el éxito de las arquitecturas de aprendizaje profundo conocidas como redes neuronales de grafos (*Graph Neural Networks*, GNN) [3]. La idea central de las GNN es aprender a agregar iterativamente

información de características a partir de vecindarios de grafos locales utilizando redes neuronales (ver figura 1.1). En este caso, una única operación de "convolución" transforma y agrega información de características de la vecindad de un nodo del grafo. Apilando múltiples convoluciones de este tipo se puede propagar la información a través de los extremos de dicho grafo. A diferencia de los modelos profundos basados exclusivamente en el contenido, las GCN (*Graph Convolutional Networks*, redes convolucionales en grafos) [4] aprovechan tanto la información del contenido como la estructura del grafo.

Figura 1.1 - Agregación de Nodos en una GNN



NOTA: Policy-GNN: Aggregation Optimization for Graph Neural Networks | Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. (s. f.). Recuperado 6 de julio de 2023, de https://dl.acm.org/doi/10.1145/3394486.3403088

Los métodos basados en GNN han establecido un nuevo estándar en innumerables puntos de referencia de sistemas de recomendación. Sin embargo, estas mejoras en los modelos aún no se han traducido en mejoras en los entornos de producción del mundo real. El principal reto de las redes neuronales basadas en grafos es la escalabilidad, tanto en el entrenamiento como en la inferencia de las incrustaciones de los nodos. La ampliación de estas redes es muy costosa ya que muchos de los supuestos básicos en los que se basa su diseño se incumplen cuando se trabaja en un entorno de *Big Data*. Por ejemplo, todos los sistemas de recomendación basados en GCN requieren operar con el conjunto completo del grafo durante el entrenamiento, un supuesto que resulta inviable cuando el grafo subyacente tiene miles de millones de nodos y su estructura evoluciona constantemente.
En este proyecto, se realizará una revisión exhaustiva del campo emergente de las redes convolucionales de grafos. En primer lugar, se realizará un recorrido por los antecedentes de las GNN en los que se abordarán campos como el *Machine Learning*, *Deep Learning* y los principales avances relacionados con el estudio de redes neuronales.

A continuación, se realizará una investigación del estado del arte de las distintas áreas de conocimiento relacionado con los grafos y las redes neuronales basadas en grafos y se finalizará con una prueba de concepto (PoC, *Proof of Concept*) en las que se apliquen estas técnicas de *Data Science*, *Machine Learning* y *Deep Learning*.

Capítulo 2
Antecedentes

2.1 Inteligencia Artificial

El concepto de inteligencia artificial (IA) [5] tiene profundas raíces en la historia de la humanidad, con mitologías antiguas y elucubraciones filosóficas que a menudo contemplan la idea de crear máquinas inteligentes. Sin embargo, el desarrollo formal de la IA como disciplina científica empezó a tomar forma a mediados del siglo XX.

El nacimiento de la IA como campo se atribuye a menudo a la Conferencia de Dartmouth celebrada en 1956 [6]. En esta conferencia se reunieron investigadores de diversas disciplinas motivados por la idea de crear máquinas que pudieran manifestar inteligencia. Exploraron la posibilidad de desarrollar algoritmos y modelos que pudieran simular la inteligencia humana y resolver problemas complejos.

En los años siguientes, los investigadores se adentraron en el desarrollo de sistemas de IA basados en la manipulación simbólica y el razonamiento lógico. Esta, conocida como IA simbólica, supuso la representación del conocimiento mediante símbolos y la creación de sistemas expertos. Estos sistemas expertos utilizaban el razonamiento basado en reglas para abordar tareas específicas. Algunos ejemplos son MYCIN [7], un sistema de diagnóstico médico, y DENDRAL [8], un sistema de análisis químico.

En las décadas de 1980 y 1990, la investigación en IA cambió de orientación con el auge del aprendizaje automático. Los investigadores empezaron a explorar algoritmos que pudieran aprender automáticamente de los datos y mejorar su rendimiento sin programación explícita. Este periodo fue testigo de la creación de las redes neuronales, el aprendizaje de árboles de decisión y la aparición de la teoría del aprendizaje estadístico.

Sin embargo, a finales de los 80 y los 90 estuvo marcado un periodo conocido como el "invierno de la IA" [9]. Las expectativas exageradas y las promesas incumplidas

de la tecnología de IA provocaron un descenso de la financiación y el interés. El progreso en la investigación de la IA se estancó y el campo experimentó un parón.

En la década de 1990 y principios de 2000, un renovado interés por las aplicaciones prácticas y los nuevos paradigmas de aprendizaje insuflaron nueva vida a la IA. El aprendizaje por refuerzo ganó importancia como subcampo del aprendizaje automático, que consistía en entrenar a agentes para que tomaran decisiones secuenciales en entornos dinámicos. Empezaron a surgir aplicaciones prácticas de la IA, como el filtrado de correo electrónico, los sistemas de visión por ordenador y las tecnologías de reconocimiento de voz.

La década de 2010 fue testigo de un resurgimiento de la IA, impulsada por la disponibilidad de grandes cantidades de datos y los avances en potencia informática. El aprendizaje profundo (*Deep Learning*), un subconjunto del aprendizaje automático, logró avances significativos utilizando redes neuronales artificiales con múltiples capas. Los modelos de aprendizaje profundo lograron resultados notables en el reconocimiento de imágenes y del habla, el procesamiento del lenguaje natural y otros ámbitos.

Hoy en día, la IA sigue avanzando rápidamente. El aprendizaje por refuerzo se está aplicando a la robótica y a los agentes de juego, ampliando los límites de lo que las máquinas pueden lograr. Modelos generativos como las redes generativas adversariales (*Generative Adversarial Networks*, GAN) [10] y los transformadores [11] han revolucionado la generación de imágenes y textos.

La historia de la Inteligencia Artificial está marcada por un ciclo continuo de innovación, retrocesos y resurgimientos. Es un campo dinámico y en constante evolución que sigue dando forma a nuestra comprensión de la inteligencia y las capacidades de las máquinas. Cada año que pasa, nuevos algoritmos, técnicas y consideraciones éticas configuran el futuro de la inteligencia artificial.

2.2 Machine Learning

El aprendizaje automático, o *Machine Learning* (ML) [12], ha surgido como un campo revolucionario en el ámbito de la inteligencia artificial, transformando la forma en que procesamos e interpretamos los datos. Se trata de un subcampo que se centra en el desarrollo de algoritmos y modelos que permiten a los ordenadores aprender y hacer predicciones o tomar decisiones. Gracias a su capacidad para

analizar grandes cantidades de información y extraer información valiosa, el aprendizaje automático ha encontrado aplicaciones en diversos ámbitos, desde la sanidad y las finanzas hasta el transporte y el entretenimiento.

En esencia, el aprendizaje automático se basa en el concepto de entrenamiento de modelos a partir de datos etiquetados. Inicialmente, se crea un modelo y se le proporciona un conjunto de ejemplos, donde cada ejemplo consiste en datos de entrada y un valor de salida u objetivo correspondiente. A continuación, el modelo aprende patrones y relaciones dentro de los datos, ajustando sus parámetros internos para minimizar la diferencia entre sus resultados previstos y los resultados reales. Este proceso, conocido como aprendizaje supervisado, permite al modelo generalizar y hacer predicciones precisas sobre datos no vistos.

Algunos de los algoritmos más utilizados en el aprendizaje supervisado son los basados en redes neuronales. Estos se inspiran en la estructura y funcionalidad del cerebro humano para construir modelos complejos. Las redes neuronales están formadas por capas interconectadas de neuronas artificiales, cada una de las cuales realiza un cálculo simple y pasa los resultados a la capa siguiente. A través de múltiples capas, la red puede aprender representaciones complejas de los datos de entrada. El aprendizaje profundo, un subconjunto del aprendizaje automático, se centra en redes neuronales con muchas capas, conocidas como redes neuronales profundas, y ha logrado un éxito notable en áreas como el reconocimiento de imágenes y el procesamiento del lenguaje natural. En los próximos capítulos se abordará este campo, sus avances, así como sus distintas aplicaciones.

Otra categoría importante del ML es el aprendizaje no supervisado, en el que el modelo aprende de datos no etiquetados sin orientación explícita. En lugar de predecir resultados específicos, los algoritmos de aprendizaje no supervisado tratan de descubrir patrones o estructuras subyacentes en los datos. Los algoritmos de agrupación, por ejemplo, agrupan puntos de datos similares basándose en sus similitudes o distancias inherentes. Las técnicas de reducción de la dimensionalidad, por su parte, pretenden reducir la complejidad de los datos captando sus características esenciales.

Por último, el aprendizaje por refuerzo se centra en el entrenamiento de agentes para que interactúen con un entorno y aprendan comportamientos óptimos mediante ensayo y error. El agente recibe retroalimentación en forma de recompensas o penalizaciones en función de sus acciones, lo que le permite

mejorar iterativamente su proceso de toma de decisiones. El aprendizaje por refuerzo ha demostrado un gran potencial en ámbitos como la robótica, los juegos y los sistemas autónomos.

Sin embargo, el aprendizaje automático no está exento de dificultades. La calidad y el sesgo de los datos, la interpretabilidad, las consideraciones éticas y la privacidad son algunos de los problemas a los que se enfrentan investigadores y profesionales. Garantizar la imparcialidad y la responsabilidad en los sistemas de aprendizaje automático es crucial para mitigar los sesgos y evitar resultados discriminatorios.

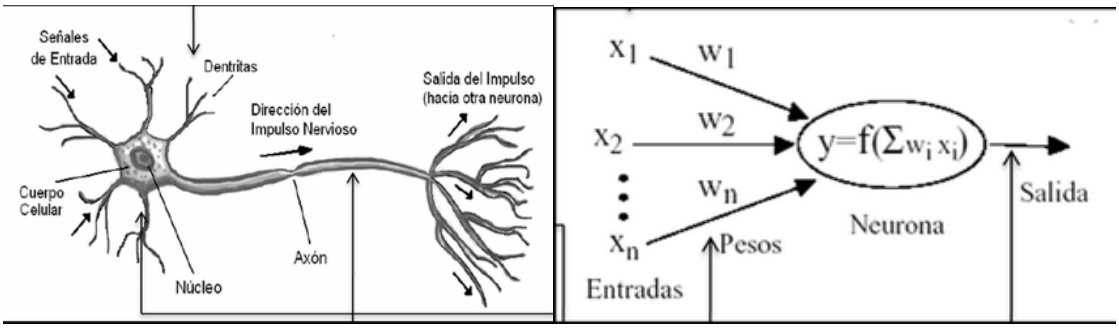
En conclusión, la inteligencia artificial y el aprendizaje automático han dado grandes pasos hacia una nueva era de procesamiento de datos y toma de decisiones. Su capacidad para aprender de grandes cantidades de datos, identificar patrones y hacer predicciones precisas ha revolucionado numerosos sectores y abierto interesantes posibilidades de innovación como el aprendizaje profundo y las redes neuronales.

