

Máster Universitario en Inteligencia Artificial Aplicada
2023-2024

Trabajo Fin de Máster

“Bayesian Graph Neural Networks to Estimate Confidence Intervals”

Oscar Llorente Gonzalez

Tutor/es

Miguel Ángel Hombrados Herrera

Jaime Boal Martín-Larrauri

Madrid, 30/09/2024

DETECCIÓN DEL PLAGIO

La Universidad utiliza el programa **Turnitin Feedback Studio** para comparar la originalidad del trabajo entregado por cada estudiante con millones de recursos electrónicos y detecta aquellas partes del texto copiadas y pegadas. Copiar o plagiar en un TFM es considerado una **Falta Grave**, y puede conllevar la expulsión definitiva de la Universidad.



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento - No Comercial - Sin Obra Derivada**

RESUMEN

Las Redes Neuronales de Grafos (GNNs) son la tecnología más efectiva a la hora de modelar una estructura como la red de telecomunicaciones. Sin embargo, aunque precisos, estos modelos son cajas negras incapaces de dar una estimación de incertidumbre o una explicación de sus predicciones. Este trabajo combina las GNNs con Redes Neuronales Bayesianas (BNNs) para añadir la capacidad de generar intervalos de confianza, aplicando la tecnología a un producto real que optimiza la red de telecomunicaciones.

Palabras clave: Aprendizaje Profundo, Redes Neuronales de Grafos, Redes Neuronales Bayesianas, Explicabilidad

DEDICATORIA

En primer lugar, a Miguel y Jaime, ya que sin su dirección y paciencia nunca habría sido posible este trabajo. Espero que quieran seguir investigando conmigo, a pesar de las prisas con las que me han sufrido en el último tramo. A todos mis compañeros de Ericsson, ya que sin su ayuda un proyecto como este sería simplemente impensable. A mis padres, por todas las veces que han insistido, aunque sin mucho éxito, que comenzara a escribir el TFM con suficiente tiempo. Porque sin ellos, estoy seguro de que mi organización habría sido aún peor. A Anita, por darme fuerza y cuidarme, porque eres quien ha visto el final de primera mano.

ÍNDICE GENERAL

1. INTRODUCCIÓN	1
2. REVISIÓN DE LA LITERATURA	2
2.1. Redes Neuronales Convencionales	2
2.2. Redes Neuronales de Grafos (GNNs)	4
2.3. Redes Neuronales Bayesianas.	6
2.4. Explicabilidad	8
3. UPINK INTERFERENCE OPTIMIZER	12
3.1. Introducción	12
3.1.1. Arquitectura	13
3.1.2. Datos utilizados	15
3.1.3. Algoritmo de Optimización	17
3.2. New Radio - 5G	19
3.2.1. Diferencias 4G vs 5G	19
3.2.2. Datos 5G	19
3.3. Bayesian Graph Neural Network	20
3.3.1. Datos para el entrenamiento.	20
3.3.2. Balanceo en el entrenamiento.	20
3.3.3. Normalización	23
3.3.4. Creación de Batches	23
3.3.5. Modelos.	24
3.3.6. Resultados	25
3.4. Optimización Probabilística	30
3.5. Explicabilidad	31
4. INVESTIGACIÓN OPEN-SOURCE	47
4.1. Datos	47
4.2. Modelos.	48
4.3. Resultados	49
5. CONCLUSIONES	57

ÍNDICE DE FIGURAS

2.1	Red Neuronal Convencional	3
2.2	Grafo	4
2.3	Red Neuronal de Grafos (GNN) [8]	5
2.4	Red de Grafos de Atención (GAT) [9]	6
2.5	Red Neuronal normal a la izquierda y Red Neuronal Bayesiana (BNN) [11] a la derecha	7
2.6	Saliency Maps [16]	9
2.7	SmoothGrad Saliency Maps [18]	10
2.8	Class Activation Maps [20]	10
2.9	Sanity Checks for Saliency Maps [21]	11
2.10	Explicabilidad en GNNs [22]	11
3.1	Red Celular	12
3.2	<i>uplink interference</i>	13
3.3	Framework de Optimización	14
3.4	Entrenamiento GNN	16
3.5	Proceso de Optimización [28]	17
3.6	Distribución alpha 4G	21
3.7	Distribución pZeroNominalPusch 4G	21
3.8	Distribución pZeroNomPuschGrant 5G	22
3.9	Predicción normal a la izquierda vs predicción con clusters [31] a la derecha	24
3.10	Example Saliency Map	37
3.11	Example SmoothGrad	41
3.12	Example Guided Backpropagation	41
3.13	Example Deconvnet	42
3.14	Loyalty Saliency Map	42
3.15	Inverse loyalty Saliency Map	43
3.16	Loyalty SmoothGrad	43

3.17	Inverse loyalty SmoothGrad	44
3.18	Loyalty Guided Backpropagation	44
3.19	Inverse loyalty Guided Backpropagation	45
3.20	Loyalty Deconvnet	45
3.21	Inverse loyalty Deconvnet	46

ÍNDICE DE TABLAS

2.1 Notación GNN	5
3.1 Datos 4G	17
3.2 Datos 5G	19
3.3 Distribución alpha 4G	20
3.4 Distribución pZeroNominalPusch 4G	22
3.5 Distribución pZeroNomPuschGrant 5G	23
3.6 Métricas entrenamiento Crystal GNN 4G	25
3.7 Métricas validación Crystal GNN 4G	26
3.8 Métricas entrenamiento GAT GNN 4G	26
3.9 Métricas validación GAT GNN 4G	27
3.10 Métricas entrenamiento Crystal GNN 5G	27
3.11 Métricas validación Crystal GNN 5G	28
3.12 Métricas entrenamiento GAT GNN 5G	28
3.13 Métricas validación GAT GNN 5G	29
3.14 Métricas test set Crystal GNN 4G	29
3.15 Métricas test set Crystal GNN 5G	30
3.16 Optimización modelo congelado <i>issue cells</i> 4G	32
3.17 Optimización modelo congelado <i>first-hop neighbor cells</i> 4G	32
3.18 Optimización modelo congelado configuraciones 4G	32
3.19 Optimización modelo sin congelar <i>issue cells</i> 4G	33
3.20 Optimización modelo sin congelar <i>first-hop neighbor cells</i> 4G	33
3.21 Optimización modelo sin congelar configuraciones 4G	34
3.22 Optimización O6 <i>issue cells</i> 4G	34
3.23 Optimización O6 <i>first-hop neighbor cells</i> 4G	35
3.24 Optimización O6 configuraciones 4G	36
3.25 Optimización modelo congelado <i>issue cells</i> 5G	37
3.26 Optimización modelo congelado <i>first-hop neighbor cells</i> 5G	37

3.27	Optimización modelo congelado configuraciones 5G	38
3.28	Optimización modelo sin congelar <i>issue cells</i> 5G	38
3.29	Optimización modelo sin congelar <i>first-hop neighbor cells</i> 5G	39
3.30	Optimización modelo sin congelar configuraciones 5G	39
3.31	Optimización O7 <i>issue cells</i> 5G	40
3.32	Optimización O7 <i>first-hop neighbor cells</i> 5G	40
3.33	Optimización O7 configuraciones 5G	40
4.1	Datos QM9 [7]	47
4.2	Resultados GCN	50
4.3	Resultados Bayesian GCN	51
4.4	Resultados Dropout GCN	52
4.5	Resultados GAT	53
4.6	Resultados Bayesian GAT	54
4.7	Resultados Dropout GAT	55
4.8	Resultados Bayesian GAT conjunto test	56

1. INTRODUCCIÓN

En la última década, la Inteligencia Artificial ha experimentado un crecimiento exponencial, especialmente en el campo del *deep learning*, con aplicaciones en áreas tan diversas como visión artificial [1], el procesamiento de lenguaje natural (NLP) [2] o la conducción autónoma [3]. En concreto, una de las aplicaciones con mayor popularidad dentro del *deep learning* son las Graph Neural Networks (GNNs), con impacto en disciplinas en las que los datos pueden ser modelados como si fueran un grafo, como las telecomunicaciones, las finanzas o la química.