Capítulo 3
Estado del Arte

3.1 Redes Neuronales y Deep Learning

Las redes neuronales (neural networks, NN) se han convertido en uno de los avances más revolucionarios en el campo del aprendizaje automático. Estos modelos computacionales, inspirados en el intrincado funcionamiento de los cerebros biológicos, han demostrado una capacidad excepcional para captar y comprender patrones y relaciones complejas en los datos (Ver figura 3.1). Al simular la interconexión de las neuronas, las redes neuronales han demostrado ser muy eficaces en diversos ámbitos, desde el reconocimiento de imágenes y del habla hasta el procesamiento del lenguaje natural e incluso los juegos.

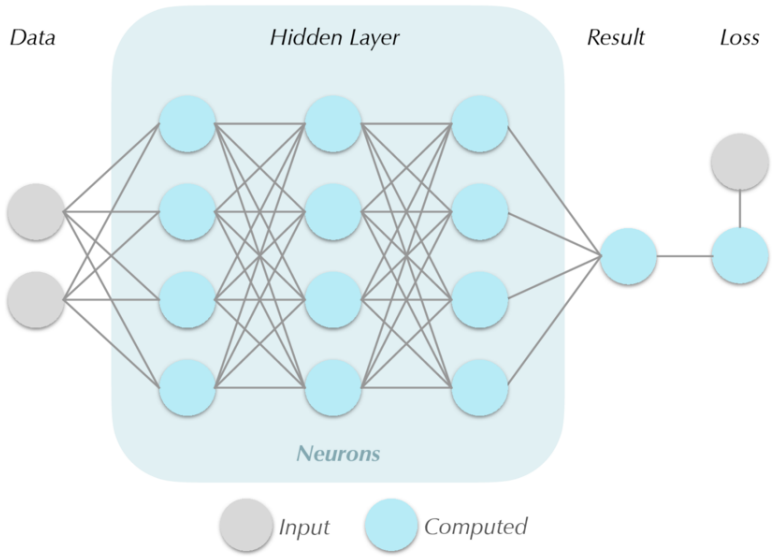
Figura 3.1 - Comparativa entre neuronas reales y artificiales



NOTA: Lao León, Y., Rivas Méndez, A., Pérez Pravia, M., & Marrero Delgado, F. (2017). Procedimiento para el pronóstico de la demanda mediante redes neuronales artificiales / Procedure for forecasting demand by using artificial neural networks. Ciencias Holguin, 23, 43-59.

Las redes neuronales profundas, también denominadas modelos de aprendizaje profundo (*Deep Learning*), llevan el concepto de redes neuronales al siguiente nivel. Estos modelos constan de múltiples capas, lo que les permite aprender representaciones jerárquicas de los datos (Ver figura 3.2). Al extraer progresivamente características de nivel superior a partir de datos de entrada sin procesar, los modelos de aprendizaje profundo han logrado notables mejoras de rendimiento en una amplia gama de aplicaciones. Esta capacidad de aprender automáticamente representaciones jerárquicas ha posicionado al *Deep Learning* como una potente herramienta para manejar conjuntos de datos grandes y desestructurados.

Figura 3.2 - Representación de las capas de una GNN



NOTA: Wang, Y., Moradi, R., Haghighi, M., & Rastegarnia, F. (2023). *Introduction of Machine Learning for Astronomy (Hands-on Workshop)*. https://doi.org/10.48550/arXiv.2302.06475

Una de las principales ventajas del aprendizaje profundo es su capacidad para aprender características directamente de los datos, eliminando la necesidad de ingeniería manual de características. Los algoritmos tradicionales de aprendizaje automático suelen basarse en características creadas a mano, lo que puede llevar mucho tiempo y ser propenso a errores humanos. El aprendizaje profundo, en cambio, destaca en el descubrimiento automático de patrones y representaciones intrincadas que podrían haber sido difíciles de definir explícitamente para los humanos. Este enfoque basado en datos ha demostrado ser muy eficaz en tareas como la clasificación de imágenes, la detección de objetos, la síntesis de voz y la traducción automática, entre otras.

El proceso de entrenamiento de las redes neuronales profundas consta de dos pasos principales: la propagación hacia delante y la retropropagación [13]. Durante la propagación hacia delante, los datos de entrada se introducen en la red y los cálculos se realizan capa por capa para producir una salida. Esta salida se compara con la salida deseada y se calcula una señal de error.

La retropropagación, un paso crucial en el aprendizaje profundo, consiste en propagar esta señal de error hacia atrás a través de la red, ajustando los parámetros de la red (pesos y sesgos) para minimizar el error. Este proceso

iterativo de propagación hacia adelante y retropropagación, a menudo apoyado por algoritmos de optimización como el descenso por gradiente, permite a las redes neuronales profundas aprender y mejorar su rendimiento con el tiempo.

La arquitectura de una red neuronal profunda puede variar en función de la tarea a realizar. Las redes neuronales convolucionales (*Convolutional Neural Networks*, CNN) [14] se han hecho muy populares en tareas relacionadas con imágenes y vídeos, ya que aprovechan la estructura espacial de los datos. Las CNN utilizan capas convolucionales para extraer patrones locales y representaciones jerárquicas, lo que las hace especialmente eficaces para tareas como la clasificación de imágenes, la detección de objetos y la segmentación de imágenes.

Las redes neuronales recurrentes (*Recurrent Neural Networks*, RNN) [15], por su parte, están diseñadas para manejar datos secuenciales, lo que las hace adecuadas para tareas como el modelado del lenguaje, el reconocimiento del habla y la traducción automática. Las conexiones recurrentes de las RNN les permiten captar dependencias temporales y procesar secuencias de entrada de longitud variable. Además, existen arquitecturas especializadas como las redes generativas adversariales (*Generative Adversarial Networks*, GAN) para generar nuevas muestras de datos y transformadores para tareas de procesamiento del lenguaje natural, incluida la generación de texto y la traducción automática.

El éxito del *Deep Learning* puede atribuirse a varios factores. En primer lugar, la disponibilidad de conjuntos de datos etiquetados a gran escala ha desempeñado un papel fundamental en el entrenamiento de modelos profundos en cantidades masivas de datos. Estos conjuntos de datos han permitido a los modelos de aprendizaje profundo aprender de una amplia gama de ejemplos, lo que permite generalizar de forma correcta ante datos no vistos con anterioridad.

Por otro lado, los avances en hardware, en particular las unidades de procesamiento gráfico (GPU) y las unidades especializadas de procesamiento tensorial (TPU) [16], han acelerado considerablemente los cálculos necesarios para entrenar redes neuronales profundas. Las capacidades de procesamiento paralelo de las GPU y las TPU han acelerado el proceso de entrenamiento, haciendo que el aprendizaje profundo sea más accesible y factible para grandes conjuntos de datos. Además, el desarrollo de marcos de aprendizaje profundo como TensorFlow y PyTorch ha simplificado la implementación y el despliegue de modelos de aprendizaje profundo, proporcionando una interfaz de alto nivel para construir

arquitecturas complejas y manejar los cálculos subyacentes de manera eficiente.

Sin embargo, a pesar de sus impresionantes logros, el aprendizaje profundo sigue enfrentándose a ciertos retos. El entrenamiento de redes neuronales profundas suele requerir importantes recursos informáticos y puede resultar caro desde el punto de vista computacional. El gran número de parámetros de los modelos profundos requiere una memoria y una capacidad de procesamiento considerables, por lo que es necesario utilizar una infraestructura de hardware de altas prestaciones. Otro problema que hay que resolver es la sobreadaptación, es decir, cuando el modelo funciona bien con los datos de entrenamiento, pero mal con los datos no observados. Continuamente se exploran técnicas como la regularización, el abandono y el aumento de datos para mitigar estos problemas y mejorar las capacidades de generalización de los modelos de aprendizaje profundo.

En conclusión, las redes neuronales y el aprendizaje profundo han revolucionado el campo del aprendizaje automático, desatando una nueva era de posibilidades. Su capacidad para aprender representaciones jerárquicas a partir de datos brutos y extraer patrones complejos ha impulsado avances en numerosos ámbitos. Con la investigación en curso, los avances en tecnología de hardware y el desarrollo de nuevos algoritmos, las aplicaciones potenciales del aprendizaje profundo son amplias y prometedoras.

3.2 Grafos en Inteligencia Artificial

Los grafos han surgido como una herramienta fundamental en el campo del aprendizaje automático y la inteligencia artificial, revolucionando la forma en que representamos y analizamos estructuras de datos complejas. Un grafo es una colección de nodos (también conocidos como vértices) conectados por aristas, donde cada arista representa una relación o conexión entre dos nodos. Esta representación gráfica nos permite capturar y explotar las dependencias e interacciones presentes en los datos del mundo real. Una de las principales ventajas del uso de grafos en el aprendizaje automático es su capacidad para modelizar datos relacionales. En muchos ámbitos, los datos no son entidades aisladas, sino entidades interconectadas a través de diversas relaciones. Al representar estos datos como un grafo, podemos capturar y aprovechar estas relaciones para obtener información valiosa y realizar predicciones.

Los algoritmos de M*achine Learning* basados en grafos ofrecen una amplia gama de aplicaciones. En el análisis de redes sociales, los grafos nos permiten analizar y comprender la estructura y la dinámica de las interacciones y relaciones entre usuarios. Al examinar la topología de los grafos, podemos identificar nodos influyentes, detectar comunidades, medir la centralidad y predecir conexiones ausentes o futuras. Estos conocimientos tienen implicaciones en ámbitos como la publicidad dirigida, los sistemas de recomendación y el estudio de la difusión de la información y la influencia social.

Los sistemas de recomendación se benefician enormemente de los enfoques basados en grafos. Representando a los usuarios y los artículos como nodos y capturando sus interacciones como aristas, es posible construir un grafo de recomendaciones. Aprovechando esta estructura de grafo, los algoritmos de aprendizaje automático pueden hacer recomendaciones personalizadas basadas en similitudes entre usuarios o elementos. Las técnicas de filtrado colaborativo, como los métodos basados en artículos o usuarios, explotan las nociones basadas en grafos para encontrar recomendaciones relevantes recorriendo el grafo y teniendo en cuenta las preferencias de usuarios similares o las similitudes entre artículos.

Por otro lado, el procesamiento del lenguaje natural (*Natural Language Processing*, NLP) [17] también se beneficia de las representaciones en forma de grafo. Los grafos de dependencia, en los que las palabras se representan como nodos y las aristas como relaciones sintácticas, permiten analizar la estructura de las frases, realizar análisis sintácticos y comprender las conexiones semánticas entre palabras. Las técnicas de NLP basadas en grafos ayudan en tareas como la respuesta a preguntas, el análisis de sentimientos y el resumen de textos. Además, los grafos de conocimiento, que codifican las relaciones semánticas entre entidades, facilitan la representación y recuperación de información estructurada y permiten realizar razonamientos e inferencias sofisticados.

Por último, en el campo de la visión por ordenador, los grafos han demostrado su valor en tareas como la segmentación de imágenes y la detección de objetos. Al construir una representación gráfica de una imagen, en la que los nodos corresponden a regiones de la imagen y los bordes codifican relaciones espaciales o contextuales, los algoritmos de aprendizaje automático pueden identificar objetos, segmentar imágenes en regiones significativas y aprovechar la información contextual global para una comprensión y un análisis precisos.

Más allá de los dominios específicos, las técnicas basadas en grafos encuentran aplicaciones en la detección de anomalías, la detección de fraudes y el análisis de redes. Los comportamientos anómalos suelen reflejarse en desviaciones de los patrones esperados dentro de la estructura del grafo. Los algoritmos de aprendizaje automático pueden aprovechar las propiedades de los grafos, como la centralidad de los nodos o la detección de comunidades, para identificar valores atípicos o patrones inusuales, lo que ayuda a detectar fraudes o anomalías en diversos sistemas.

Con todo esto, los grafos se han convertido en una herramienta indispensable para el aprendizaje automático y la inteligencia artificial. Proporcionan un marco versátil para representar y analizar estructuras de datos complejas, permitiendo aprovechar las relaciones y dependencias para obtener información y tomar decisiones informadas. Con aplicaciones que abarcan el análisis de redes sociales, los sistemas de recomendación, la NLP, la visión por ordenador y la detección de anomalías, los enfoques basados en grafos siguen mejorando las capacidades del aprendizaje automático y contribuyendo al desarrollo de sistemas inteligentes en diversos ámbitos.

Otro de los ámbitos donde los grafos se han convertido en un componente de gran relevancia es en el campo del aprendizaje profundo. Las redes neuronales de grafos (*Graph Neural Networks*, GNN) aprovechan la estructura y la conectividad inherentes a los datos de grafos para mejorar las capacidades de aprendizaje y razonamiento. Las GNN permiten a los modelos de aprendizaje profundo propagar información a través de los nodos de los grafos, capturando las dependencias locales y globales y aprendiendo representaciones que incorporan la topología de los grafos. Al fusionar la potencia del aprendizaje profundo con las estructuras de grafos, las GNN han hecho avanzar significativamente tareas como la clasificación de nodos, la predicción de enlaces y la generación de grafos, abriendo nuevas vías para aprovechar datos relacionales complejos y allanando el camino para modelos más eficaces e interpretables en el ámbito de la inteligencia artificial.

3.3 Redes Neuronales Basadas en Grafos

Las redes neuronales basadas en grafos han surgido como potentes herramientas para modelar y analizar datos estructurados complejos. Las GNN son una clase de redes neuronales diseñadas específicamente para operar con datos estructurados en grafos. Estas aprovechan la información estructural inherente a

los grafos para aprender representaciones significativas de nodos y aristas. Al propagar la información entre los nodos conectados, las GNN pueden captar las dependencias locales y globales de un grafo, lo que les confiere una gran capacidad de razonamiento y predicción.

El componente básico de una GNN es la capa convolucional de grafos. Al igual que las capas convolucionales de las redes neuronales convolucionales (*Convolutional Neural Networks*, CNN) tradicionales, las capas convolucionales de grafos aplican una operación localizada a cada nodo agregando información de sus vecinos. Este paso de agregación, a menudo denominado paso de mensajes, permite a la red captar la influencia de los nodos vecinos sobre el nodo objetivo. Al apilar varias capas convolucionales de grafos, las GNN pueden captar patrones y relaciones cada vez más complejos en el grafo.

Las GNN han demostrado un éxito notable en una amplia gama de aplicaciones. En el campo del análisis de redes sociales, pueden predecir las preferencias de los usuarios, identificar comunidades, detectar comportamientos anómalos e incluso ayudar en campañas de marketing viral. En bioinformática, por ejemplo, se han utilizado para predecir estructuras de proteínas, analizar interacciones moleculares y diseñar nuevos fármacos. Además, las GNN se han mostrado prometedoras en sistemas de recomendación, predicción del tráfico, detección del fraude, procesamiento del lenguaje natural y muchos otros ámbitos.

Otra línea de investigación a tener en cuenta es la interpretabilidad de las GNN. A medida que los modelos de aprendizaje profundo se vuelven cada vez más complejos, resulta crucial comprender el razonamiento que subyace a las predicciones. Se están realizando esfuerzos para desarrollar métodos que expliquen el proceso de toma de decisiones de las GNN y proporcionen información sobre las características y relaciones importantes en el grafo. Esta interpretabilidad puede ayudar a generar confianza en las GNN, especialmente en ámbitos como la sanidad y las finanzas, donde la transparencia en la toma de decisiones es crucial.

Además, recientemente las GNN se han ampliado para manejar estructuras gráficas más complejas, como grafos dirigidos, “multigrafos” e “hipergrafos” [18]. Estas extensiones pretenden captar propiedades estructurales adicionales y mejorar la capacidad expresiva de las GNN. Los grafos dirigidos, por ejemplo, permiten modelar relaciones asimétricas, mientras que los “multigrafos” permiten capturar múltiples tipos de interacciones entre entidades. Los “hipergrafos”, por

su lado, ofrecen una representación más flexible al permitir que las aristas conecten varios nodos simultáneamente. Estos avances mejoran la capacidad de las GNN para modelar sistemas del mundo real con patrones de conectividad complejos y diversos.

Cabe mencionar que, se están desarrollando arquitecturas GNN capaces de gestionar grafos dinámicos, en los que la estructura del grafo cambia con el tiempo. Las GNN dinámicas permiten modelar dependencias temporales y relaciones en evolución en sistemas como las redes sociales o las redes de transporte [19].

3.4 Tipos de Redes Neuronales Basadas en Grafos

Existen un gran número de tipos de GNN. No obstante, la mayoría de ellas se pueden agrupar en cuatro grandes conjuntos [20]:

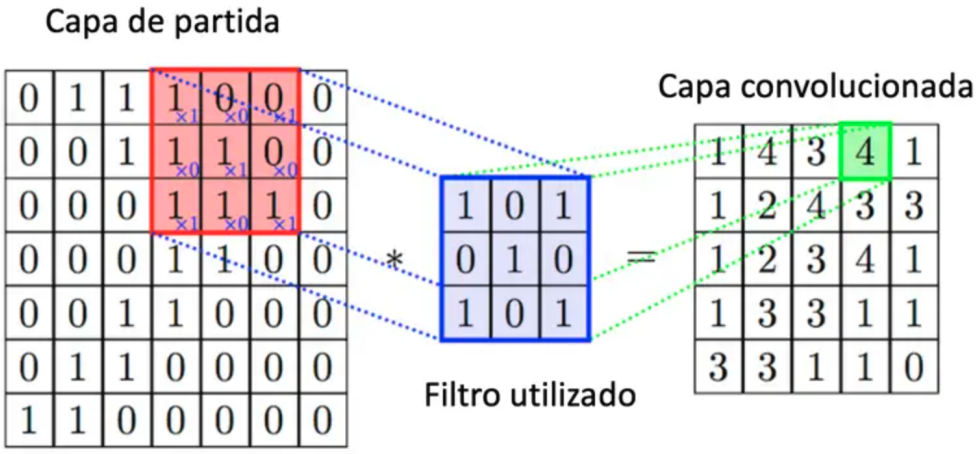
- Redes convolucionales de grafos (*Graph Convolutional Networks*, GCN)
- Redes neuronales recurrentes de grafos (*Graph Recurrent Neural Networks*, GRNN)
- Autocodificadores de grafos (*Graph Auto Encoders*, GAE)
- Redes neuronales gráficas espaciotemporales (*Spatio Temporal Graph Neural Networks*, STGNN)

A continuación, se realizará un estudio de cada una de ellas y se establecerán sus principales ventajas y desventajas, así como sus aplicaciones.

3.4.1 Redes convolucionales de grafos (GCN)

La base de las GCN es el concepto de convolución. En las redes neuronales convolucionales (CNN) tradicionales, las convoluciones se realizan sobre estructuras regulares en forma de cuadrícula, como las imágenes, en las que los filtros se aplican a vecindarios locales de píxeles (Ver figura 3.3). En las GCN, las convoluciones se adaptan para funcionar en estructuras gráficas irregulares. En lugar de operar sobre píxeles, los GCN lo hacen sobre nodos, y en lugar de considerar los vecindarios locales, tienen en cuenta los patrones de conectividad de los nodos del grafo.

Figura 3.3 - Ejemplo de Convolución



NOTA: Calvo, D. (2017, julio 20). Red Neuronal Convolucional CNN. *Diego Calvo*. https://www.diegocalvo.es/red-neuronal-convolucional/

Para realizar convoluciones de grafos, se comienza con un grafo de entrada representado por una matriz de adyacencia y una matriz de características. La matriz de adyacencia captura la información de conectividad del grafo, mientras que la matriz de características contiene las características asociadas a cada nodo. Estas características pueden ser cualquier tipo de información asociada a los nodos, como atributos de nodos o incrustaciones.

La operación de convolución del grafo consiste en propagar la información de los nodos vecinos para actualizar las representaciones de los nodos. Para ello, se multiplica la matriz de características por la matriz de adyacencia, agregando así la información de los nodos adyacentes. Sin embargo, aplicar esta operación una sola vez puede no capturar toda la información del vecindario, especialmente en arquitecturas profundas. Para solucionar este problema, las GCN suelen apilar varias capas de convolución de grafos para capturar la información de vecindarios cada vez más amplios.

Además de la convolución basada en la matriz de adyacencia, las GCN también incorporan un paso de normalización para abordar el problema de los distintos grados de los nodos. Este paso garantiza que la información propagada a cada nodo se escala en función del número de vecinos que tenga. Una técnica de normalización muy utilizada es la normalización simétrica, en la que la matriz de adyacencia se normaliza por sus sumas de filas y columnas.

Así mismo, además de la operación básica de convolución de grafos, los GCN pueden mejorarse con diversas técnicas para aumentar su capacidad expresiva. Una de ellas es el uso de funciones de activación, que introducen no linealidad en la red y le permiten modelar relaciones complejas. Entre las funciones de activación más utilizadas en las GCN se encuentran la unidad lineal rectificada (*Rectified Lineal Unit*, ReLU) y la unidad lineal exponencial (*Exponential Linear Unit*, ELU) [21].

Otra mejora es la incorporación de operaciones de agrupamiento y desagrupamiento de grafos, similares a las operaciones de agrupamiento de las CNN. El pooling [22] permite a la red agregar información de un subconjunto de nodos, lo que reduce el tamaño del grafo y aumenta la eficiencia computacional. Por su parte, la desagrupación permite aumentar el tamaño del grafo duplicando nodos en función de determinados criterios.

Por otro lado, para mejorar la capacidad de representación de las GCN, se pueden añadir conexiones de salto entre distintas capas. Estas conexiones permiten a la red retener información de capas anteriores y propagarla a capas más profundas, lo que facilita el flujo de información y resuelve el problema del gradiente de fuga [23].