Sin embargo, uno de los problemas que continúan aún por resolver es cómo se pueden combinar el rendimiento de las redes neuronales con modelos probabilísticos. Especialmente cuando estas tecnologías son parte de un producto, el hecho de poder proporcionar confianzas y estimaciones de incertidumbre es un elemento fundamental para los clientes.

Uno de los proyectos que motiva este trabajo es precisamente un producto industrial, un optimizador de la red de telecomunicaciones que está actualmente desplegado en varios clientes. Una de las dificultades en el equipo de Ericsson responsable de este proyecto era explicar las decisiones del modelo y poder proporcionar confianzas a los clientes, estimaciones de la fiabilidad de las predicciones del modelo.

Con este objetivo en mente, surge la idea de combinar las tecnologías que se usan para modelar las redes de telecomunicaciones, las GNNs, y combinarlas con Redes Neuronales Bayesianas (BNNs) para proporcionar esas confianzas en las predicciones del modelo. En concreto, el presente trabajo resuelve las siguientes cuestiones:

- Integración de las BNNs y las GNNs para crear una solución industrial capaz de proporcionar estimaciones de incertidumbre al optimizar la red de telecomunicaciones, siendo el primer producto industrial que utilizaría esta tecnología.
- Añadir explicabilidad a esta clase de modelos, aplicando por primera vez las técnicas de explicabilidad a las Redes Neuronales Bayesianas de Grafos (BGNNs).

2. REVISIÓN DE LA LITERATURA

En este primer capítulo, se expondrá una revisión de las técnicas existentes en la literatura. En específico, se presentará un desarrollo centrado en estos cuatro bloques fundamentales:

- Introducción a las Redes Neuronales de Grafos (GNNs)
- Introducción a las Redes Neuronales Bayesianas (BNNs)
- Redes Neuronales Bayesianas de Grafos (BGNNS)
- Explicabilidad en GNNs

Por tanto, primero se explicarán brevemente las redes neuronales convencionales, y sus limitaciones a la hora de lidiar con datos que están representados en un grafo. A continuación, se explicarán en detalle las GNNs y su mecanismo para modelar la información de un grafo.

Por otro lado, al finalizar esta parte, comenzará una descripción de las diversas técnicas existentes para construir modelos probabilísticos basados en redes neuronales, detallando sus diferencias y describiendo la solución elegida para el presente trabajo, las redes neuronales bayesianas.

2.1. Redes Neuronales Convencionales

En primer lugar, es importante entender la diferencia entre las redes neuronales convencionales y las GNNs. Con este objetivo, en primer lugar se explicarán cómo funcionan las redes neuronales convencional brevemente, exponiendo los principales problemas que emergen cuando se utilizan con datos con estructura de grafo. Además, se detallarán las posibles soluciones a este problema para introducir el mecanismo que emplean las GNNs y los beneficios que muestra en comparación. Para ello, se utilizará como ejemplo la red que se muestra en la Figura 2.1, compuesta simplemente por capas lineales y activaciones no lineales, aunque lo expuesto a continuación es generalizable a otros tipos de redes como las convolucionales (CNNs) o las redes recurrentes (RNNs).