El entrenamiento de las GCN suele implicar la definición de una función de pérdida adecuada que capte el objetivo de la tarea en cuestión. Para la clasificación de nodos, una función de pérdida común es la pérdida de entropía cruzada, que compara las etiquetas de nodos predichas con las etiquetas reales. A continuación, los parámetros del GCN se optimizan mediante técnicas como la retropropagación y el descenso de gradiente [24].

Los GCN han demostrado un rendimiento prometedor en diversos ámbitos. Se han aplicado con éxito al análisis de redes sociales, sistemas de recomendación, bioinformática y muchas otras áreas en las que prevalecen los datos estructurados. Tienen la capacidad de aprovechar tanto la información topológica como las características de los nodos para realizar predicciones, lo que los hace especialmente útiles para tareas relacionadas con datos estructurados en grafos.

3.4.2 Redes neuronales recurrentes de grafos (GRNN)

Las redes neuronales recurrentes basadas en grafos (*Graph Recurrent Neural Networks*, GRNN) son un tipo de arquitectura de red neuronal diseñada específicamente para procesar y modelar datos secuenciales en dominios estructurados en grafos. Mientras que las redes neuronales recurrentes (RNN) tradicionales destacan en el tratamiento de datos secuenciales de forma lineal, las GRNN amplían esta capacidad a los datos estructurados en grafos, en los que los nodos están conectados de forma no lineal y potencialmente cíclica.

En las GRNN se incorporan conexiones recurrentes que permiten que la información fluya por el grafo a lo largo del tiempo. Cada nodo del grafo mantiene su propio estado oculto, que puede considerarse como una célula de memoria que retiene información sobre sus estados anteriores y los estados de sus nodos vecinos. Estos estados ocultos se actualizan de forma iterativa, propagando la información a través de las aristas del grafo y capturando la dinámica temporal del proceso subyacente.

Para realizar cálculos en las GRNN, cada nodo tiene una unidad recurrente asociada, como una unidad recurrente con puerta (*Gated Recurrent Units*, GRU) o una unidad de memoria a corto plazo (*Long Short-Term Memory*, LSTM) [25]. Estas unidades recurrentes actualizan el estado oculto de cada nodo basándose en su estado actual, en los estados de sus vecinos y, potencialmente, en información adicional, como los atributos de los nodos o los pesos de las aristas. Las reglas de actualización específicas de las unidades recurrentes determinan cómo se combina y transforma la información en cada nodo.

La propagación de la información en las GRNN sigue un orden secuencial, en el que los estados de los nodos se actualizan uno tras otro en función de la estructura del grafo. Este proceso puede verse como una serie de iteraciones, donde los estados ocultos de los nodos se actualizan de forma recurrente hasta la convergencia o un número predefinido de pasos. Estas, también se benefician de la capacidad de incorporar entradas u observaciones externas en cada paso temporal. Las entradas pueden utilizarse para proporcionar un contexto u orientación adicionales a la red durante el procesamiento secuencial. Por ejemplo, en el contexto del análisis de redes sociales, las entradas externas podrían representar información textual o numérica asociada a cada usuario de la red.

El entrenamiento de las GRNN consiste en optimizar los parámetros de la red para minimizar una función de pérdida. Dicha función de pérdida depende de la tarea específica que se aborde, como la clasificación de nodos, la predicción de enlaces o la predicción a nivel de grafo. Para actualizar los parámetros de la red y ajustar las conexiones recurrentes se pueden emplear técnicas de optimización, como la retropropagación o el descenso de gradiente.

Las GRNN han encontrado aplicaciones en varios dominios en los que los datos secuenciales se representan de forma natural como grafos. Se han aplicado con éxito a tareas como el análisis dinámico de redes sociales, la predicción de series temporales sobre datos estructurados y el modelado espacio-temporal en redes de transporte o de sensores. Al considerar explícitamente la estructura del grafo y capturar las dependencias temporales, las GRNN proporcionan un potente marco para analizar y predecir procesos secuenciales en datos estructurados en grafos.

3.4.3 Autocodificadores de grafos (GAE)

Los autocodificadores de grafos (*Graph Auto Encoders*, GAE) son una clase de modelos de redes neuronales diseñados específicamente para el aprendizaje no supervisado de datos estructurados en grafos. Inspirados en los autocodificadores tradicionales, los GAE pretenden aprender una representación latente de baja dimensión de los nodos de un grafo, capturando patrones significativos e información estructural inherente a los datos.

El objetivo principal de un GAE es reconstruir el grafo de entrada mediante un marco codificador-decodificador [26]. El codificador asigna cada nodo del grafo a un espacio latente de dimensiones inferiores, mientras que el decodificador reconstruye el grafo a partir de las representaciones latentes aprendidas. Al obligar al modelo a reconstruir el grafo de entrada con precisión, los GAE aprenden a captar las características más destacadas y la estructura subyacente de los datos.

La arquitectura codificador-decodificador de los GAE consta de dos componentes principales: el codificador y el decodificador de grafos. El codificador de grafos toma como entrada la matriz de adyacencia del grafo y la matriz de características, que representan la conectividad y la información de atributos de los nodos, respectivamente. El codificador suele consistir en varias capas de operaciones convolucionales de grafos, que agregan información de los nodos vecinos y

actualizan las representaciones de los nodos. El objetivo de este proceso es aprender una representación latente compacta y significativa de cada nodo.

Una vez codificado el grafo en un espacio de dimensiones inferiores, el decodificador reconstruye el grafo generando una matriz de adyacencia prevista. El decodificador también incorpora las representaciones latentes aprendidas de los nodos. El proceso de reconstrucción se consigue mediante una combinación de técnicas de factorización de matrices seguidas de transformaciones no lineales para obtener la matriz de adyacencia de predicción final.

Al igual que para los casos anteriores, para entrenar los GAE, se define una función de pérdida para medir la diferencia entre la matriz de adyacencia predicha y la matriz de adyacencia real del grafo de entrada. Esta función de pérdida ayuda al modelo a reconstruir el grafo con la mayor precisión posible. Entre las funciones de pérdida más utilizadas se encuentran la pérdida de entropía cruzada binaria o el error cuadrático medio, dependiendo de la naturaleza de los datos del gráfico.

Los GAE pueden mejorarse aún más incorporando técnicas de regularización para evitar el sobreajuste y mejorar la capacidad de generalización del modelo. Pueden aplicarse técnicas de regularización como el dropout o la regularización L2 [28] a los parámetros del modelo para controlar la complejidad y evitar una dependencia excesiva de nodos o características específicas. Además, los GAE también pueden ampliarse para manejar tareas como la clasificación de nodos, la predicción de enlaces o la detección de anomalías. Aprovechando las representaciones latentes aprendidas, los GAE pueden generalizarse eficazmente a datos de grafos desconocidos y obtener buenos resultados en tareas posteriores.

Así mismo, los GAE pueden adaptarse para manejar varios tipos de datos de grafos, incluidos grafos dirigidos, grafos ponderados o grafos con nodos. Modificando la arquitectura o incorporando información adicional, los GAE pueden adaptarse a diferentes estructuras y características de los grafos.

3.4.4 Redes neuronales gráficas espaciotemporales (STGNN)

Las redes neuronales de grafos espacio-temporales (*Spatio-Temporal Graph Neural Networks*, STGNN) son una clase de modelos de redes neuronales diseñados específicamente para modelar y analizar datos que presentan dependencias espaciales y temporales en dominios estructurados en grafos. Las

STGNN combinan la potencia de las redes neuronales de grafos (GNN) con la capacidad de captar la dinámica temporal, lo que permite modelar y predecir con eficacia procesos espacio-temporales complejos.

Las STGNN resultan especialmente útiles cuando se trabaja con datos en los que las relaciones espaciales y temporales son cruciales, como el flujo de tráfico, los patrones meteorológicos, las redes sociales o la movilidad humana. En estos ámbitos, los datos subyacentes pueden representarse como grafos, en los que los nodos corresponden a ubicaciones espaciales o entidades, y las aristas representan las relaciones o interacciones entre ellas.

La idea clave de las STGNN es incorporar información espacial y temporal en la arquitectura del modelo. Esto se consigue normalmente mediante una combinación de operaciones convolucionales gráficas para captar las dependencias espaciales y operaciones convolucionales recurrentes o temporales para captar las dependencias temporales. En cada paso temporal, se toma como entrada un grafo con las características espaciales de cada nodo y la información temporal asociada al paso temporal actual. Las características espaciales pueden representar atributos, incrustaciones o medidas asociadas a cada nodo, mientras que la información temporal pueden ser datos de series temporales u observaciones secuenciales.

Por un lado, el componente de modelado espacial de las STGNN consiste en aplicar convoluciones de grafos para propagar la información a través del dominio espacial. Las convoluciones de grafos capturan las relaciones entre nodos vecinos, lo que permite al modelo capturar dependencias y patrones espaciales. Esta operación agrega información de los nodos vecinos y actualiza las representaciones de los nodos, proporcionando un contexto espacial para el posterior modelado temporal.

Por otro lado, el componente de modelado temporal de las STGNN se centra en capturar la dinámica temporal y los patrones de los datos. Esto se consigue mediante redes neuronales recurrentes (RNN) o mediante operaciones convolucionales temporales. Estas operaciones procesan las secuencias temporales o las observaciones asociadas a cada nodo, lo que permite al modelo captar las dependencias temporales y los patrones de los datos.

Los resultados de los componentes de modelado espacial y temporal se combinan para formar la representación final de cada nodo en cada paso temporal. Esta

representación combinada captura tanto la información espacial como la temporal y proporciona una comprensión global de la dinámica espacio-temporal de los datos.

El entrenamiento de las STGNN se realiza a través de una función de pérdida que capta el objetivo de la tarea en cuestión, como la predicción o la clasificación espacio-temporal. A continuación, los parámetros de la STGNN se optimizan mediante técnicas como la retropropagación y el descenso gradiente permitiendo modelar y predecir con precisión procesos espacio-temporales complejos al incorporar eficazmente dependencias espaciales y temporales en un marco unificado.

Por último, las STGNN han demostrado un éxito significativo en varios dominios que implican datos espacio-temporales. Se han aplicado a tareas como la predicción del tráfico, el reconocimiento de la actividad humana, la modelización del clima y la predicción de epidemias.

Capítulo 4
Diseño e Implementación

Una vez conocido el estado del arte, y entendido los distintos tipos de redes neuronales basadas en grafos, se ha procedido al diseño e implementación de una prueba de concepto (*Proof of Concept*, PoC) en la que se apliquen las técnicas estudiadas y se evalúen de forma práctica.

En primer lugar, se han estudiado las distintas tecnologías y principales *frameworks* capaces de implementar modelos de D*eep Learning* y tratar con datos estructurados en grafos. Tras ello, se han seleccionado las librerías que mejor se adapten a la prueba de concepto seleccionada y se ha procedido a la implementación de la misma.

4.1 Conceptos Tecnológicos, CPU vs GPU

A la hora de entrenar redes neuronales sobre grafos, el uso de una GPU (*Graphics Processing Unit*, unidad de procesamiento gráfico) ofrece varias ventajas sobre una CPU (*Central Processing Unit*, unidad central de procesamiento). Las GNN suelen implicar operaciones de alta carga computacional sobre grandes grafos, por lo que las GPU son idóneas para manejar el procesamiento paralelo necesario para este tipo de tareas.

Una de las principales ventajas de utilizar una GPU es su capacidad de paralelización. El entrenamiento de GNN implica el procesamiento simultáneo de múltiples nodos y aristas, lo que puede realizarse de forma paralela y eficiente en una GPU. Al utilizar los miles de núcleos disponibles en una GPU, los cálculos pueden realizarse de forma simultánea, lo que se traduce en una considerable aceleración en comparación con una CPU. Además, las GNN implican operaciones matriciales, como multiplicaciones y convoluciones de matrices, que son muy complejas desde el punto de vista computacional. Las GPU destacan en la ejecución de estas operaciones gracias a su arquitectura optimizada para el cálculo de matrices. Pueden aprovechar el procesamiento paralelo para acelerar estas operaciones y manejar matrices de gran tamaño de forma eficiente.

Otra ventaja de las GPU es su gran ancho de banda de memoria. El entrenamiento de GNN implica un movimiento frecuente de datos entre el procesador y la memoria. Las GPU tienen un gran ancho de banda de memoria, lo que posibilita el manejo de las ingentes cantidades de datos necesarias para el entrenamiento de las GNN. Esto acelera la transferencia de datos y reduce los cuellos de botella asociados al acceso a la memoria. Así mismo, los marcos de deep learning más conocidos, como TensorFlow y PyTorch, proporcionan implementaciones optimizadas para GNN aceleradas en la GPU, lo que permite una integración perfecta con las GPU y facilita el aprovechamiento de su potencia de cálculo.

Cabe destacar que, aunque las GPU ofrecen importantes ventajas para el entrenamiento de GNN, las CPU siguen desempeñando un papel crucial en el proceso de entrenamiento. Las CPU suelen utilizarse para el pre-procesamiento de datos, la configuración de modelos y la gestión de los procesos de entrenamiento. Además, se pueden utilizar sistemas híbridos CPU-GPU para explotar los puntos fuertes de ambos procesadores y conseguir un flujo de trabajo de entrenamiento más eficiente y equilibrado.

4.1.1 NVIDIA CUDA

CUDA (*Compute Unified Device Architecture*) [28] es una plataforma de cálculo paralelo y un modelo de programación desarrollado por NVIDIA. Permite a los desarrolladores aprovechar la potencia de las GPU NVIDIA para tareas de cálculo de propósito general, lo que incluye acelerar cargas de trabajo de alta carga computacional como el aprendizaje profundo, las simulaciones científicas y el procesamiento de datos.

CUDA proporciona una interfaz de programación y un entorno de ejecución que permite a los desarrolladores escribir aplicaciones aceleradas en la GPU utilizando lenguajes de programación estándar como Python, C++ o Fortran. Con CUDA, los desarrolladores pueden descargar en la GPU partes del código que requieren un uso intensivo de recursos computacionales, aprovechando su capacidad de procesamiento paralelo y su elevado ancho de banda de memoria.

Entre las principales características de CUDA se incluyen:

1. Modelo de cálculo paralelo: CUDA introduce un modelo de cálculo paralelo que permite a los desarrolladores definir explícitamente el paralelismo en sus aplicaciones. Al dividir las tareas en múltiples subprocesos paralelos que

pueden ejecutarse de forma simultánea, CUDA permite utilizar eficientemente la capacidad de procesamiento de la GPU.

2. Gestión de la memoria de la GPU: CUDA proporciona mecanismos para gestionar la memoria en la GPU. Incluye memorias independientes, como la memoria global, la memoria compartida y la memoria constante, a las que pueden acceder los subprocesos de la GPU. Los desarrolladores pueden gestionar el movimiento de datos entre los espacios de memoria de la CPU y la GPU para optimizar el rendimiento.
3. Jerarquía de subprocesos: CUDA introduce una organización jerárquica de los subprocesos para utilizar de forma eficiente los recursos computacionales de la GPU. Es posible organizar los subprocesos en cuadrículas, bloques e hilos, lo que permite un control detallado de la ejecución paralela. Esta jerarquía permite una gran coordinación y sincronización de los subprocesos dentro de un núcleo de GPU.
4. Bibliotecas CUDA: NVIDIA proporciona una colección de librerías optimizadas que aprovechan CUDA para diversos dominios, como cuBLAS para cálculos de álgebra lineal, cuDNN para redes neuronales profundas y cuFFT para transformadas rápidas de Fourier. Estas librerías ofrecen implementaciones de alto rendimiento, aceleradas por GPU, de algoritmos y funciones comunes, lo que permite acelerar aplicaciones sin necesidad de implementar código de bajo nivel en la unidad gráfica.
5. Herramientas de desarrollo: CUDA incluye un conjunto de herramientas de desarrollo que facilitan el desarrollo de aplicaciones aceleradas por GPU. Esto incluye el CUDA Toolkit, que proporciona compiladores, depuradores, perfiladores y herramientas de análisis del rendimiento. Estas herramientas ayudan a los desarrolladores a optimizar el programa implementado, identificar cuellos de botella y garantizar una utilización eficiente de la GPU.

CUDA se ha generalizado en los campos de la computación científica, el aprendizaje automático y el cálculo de alto rendimiento gracias a su capacidad para aprovechar la capacidad de procesamiento paralelo de las GPU. Se ha convertido en una tecnología esencial para acelerar cargas de trabajo de alta carga computacional y ha permitido importantes avances en diversos campos.

4.2 Librerías de Aprendizaje Profundo

4.2.1 Tensor Flow

TensorFlow es un *framework* de *Deep Learning* de código abierto que ha ganado gran popularidad gracias a su flexibilidad, escalabilidad y amplio conjunto de herramientas. Proporciona un ecosistema flexible y eficiente para construir y desplegar varios modelos de aprendizaje automático, incluidas las redes neuronales.

Uno de los puntos fuertes de TensorFlow es su abstracción de grafos computacionales. TensorFlow representa los cálculos como un grafo dirigido, donde los nodos representan operaciones, como cálculos matemáticos o manipulaciones de datos, y los bordes representan el flujo de datos entre estas operaciones. Este enfoque basado en grafos permite un paralelismo eficiente entre la diferenciación automática y la optimización, lo que se traduce en cálculos de alto rendimiento.

TensorFlow ofrece un gran conjunto de APIs y herramientas para apoyar el desarrollo de modelos de aprendizaje automático. Proporciona APIs de bajo nivel, como TensorFlow Core, que permiten a los usuarios definir y ejecutar cálculos de forma detallada, dando un control total sobre la arquitectura del modelo y el proceso de entrenamiento. Por otro lado, TensorFlow también ofrece APIs de alto nivel como Keras, que proporcionan una interfaz más fácil de usar e intuitiva para construir y entrenar modelos con menos código repetitivo.

Además, TensorFlow ofrece soporte para computación distribuida, permitiendo escalar sus modelos y procesos de entrenamiento a través de múltiples máquinas o GPUs. Incluye herramientas como TensorFlow Distributed, TensorFlow Cluster y TensorFlow Estimator, que simplifican la distribución y coordinación de los cálculos, permitiendo manejar conjuntos de datos más grandes y entrenar modelos más complejos de manera eficiente.

Por último, cabe destacar su compatibilidad con diferentes plataformas y escenarios de despliegue. TensorFlow es compatible con varios aceleradores de hardware, como CUDA, y chips especializados como TPU (*Tensor Processing Units*), lo que permite aprovechar las optimizaciones específicas del hardware.

4.2.2 PyTorch y PyTorch Geometric

PyTorch Geometric (PyG) [29] es una potente librería diseñada específicamente para implementar Redes Neuronales sobre Grafos (GNN) utilizando PyTorch, un framework de aprendizaje profundo ampliamente utilizado. PyG proporciona un gran conjunto de herramientas, funcionalidades que facilitan en gran medida el desarrollo de modelos con datos estructurados en grafos.

PyG proporciona una API que permite el manejo de grafos. Además, ofrece una gran versatilidad a la hora de cargar datos y utilidades para preprocesar, manipular y transformar conjuntos de datos estructurados en grafos. Esto incluye métodos para manejar diferentes formatos de grafos, cargar atributos de nodos y aristas y dividir los datos en conjuntos de entrenamiento, validación y prueba.

La funcionalidad principal de PyG reside en la implementación de varias capas convolucionales de grafos. Soporta una amplia gama de arquitecturas GNN populares, incluyendo redes convolucionales de grafos (*Graph Convolutional Networks*, GCN), redes de atención de grafos (*Graph Attention Networks*, GATs), GraphSAGE y redes de isomorfismo de grafos (*Graph Isomorphism Networks*, GINs), entre otras. Estas capas permiten el paso eficiente de mensajes y la agregación de información a través del grafo, capturando las dependencias relacionales entre nodos. Además, aprovecha la aceleración por GPU de PyTorch, lo que permite un entrenamiento y una inferencia eficientes de los modelos GNN en conjuntos de datos de grafos a gran escala.

Una de sus principales ventajas es la existencia de una comunidad activa y solidaria. La documentación oficial proporciona explicaciones detalladas, tutoriales y ejemplos que ayudan a los usuarios a entender e implementar GNNs con PyG. Además, la comunidad contribuye activamente a la biblioteca, mejorando continuamente su funcionalidad, rendimiento y ampliando sus capacidades.

PyTorch Geometric ha ganado una popularidad significativa en la comunidad de investigación GNN debido a su facilidad de uso, flexibilidad y compatibilidad con PyTorch. Su API bien diseñada, su amplia colección de conjuntos de datos y sus modelos predefinidos la convierten en una valiosa herramienta tanto para investigadores como para profesionales que trabajan con datos estructurados en grafos.

4.2.3 Deep Graph Library

Deep Graph Library (DGL) es una biblioteca diseñada específicamente para implementar redes neuronales sobre grafos (GNN) en múltiples marcos de *Deep Learning*, incluidos PyTorch y TensorFlow. DGL proporciona un amplio conjunto de herramientas y funcionalidades de manipulación de grafos, lo que la convierte en una opción popular para trabajar con datos estructurados en grafos.

Uno de los puntos fuertes de DGL es su capacidad para soportar múltiples marcos de *Deep Learning*. Esta flexibilidad permite aprovechar las ventajas del marco que se desee a la vez que se aprovechan las capacidades de procesamiento de grafos de DGL. Además, ofrece una interfaz de alto nivel que simplifica la implementación de modelos GNN. Proporciona una gran variedad de capas convolucionales preimplementadas y operaciones de agrupación, lo que permite construir fácilmente arquitecturas GNN complejas. Así mismo, DGL admite predicciones tanto a nivel de grafo como de nodo, lo que lo hace adecuado para una amplia gama de tareas relacionadas con grafos.