Como se puede observar en la Figura 2.1, las entradas de esta red se modelan como vectores. Específicamente, estos vectores tendrán solamente una dimensión (es importante tener en cuenta que en la práctica estos vectores suelen representarse en forma de matriz donde la primera dimensión es el batch, i.e., la el número de muestras). Por tanto, cada ejemplo que se quiera introducir como entrada para realizar una predicción con el modelo será representado como un vector de la dimensión de entrada. En la mayoría de los casos,

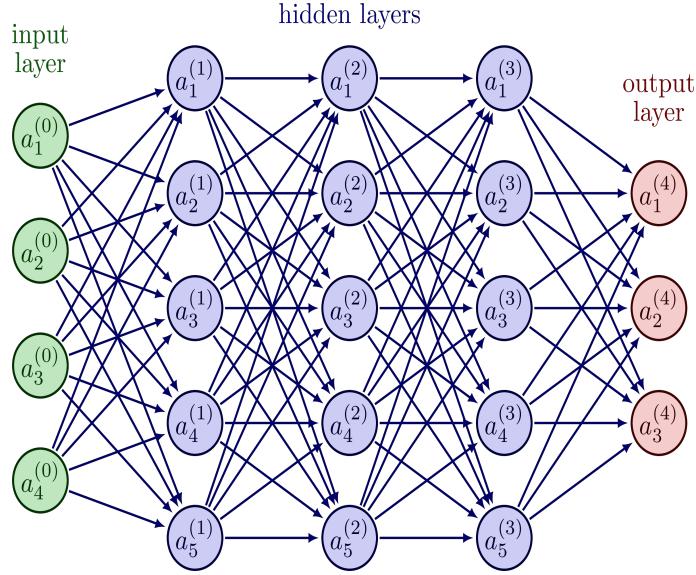


Fig. 2.1. Red Neuronal Convencional

estos vectores son independientes, por lo que no es necesario que para realizar una predicción el modelo tenga en cuenta otras entradas diferentes a la que está siendo clasificada. Sin embargo, en un grafo, como el que se muestra en la Figura 2.2, todos los puntos están conectados, lo que imposibilita el esquema que se ha explicado anteriormente. En concreto, en un grafo podemos encontrar dos tipos de tareas de predicción:

- Predicción a nivel de nodo. En este caso, cada nodo sería representado como un vector. Si se usara una red convencional para realizar la predicción en dicho nodo, no se tendría en cuenta a los nodos vecinos que pueden afectar la predicción.
- Predicción a nivel de grafo. En este caso, el grafo entero podría ser modelado como un solo vector o embedding. Sin embargo, para generar dicho embedding teniendo en cuenta las relaciones entre los nodos, sería necesario algún tipo de modelo que tuviera en cuenta la naturaleza del grafo (el hecho de que los nodos no son muestras aisladas), originando el mismo problema que en el punto anterior.

Una posible solución, utilizando las redes neuronales convencionales podría ser concatenar las representaciones del nodo que está siendo clasificado y los vecinos, originando un vector mayor que contuviera toda la información de los vecinos que afectan. Sin embargo, el problema que surge al plantear esta solución es que el número de vecinos tendría que ser un parámetro estático que no puede variar, ya que si cambia el tamaño de la red neuronal tendría también que cambiar. Por tanto, aunque podría ser una solución factible ante un número fijo de vecino, no se puede usar si el número de vecinos puede variar. Además, el coste computacional de la solución escalaría exponencialmente, con el número de vecinos, ya que cada vecino más supondría n entradas más en la primera capa de la red, siendo n el tamaño de cada vector de entrada.

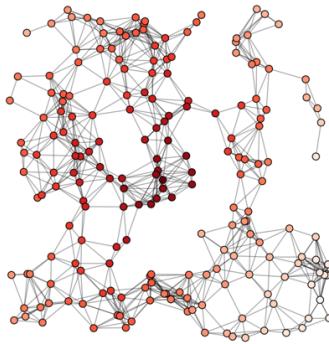


Fig. 2.2. Grafo

Una segunda solución, que resolvería el problema planteado en el párrafo anterior, sería utilizar una capa de pooling [4], con el objetivo de eliminar la dimensión de los vecinos haciendo el cómputo de la media o el máximo, o cualquier otra función de agregación que se quiera utilizar. El principal problema de este enfoque, es que la función que usamos es fija, y no depende del valor de los vecinos, i.e. siempre se realiza el cómputo de la misma función de agregación. Por tanto, si hay una relación más compleja detrás de la interacción de los vecinos del grafo, no será posible aproximarse más a ella con este enfoque.

Por las limitaciones expuestas anteriormente, se formulan las GNNs con un esquema que permite solventar los problemas descritos.

2.2. Redes Neuronales de Grafos (GNNs)

En primer lugar, antes de comenzar la descripción de la arquitectura de los modelos, y el sistema que utilizan este tipo de modelos, es importante detenerse un momento en la terminología que se va a utilizar a partir de este momento. En este trabajo se seguirá la misma notación que la expuesta en [5]:

El esquema que en el que se fundamentan las GNNs se denomina Message Passing [6], y se basa en la idea de actualizar las representaciones de los nodos (embeddings) iterativamente, en base a la agregación de la información de los vecinos, combinada con la del nodo en cuestión. Por tanto, cada capa de una GNN se divide en dos operaciones:

- Agregación: En este paso se agrega la información procedente de los vecinos.
- Combinación: En este segundo paso se combina la información agregada de los vecinos con el nodo en cuestión para actualizar la representación (embedding) de cada nodo.

Concept	Notation
Grafo	$\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Matriz de Adyacencia	$A \in \mathbb{R}^{N \times N}$
Atributos de los Nodos	$X \in \mathbb{R}^{N \times C}$
Número de capas de la GNN	K
Representaciones de los nodos en la capa k-th	$H^k \in \mathbb{R}^{N \times F}, k \in \{1, 2, \dots, K\}$

TABLA 2.1. NOTACIÓN GNN

Basándonos en la formulación que se sigue en PyTorch Geometric [7], una de las librerías de referencia para implementar esta clase de modelos, el esquema Message Passing expresado matemáticamente en la Ecuación 2.1.

$$\mathbf{x}'_i = \gamma_{\Theta} \left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}(i)} \phi_{\Theta} (\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{j,i}) \right), \quad (2.1)$$

siendo \mathbf{x}_i el embedding del nodo actual, j un vecino inmediato de i , \mathbf{x}_j el embedding del vecino j , \mathbf{e}_{ij} la arista entre los nodos i y j , \bigoplus una función diferenciable invariable a la permutación (como puede ser la suma o el máximo), y, γ_{Θ} y ϕ_{Θ} , siendo funciones diferenciables como MLPs (redes multi-capas, o por sus siglas en inglés Multilayer Perceptron, compuestas por capas lineales y activaciones no lineales). El esquema de este tipo de modelos en la Figura 2.3.

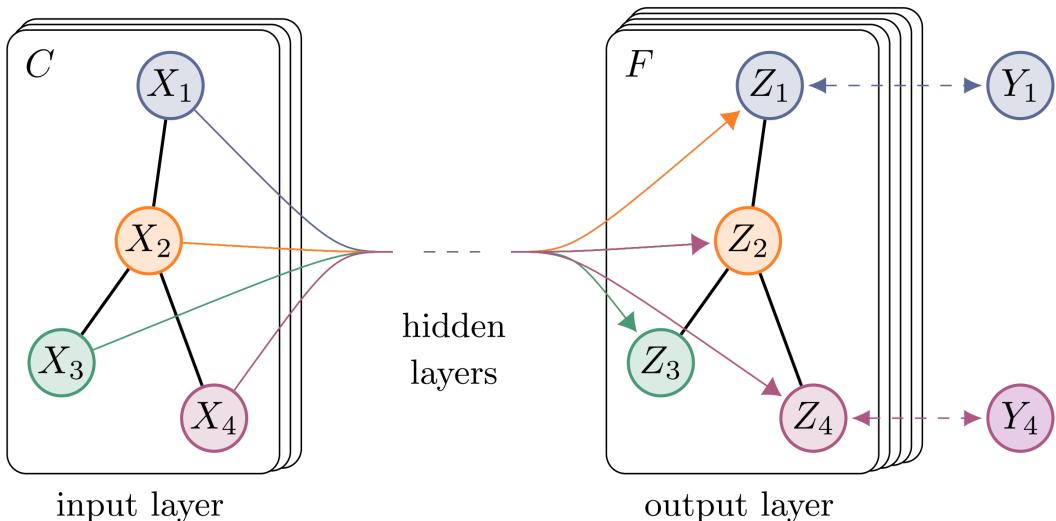


Fig. 2.3. Red Neuronal de Grafos (GNN) [8]