Por otro lado, otra de las principales características de DGL es su capacidad para manejar grafos a gran escala. Gracias al uso de técnicas de muestreo y procesamiento por lotes de grafos, esenciales para procesar grafos con millones, o incluso miles de millones de nodos y aristas. También admite el entrenamiento distribuido, lo que permite escalar los modelos GNN en varias máquinas o GPU para mejorar el rendimiento.

Por último, DGL incluye una amplia colección de algoritmos y utilidades de grafos que permiten a los usuarios realizar tareas de análisis de grafos que van más allá de las GNN. Ofrece funciones de agrupamiento de grafos, cálculo de similitud de grafos, recorrido de grafos y extracción de subgrafos. Estas utilidades permiten explorar y analizar eficientemente las estructuras y propiedades de los grafos.

4.2.4 StellarGraph

StellarGraph es una biblioteca de Python de código abierto que proporciona una interfaz intuitiva para construir y entrenar GNN en datos estructurados en grafos a gran escala. Destaca por su enfoque para manejar distintos tipos de datos, admitiendo varios tipos de grafos, incluidos los grafos homogéneos, los heterogéneos y los temporales. Esta versatilidad permite aplicar las GNN a una

amplia gama de escenarios del mundo real, como grafos de conocimiento y sistemas de recomendación.

Al igual que otras librerías, StellarGraph proporciona una API que simplifica el proceso de carga, preprocesamiento y manipulación de datos. Ofrece estructuras de datos y algoritmos específicamente diseñados para el cálculo sobre grafos. También incluye utilidades para el procesamiento de datos, la visualización y el muestreo de grafos, lo que permite analizar y procesar eficazmente grafos de gran tamaño.

La biblioteca admite diversas arquitecturas GNN, como las redes convolucionales de grafos o GraphSAGE. Estas arquitecturas se implementan utilizando TensorFlow o PyTorch, lo que permite aprovechar las capacidades computacionales y el ecosistema de estos *frameworks* de *Deep Learning*. Cabe destacar que, el diseño modular de StellarGraph facilita la personalización y ampliación de estas arquitecturas para adaptarlas a requisitos de modelado específicos. También admite el entrenamiento distribuido, lo que permite entrenar modelos GNN en clusters de máquinas o GPU, mejorando la escalabilidad y el rendimiento.

Por último, StellarGraph también ofrece utilidades de preprocesamiento para tareas relacionadas con grafos, como la predicción de enlaces, la clasificación de nodos, la agrupación de grafos y la incrustación de grafos.

4.2.5 Spektral

Spektral es una biblioteca de Python que se centra en simplificar el desarrollo y la implementación de redes neuronales sobre grafos (GNN) utilizando el *framework* Keras de *Deep Learning*. Una de las características clave de Spektral es su amplio soporte para varias arquitecturas GNN. Ofrece una gran cantidad de capas y modelos preimplementados, como las redes convolucionales de grafos (GCN), las redes de atención de grafos (GAT) o GraphSAGE. Estos componentes preconstruidos permiten crear prototipos y experimentar con diferentes arquitecturas GNN, ahorrando tiempo de desarrollo.

Spektral se integra perfectamente con Keras, conocido por su simplicidad y facilidad de uso. Keras proporciona una interfaz de alto nivel y fácil de usar para construir redes neuronales, y Spektral amplía esta funcionalidad específicamente para las GNN. Esta combinación facilita a los usuarios familiarizados con Keras la transición a la construcción de modelos GNN utilizando Spektral. La biblioteca ofrece herramientas para el preprocesamiento de datos, incluyendo métodos para

cargar y manipular conjuntos de datos de grafos, así como para manejar características de nodos y aristas. Además, Spektral admite varios formatos de grafos y permite trabajar con estructuras de grafos tanto homogéneas como heterogéneas.

Al igual que en otras librerías, una de las características destacables de Spektral es que admite predicciones tanto a nivel de nodos como de grafos. Los usuarios pueden aprovechar las capacidades de la biblioteca para realizar tareas como la clasificación de nodos, la predicción de enlaces, la clasificación de grafos y la regresión de grafos.

Spektral también incorpora técnicas avanzadas para el cálculo eficiente de grafos. Aprovecha las operaciones con matrices dispersas y la aceleración de la GPU para garantizar un procesamiento rápido y eficiente en memoria de grafos a gran escala. Con ello, es posible entrenar y evaluar modelos GNN en grafos con millones o incluso miles de millones de nodos y aristas.

Capítulo 5
PoC y Resultados

En este capítulo, se presenta la prueba de concepto y los resultados obtenidos durante el estudio de técnicas de *Deep Learning* utilizando GNN para el análisis de datos. El objetivo de este capítulo es demostrar la eficacia y el rendimiento de nuestro enfoque propuesto en la resolución de problemas del mundo real utilizando GNN. Para ello, se ha empleado PyTorch Geometric como librería principal para implementar modelos GNN y se ha utilizado CUDA para aprovechar la potencia de aceleración de la GPU durante el entrenamiento. A continuación, se exponen los motivos por los qué se ha optado por PyTorch Geometric y CUDA, destacando sus beneficios y ventajas en el caso de uso.

PyTorch Geometric es una librería potente y ampliamente utilizada para implementar GNNs en el framework PyTorch. Se ha optado por PyTorch Geometric por varias razones:

1. Gran funcionalidad: PyTorch Geometric ofrece una amplia colección de operaciones, módulos y utilidades específicas para grafos, diseñadas específicamente para simplificar el desarrollo y entrenamiento de modelos GNN. La librería proporciona varios bloques de construcción para construir arquitecturas GNN, incluyendo *Graph Convolutional Networks* (GCN). Además, PyTorch Geometric incluye capacidades avanzadas de manejo de datos, lo que permite una manipulación eficiente de datos estructurados en grafos.

2. Integración perfecta con PyTorch: PyTorch Geometric se integra perfectamente con el ecosistema PyTorch, aprovechando su grafo computacional dinámico y sus capacidades de diferenciación automática. Esta integración permite aprovechar la amplia funcionalidad de PyTorch, como las operaciones tensoriales, los algoritmos de optimización y el soporte de GPU, mientras se trabaja con GNN.

3. Amplio soporte de la comunidad: PyTorch Geometric con una comunidad activa de desarrolladores, que contribuyen continuamente con nuevas características, modelos y mejoras. Esta comunidad garantiza que la biblioteca se mantenga actualizada con los últimos avances en la investigación de GNN y proporciona una gran cantidad de recursos y documentación para la formación y solución de problemas.

Por otro lado, para acelerar el entrenamiento del modelo GNN por GPU, se ha utilizado CUDA, Se ha seleccionado por las siguientes razones:

1. Paralelización en la GPU: Las GPU son procesadores altamente paralelos que destacan en la ejecución de tareas repetitivas y de alta carga computacional. Al utilizar CUDA, es posible aprovechar el paralelismo masivo de las GPU para acelerar el proceso de entrenamiento de nuestros modelos GNN. Esta aceleración de la GPU reduce considerablemente el tiempo de entrenamiento, lo que permite experimentar con conjuntos de datos más grandes y arquitecturas más complejas.

2. Operaciones tensoriales eficientes: CUDA es capaz de realizar operaciones tensoriales que explotan el paralelismo de las GPU y dan lugar a cálculos más rápidos y eficientes. Estas operaciones tensoriales aceleradas son cruciales para las multiplicaciones de matrices, las convoluciones y las operaciones con elementos que se utilizan habitualmente en el entrenamiento de GCN.

3. Amplia compatibilidad y soporte: CUDA es un software propietario de NVIDIA lo que garantiza la compatibilidad con tarjetas gráficas de la marca. Además, al ser un software de poca antigüedad cuenta con el total soporte de la empresa y una política de actualizaciones periódicas. Así mismo, cuenta con el respaldo en la comunidad de *Deep Learning* y es compatible con la mayoría de frameworks, incluido PyTorch. Esto garantiza el correcto funcionamiento de CUDA en la implementación del modelo en PyTorch Geometric.

Utilizando PyTorch Geometric para la implementación de GNN y aprovechando CUDA para la aceleración en la GPU, se pretende demostrar la eficacia y eficiencia del enfoque propuesto en el manejo de datos estructurados en grafos y lograr un rendimiento superior en términos de tiempo de entrenamiento y precisión del modelo.

A continuación, se discutirán los resultados obtenidos en los experimentos, proporcionando un análisis exhaustivo del rendimiento de los modelos GNN.

5.1 Proof of Concept

El objetivo de esta PoC es demostrar la eficacia y el rendimiento de las GNN en la resolución de problemas del mundo real utilizando el conjunto de datos de grafos de Twitch Gamers. El preprocesamiento, la construcción del modelo y el entrenamiento se ha implementado utilizando una arquitectura de red convolucional gráfica (GCN). En esta sección, se realiza una explicación exhaustiva de la prueba de concepto, incluyendo la implementación de GCN, el entrenamiento y las pruebas.

5.1.1 Preprocesamiento del conjunto de datos

Para el estudio, se ha utilizado el conjunto de datos sobre grafos de Twitch Gamers, obtenido del repositorio Stanford Network Analysis Project (SNAP) [30]. Este conjunto de datos representa una red social de jugadores de Twitch, con nodos que representan a los usuarios y aristas que representan las interacciones entre los usuarios. Para preprocesar el conjunto de datos, se han realizado varios pasos:

1. Carga de datos: el dataset se encuentra estructurado en dos ficheros txt. En el primero de ellos, se define la lista de adyacencia entre los distintos nodos del grafo. En el otro, se definen las características (*features*) de cada uno de los nodos per se.

2. Extracción de características de nodos: cada nodo del conjunto de datos de Twitch Gamers está asociado a un conjunto de características, como el número de seguidores, el número de canales seguidos y la duración media de la transmisión. Extrajimos estas características y las codificamos adecuadamente para introducirlas en nuestro modelo GCN.

3. Construcción de grafos: a partir del archivo con la lista de aristas, construimos la estructura del grafo creando la representación adecuada de la matriz de adyacencia. Esta matriz de adyacencia captura las relaciones e interacciones entre los jugadores de Twitch.

4. División entrenamiento-prueba: para evaluar el rendimiento de nuestro modelo GCN, dividimos el conjunto de datos en conjuntos de entrenamiento y de prueba. Utilizamos una división estándar de entrenamiento-prueba, reservando una parte de los datos para fines de evaluación.

5.1.2 Implementación del modelo GCN

Con los datos preprocesados, se ha procedido a construir el modelo GCN utilizando PyTorch Geometric. La implementación de la clase GCN se ha realizado en varios pasos:

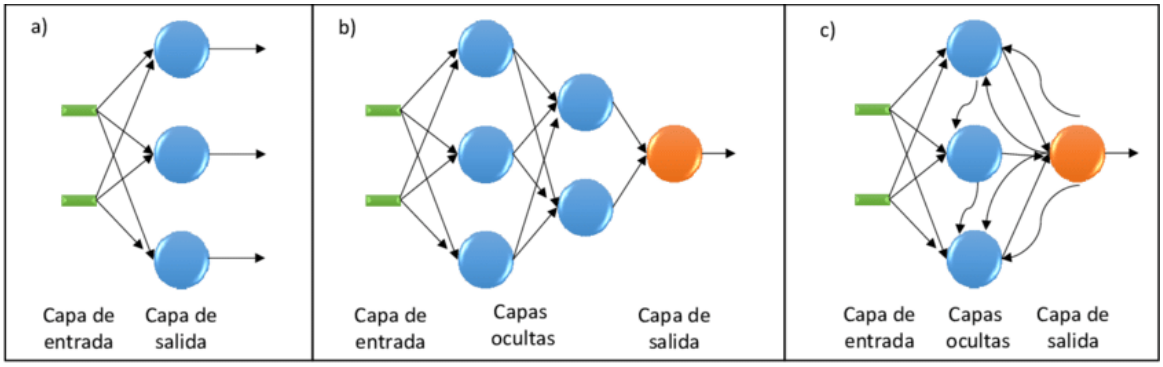
1. Inicialización del modelo: se han definido los hiperparámetros necesarios, como el número de características de entrada y salida, el número de capas ocultas y la función de activación de cada capa. Estos hiperparámetros se han elegido basándose en la experimentación y en investigaciones previas sobre arquitecturas GCN.
2. Construcción de capas: la clase GCN utilizó el módulo GCNConv de PyTorch Geometric para definir las capas del modelo. Se han implementado 3 capas, con las dimensiones de entrada y salida apropiadas, verificando que las dimensiones de los datos se ajustaban a las expectativas del modelo.
3. Propagación hacia delante: el método `forward()` de la clase GCN define la lógica de propagación hacia delante del modelo. Consiste en pasar las características de los nodos ya preprocesados a través de cada capa, aplicando la función de activación y actualizando las representaciones de los nodos en cada capa.

5.1.3 Entrenamiento y pruebas

Para entrenar el modelo GCN, se ha dividido el dataset en conjuntos de entrenamiento y testeo (*train* y *test*) y se han realizado los siguientes pasos:

1. Optimización del modelo: en este paso, se define la función de pérdida de entropía cruzada [31] para tareas de clasificación, y se selecciona un optimizador, en este caso Adam [32], para actualizar los parámetros del modelo durante el entrenamiento.
2. Iteraciones de *epochs*: se ejecuta un bucle durante un número determinado de épocas. Cada época consiste en múltiples iteraciones sobre los datos de entrenamiento. Durante cada iteración, se realiza la propagación hacia delante, calculando la pérdida y propagando hacia atrás los gradientes para actualizar los parámetros del modelo (figura 5.1).

Figura 5.1 - Ejemplo de propagación hacia delante y hacia atrás



NOTA: Manjarrez, L. (2014). *Relaciones Neuronales Para Determinar la Atenuación del Valor de la Aceleración Máxima en Superficie de Sitios en Roca Para Zonas de Subducción.*

3. Evaluación del modelo: tras el entrenamiento, se ha evaluado el rendimiento del modelo GCN entrenado con el conjunto de datos de prueba. Para ello se ha aplicado el modelo entrenado a los datos de testeo y calculado las métricas de evaluación relevantes, como el *accuracy*, la *precision*, el *recall* y el *F1-Score*. Con ello, se obtiene el rendimiento general del modelo.

5.2 Resultados y Análisis

Los resultados obtenidos de los experimentos proporcionan información sobre la eficacia del modelo GCN en el conjunto de datos de Twitch Gamers. Observamos que el modelo GCN alcanza un rendimiento prometedor en términos de precisión de clasificación, capturando los patrones y relaciones subyacentes en la red social Twitch Gamers. El modelo demuestra su capacidad para predecir características o comportamientos importantes de los jugadores de Twitch basándose en sus interacciones y características asociadas.

A través de la experimentación, se ha descubierto que al entrenar el modelo GCN con 1200 epochs se aprecia un cambio en la tendencias de la pérdida de entropía y se obtienen resultados satisfactorios. El modelo aprendió eficazmente de la estructura del grafo y de las características de los nodos para realizar predicciones precisas. Al analizar las distintas métricas implementadas, es posible confirmar el buen funcionamiento del modelo.

Figura 5.2 - Extracto de epochs durante el entrenamiento

```
Epoch: 0, Loss: 3.703075408935547
Epoch: 1, Loss: 3.650458812713623
Epoch: 2, Loss: 3.620821714401245
Epoch: 3, Loss: 3.5994694232940674
Epoch: 4, Loss: 3.579357624053955
Epoch: 5, Loss: 3.5599303245544434
Epoch: 6, Loss: 3.5413503646850586
Epoch: 7, Loss: 3.5234768390655518
Epoch: 8, Loss: 3.5061843395233154
Epoch: 9, Loss: 3.4894297122955322
Epoch: 10, Loss: 3.472989559173584
```

Estas métricas son:

- *Accuracy*: el modelo alcanza una exactitud del 0.847607, lo que significa que clasifica de forma correcta aproximadamente el 85% de los nodos de manera precisa.
- *Precision*: la precisión obtenida es del 0.872467, lo que indica que de todas las predicciones positivas realizadas por el modelo, el 87% son verdaderamente positivas.
- *Recall*: El *recall* alcanza un valor del 0.824349, lo que implica que el modelo logra identificar y recuperar el 82% de todos los casos positivos presentes en los datos.
- *F1-Score*: el F1-Score es una métrica que se obtiene al calcular la media entre la precisión y el recall. En este caso, el valor obtenido es aproximadamente el 85% (0,848408). Un *F1-Score* alto indica un buen equilibrio entre la capacidad de clasificar correctamente los casos positivos y la capacidad de encontrar la mayoría de los casos positivos.

Figura 5.3 - Resultados de la evaluación del modelo

```
Test accuracy: 0.847607
Test precision: 0.872467
Test recall: 0.824349
Test f1_score: 0.848408
```

En resumen, la prueba de concepto y los resultados presentados en este capítulo demuestran la eficacia del uso de PyTorch Geometric y la arquitectura GCN para analizar el conjunto de datos gráficos de Twitch Gamers. A través de los pasos de preprocesamiento, la implementación del modelo GCN, el entrenamiento y las pruebas, hemos aprovechado con éxito el poder de las GNNs para capturar las relaciones y patrones inherentes en la red social de los jugadores de Twitch. Los resultados obtenidos validan el potencial de las GNN en el ámbito del análisis de datos basado en grafos, demostrando su capacidad para extraer información significativa y realizar predicciones precisas a partir de datos estructurados en grafos.

Capítulo 6
Conclusiones y Líneas Futuras

Desde la concepción del análisis de datos hasta la actualidad, los métodos para inferir conocimiento a partir del mundo que nos rodea se han ido sofisticando cada vez más. Tanto la forma de almacenar como de estudiar los datos han cambiado enormemente, y con ello la forma de entender y aprovechar la información contenida en ellos. En el pasado, los métodos estadísticos tradicionales y los algoritmos de aprendizaje automático se diseñaban principalmente para datos estructurados. Sin embargo, con la llegada de las nuevas tecnologías y la explosión de datos interconectados, surgió la necesidad de desarrollar nuevas técnicas especializadas capaces de captar y analizar las complejas relaciones inherentes a estos conjuntos de datos estructurados en grafos.

Las redes neuronales basadas en grafos han surgido como un potente marco para abordar los retos que plantean los datos estructurados en grafos. Estas aprovechan la estructura inherente de los grafos, formada por nodos y aristas, para modelar y aprender de la interconexión de los datos. Gracias a su capacidad para manejar diversos tipos de grafos y datos, las GNN han encontrado aplicaciones en muchos ámbitos. En el análisis de redes sociales, las GNN pueden descubrir estructuras comunitarias, identificar nodos influyentes y predecir el comportamiento de los usuarios. En los sistemas de recomendación, las GNN pueden aprovechar los patrones de filtrado colaborativo presentes en los grafos usuario-artículo para ofrecer recomendaciones personalizadas. Y en bioinformática, los GNN pueden ayudar en la predicción del plegamiento de proteínas, el descubrimiento de fármacos y la clasificación de enfermedades explotando las relaciones entre moléculas y proteínas.

El cambio hacia el análisis de datos basado en grafos también ha impulsado avances en las metodologías de almacenamiento y estudio. Las bases de datos basadas en grafos han ganado popularidad como medio eficaz de almacenamiento y consulta de datos estructurados en grafos, ya que ofrecen capacidades optimizadas de navegación y consulta de grafos. Además, las técnicas de visualización de grafos han evolucionado para facilitar la interpretación y exploración de estructuras gráficas complejas, lo que permite a investigadores y

analistas obtener información y comprender visualmente las relaciones presentes en los datos.

En conclusión, las técnicas de análisis de datos han evolucionado hacia enfoques basados en grafos, impulsados por la necesidad de captar y analizar las intrincadas relaciones presentes en los datos estructurados en grafos. Las GNN han surgido como un potente marco para desentrañar los patrones y el conocimiento incrustado en los grafos. Con los avances en almacenamiento, visualización, escalabilidad e interpretabilidad, las GNN están preparadas para revolucionar la forma en que entendemos e inferimos el conocimiento del mundo interconectado que nos rodea. Si aprovechamos el potencial de las GNN y ampliamos continuamente los límites de la investigación, podremos desbloquear nuevos conocimientos y descubrimientos que darán forma a nuestra comprensión de los complejos sistemas que nos rodean.

Líneas Futuras

Aunque las GNN han demostrado su potencia en el aprendizaje de datos de grafos, siguen existiendo retos debido a la complejidad de los grafos. En esta sección, se sugieren cuatro direcciones futuras para las GNN.

1. Nuevos modelos para estructuras de grafos no estudiadas: existe la necesidad de modelos de aprendizaje profundo especializados para manejar diversas estructuras de grafos, como grafos heterogéneos, redes firmadas e hipergrafos.

2. Composicionalidad de modelos existentes: la integración y composición de arquitecturas existentes, como el uso de una red convolucional de grafos como capa en diferentes modelos, requiere enfoques sistemáticos y la incorporación de conocimientos interdisciplinarios.

3. Grafos dinámicos: modelizar las características evolutivas de los grafos dinámicos, en los que nodos, aristas y características cambian con el tiempo, es un área de investigación importante. Las redes neuronales recurrentes de grafos resultan prometedoras para captar la dinámica temporal.

4. Interpretabilidad y robustez: la interpretación de modelos de aprendizaje profundo en grafos es un reto debido a la interconexión de nodos y aristas. Mejorar la interpretabilidad y la robustez es crucial, especialmente en escenarios de toma de decisiones y dominios sensibles al riesgo.

Estas líneas de investigación pretenden abordar los retos y oportunidades del aprendizaje profundo basado en grafos, allanando el camino para avanzar en la comprensión de estructuras gráficas complejas y extraer información valiosa de los datos de grafos.

Capítulo 7
Summary and Conclusions

From the conception of data analysis to the present day, methods for inferring knowledge from the world around us have become increasingly sophisticated. Both the way we store and study data has changed enormously, and with it the way we understand and exploit the information contained in it. In the past, traditional statistical methods and machine learning algorithms were primarily designed for structured data. However, with the advent of new technologies and the explosion of interconnected data, the need arose to develop new specialised techniques capable of capturing and analysing the complex relationships inherent in these graph-structured data sets.

Graph-based neural networks have emerged as a powerful framework for addressing the challenges posed by graph-structured data. They take advantage of the inherent structure of graphs, consisting of nodes and edges, to model and learn from the interconnectedness of data. Thanks to their ability to handle various types of graphs and data, GNNs have found applications in many fields. In social network analysis, GNNs can discover community structures, identify influential nodes and predict user behaviour. In recommender systems, GNNs can take advantage of collaborative filtering patterns present in user-article graphs to provide personalised recommendations. And in bioinformatics, GNNs can help in protein folding prediction, drug discovery and disease classification by exploiting molecule-protein relationships.

The shift towards graph-based data analysis has also driven advances in storage and study methodologies. Graph-based databases have gained popularity as an efficient means of storing and querying graph-structured data, as they offer optimised graph browsing and querying capabilities. In addition, graph visualisation techniques have evolved to facilitate the interpretation and exploration of complex graph structures, allowing researchers and analysts to gain insight and visually understand the relationships present in the data.

In conclusion, data analysis techniques have evolved towards graph-based approaches, driven by the need to capture and analyse the intricate relationships present in graph-structured data. GNNs have emerged as a powerful framework for unravelling the patterns and knowledge embedded in graphs. With advances in

storage, visualisation, scalability and interpretability, GNNs are poised to revolutionise the way we understand and infer knowledge about the interconnected world around us. By harnessing the potential of GNNs and continually pushing the boundaries of research, we can unlock new insights and discoveries that will shape our understanding of the complex systems around us.

Future Directions

Although GNNs have demonstrated their power in learning graph data, challenges remain due to the complexity of graphs. In this section, four future directions for GNNs are suggested.

New models for unstudied graph structures: there is a need for specialised deep learning models to handle diverse graph structures, such as heterogeneous graphs, signed networks and hypergraphs.

Compositionality of existing models: integration and composition of existing architectures, such as using a convolutional network of graphs as a layer in different models, requires systematic approaches and the incorporation of interdisciplinary expertise.

Dynamic graphs: modelling the evolutionary characteristics of dynamic graphs, where nodes, edges and features change over time, is an important area of research. Recurrent neural networks of graphs show promise for capturing temporal dynamics.

Interpretability and robustness: interpreting deep learning models in graphs is challenging due to the interconnectedness of nodes and edges. Improving interpretability and robustness is crucial, especially in decision-making scenarios and risk-sensitive domains.

These research lines aim to address the challenges and opportunities of graph-based deep learning, paving the way to advance the understanding of complex graph structures and extract valuable information from graph data.

Capítulo 8
Presupuesto

En este capítulo se propone un modelo de presupuesto en el que se detallan las horas de trabajo en función de las horas empleadas en cada fase del proyecto. Este se subdivide en 5 partes: revisión bibliográfica, estudio de antecedentes, desarrollo del código y escritura de la memoria.

8.1 Presupuesto

Fases	Horas	Precio
Revisión bibliográfica	40	25€/h
Estudio de antecedentes	30	25€/h
Desarrollo del Código	80	30€/h
Escritura de memoria	50	30€/h
Total:	**200**	**5650€**

Tabla 7.1 - Presupuesto

Apéndice: Código Destacable

A.1 Definición de la clase GCN

/** *

GCN.ipynb

** *

DESCRIPCIÓN: Definición de la clase GCN utilizando la librería pytorch

**/

```
class GCN(torch.nn.Module):
    def __init__(self):
        super(GCN, self).__init__()
        torch.manual_seed(12345)

        self.conv1 = GCNConv(dataset.num_features, 4)
        self.conv2 = GCNConv(4, 4)
        self.conv3 = GCNConv(4, 2)
        self.classifier = Linear(2, dataset.num_classes)

    def forward(self, x, edge_index):
        h = self.conv1(x, edge_index)
        h = h.tanh()
        h = self.conv2(h, edge_index)
        h = h.tanh()
        h = self.conv3(h, edge_index)
        h = h.tanh()  # Final GNN embedding space.

        # Apply a final (linear) classifier.
        out = self.classifier(h)

        return out, h
```

Bibliografía

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), Article 7553. https://doi.org/10.1038/nature14539

[2] Zhang, W., Du, Y., Yoshida, T., & Yang, Y. (2019). DeepRec: A deep neural network approach to recommendation with item embedding and weighted loss function. *Information Sciences*, *470*, 121-140. https://doi.org/10.1016/j.ins.2018.08.039

[3] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4-24. https://doi.org/10.1109/TNNLS.2020.2978386

[4] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 974-983. https://doi.org/10.1145/3219819.3219890

[5] *Artificial Intelligence: A Philosophical Introduction—Jack Copeland—Google Libros*. (s. f.). Recuperado 26 de junio de 2023, de https://books.google.es/books?hl=es&lr=&id=u-NRCgAAQBAJ&oi=fnd&pg=PR10&ots=IVHOv_6kPt&sig=zsMDLST0H9CzG3J8QABg1bk9g5M&redir_esc=y#v=onepage&q&f=false

[6] Kline, R. (2011). Cybernetics, Automata Studies, and the Dartmouth Conference on Artificial Intelligence. *IEEE Annals of the History of Computing*, *33*(4), 5-16. https://doi.org/10.1109/MAHC.2010.44

[7] Shortliffe, E. (2012). *Computer-Based Medical Consultations: MYCIN*. Elsevier.

[8] Smith, D. H., Gray, N. A. B., Nourse, J. G., & Crandell, C. W. (1981). The dendral project: Recent advances in computer- assisted structure elucidation. *Analytica Chimica Acta*, *133*(4), 471-497. https://doi.org/10.1016/S0003-2670(01)95414-5

[9] Hendler, J. (2008). Avoiding Another AI Winter. *IEEE Intelligent Systems*, *23*(02), 2-4. https://doi.org/10.1109/MIS.2008.20

[10] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, *35*(1), 53-65. https://doi.org/10.1109/MSP.2017.2765202

[11] Hudson, D. A., & Zitnick, L. (2021). Generative Adversarial Transformers. *Proceedings of the 38th International Conference on Machine Learning*, 4487-4499. https://proceedings.mlr.press/v139/hudson21a.html

[12] *Introduction to Machine Learning, Neural Networks, and Deep Learning | TVST | ARVO Journals*. (s. f.). Recuperado 28 de junio de 2023, de https://tvst.arvojournals.org/article.aspx?articleid=2762344

[13] Johansson, E. m., Dowla, F. u., & Goodman, D. m. (1991). Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems*, *02*(04), 291-301. https://doi.org/10.1142/S0129065791000261

[14] Aloysius, N., & Geetha, M. (2017). A review on deep convolutional neural networks. *2017 International Conference on Communication and Signal Processing (ICCSP)*, 0588-0592. https://doi.org/10.1109/ICCSP.2017.8286426

[15] Mandic, D., & Chambers, J. (2001). *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. Wiley. https://orca.cardiff.ac.uk/id/eprint/31639/

[16] *Accelerating applications using edge tensor processing units | Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. (s. f.). Recuperado 6 de julio de 2023, de https://dl.acm.org/doi/abs/10.1145/3458817.3476177

[17] Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N., & Fox, E. A. (2021). *Natural Language Processing Advancements By Deep Learning: A Survey* (arXiv:2003.01200). arXiv. https://doi.org/10.48550/arXiv.2003.01200

[18] Estrada, E., & Rodriguez-Velazquez, J. A. (2006). Complex Networks as Hypergraphs. *Physica A: Statistical Mechanics and its Applications*, *364*, 581-594. https://doi.org/10.1016/j.physa.2005.12.002

[19] Yang, C., Wang, C., Lu, Y., Gong, X., Shi, C., Wang, W., & Zhang, X. (2022). Few-shot Link Prediction in Dynamic Networks. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 1245-1255. https://doi.org/10.1145/3488560.3498417

[20] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4-24. https://doi.org/10.1109/TNNLS.2020.2978386

[21] Maria Cristina, G. N., Cruz Sanchez, V. G., Vergara Villegas, O. O., Nandayapa, M., Ochoa Dominguez, H. de J., & Sossa Azuela, J. H. (2021). Study of the Effect of Combining Activation Functions in a Convolutional Neural Network. *IEEE Latin America Transactions*, *19*(5), 844-852. https://doi.org/10.1109/TLA.2021.9448319

[22] Sun, M., Song, Z., Jiang, X., Pan, J., & Pang, Y. (2017). Learning Pooling for Convolutional Neural Network. *Neurocomputing*, *224*, 96-104. https://doi.org/10.1016/j.neucom.2016.10.049

[23] Wei, W., & Liu, L. (2022). Gradient Leakage Attack Resilient Deep Learning. *IEEE Transactions on Information Forensics and Security*, *17*, 303-316. https://doi.org/10.1109/TIFS.2021.3139777

[24] Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., & Talwalkar, A. (2022). *Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability* (arXiv:2103.00065). arXiv. https://doi.org/10.48550/arXiv.2103.00065

[25] Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, *2*(6), 568-576. https://doi.org/10.1109/72.97934

[26] Kaur, R., GholamHosseini, H., & Sinha, R. (2022). Skin lesion segmentation using an improved framework of encoder-decoder based convolutional neural network. *International Journal of Imaging Systems and Technology*, *32*(4), 1143-1158. https://doi.org/10.1002/ima.22699

[27] Nusrat, I., & Jang, S.-B. (2018). A Comparison of Regularization Techniques in Deep Neural Networks. *Symmetry*, *10*(11), Article 11. https://doi.org/10.3390/sym10110648

[28] Luebke, D. (2008). CUDA: Scalable parallel programming for high-performance scientific computing. *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 836-838. https://doi.org/10.1109/ISBI.2008.4541126

[29] Fey, M., & Lenssen, J. E. (2019). *Fast Graph Representation Learning with PyTorch Geometric* (arXiv:1903.02428). arXiv. https://doi.org/10.48550/arXiv.1903.02428

[30] *Stanford Large Network Dataset Collection*. (s. f.). Recuperado 6 de julio de 2023, de https://snap.stanford.edu/data/

[31] Zhang, Z., & Sabuncu, M. (2018). Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. *Advances in Neural Information Processing Systems*, *31*. https://proceedings.neurips.cc/paper_files/paper/2018/hash/f2925f97bc13ad2852a7a551802feea0-Abstract.html

[32] Tato, A., & Nkambou, R. (2018). *IMPROVING ADAM OPTIMIZER*. https://openreview.net/forum?id=HJfpZq1DM