Con base en este esquema, múltiples tipos de modelos se han desarrollado, siendo uno

de los primeros la Red de Grafos Convolucional (GCN) [8], formulado a continuación en la Ecuación 2.2

$$H^{k+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^kW^k\right). \quad (2.2)$$

siendo $\tilde{A} = A + \mathbf{I}$ y W_k la matriz de pesos de la capa k .

Adicionalmente, en los últimos años han surgido modelos más complicados, alcanzando mejores resultados, como la Red de Grafos de Atención (GAT) [9], que se muestra en la Figura 2.4.

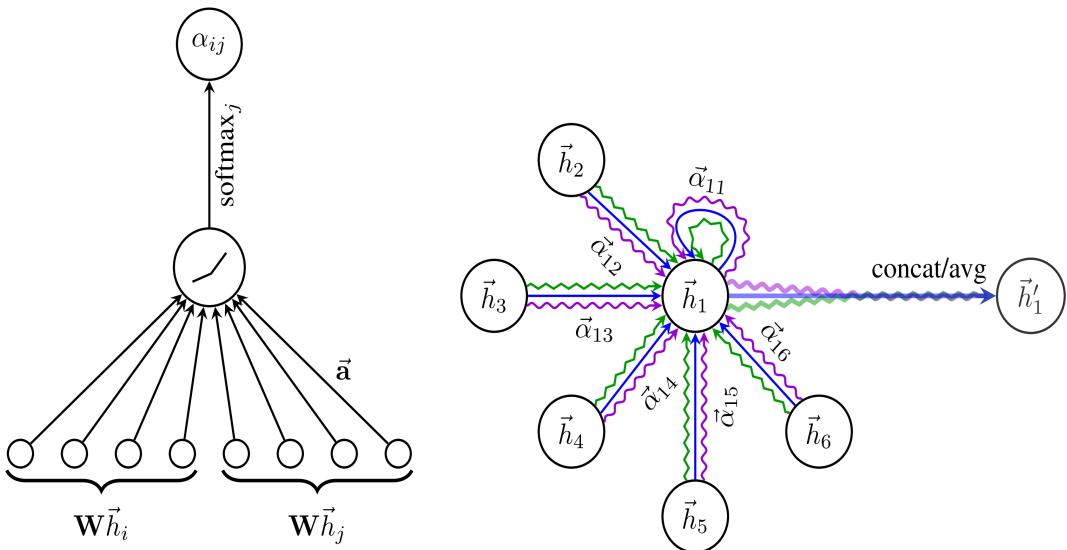


Fig. 2.4. Red de Grafos de Atención (GAT) [9]

Debido al gran número de diferentes técnicas en este dominio, no es posible hacer una revisión exhaustiva en el presente documento, pero aquí se presentan una breve lista de los diferentes modelos que se han usado en este estudio:

2.3. Redes Neuronales Bayesianas

Uno de los problemas que tienen las redes neuronales, como se menciona en la Sección 1, es que no proporcionan una estimación de la incertidumbre para lo que necesitaríamos un modelo probabilístico, como puede ser un Proceso Gausiano [10]. Una de las maneras de combinar este tipo de modelos con las clásicas redes neuronales es construir una Red Neuronal Bayesiana (BNN), sobre las que podemos encontrar una amplia literatura, como en [11].

Este tipo de redes, como se puede observar en la Figura 2.5, redefinen sus matrices de pesos para que cada peso sea ahora una variable aleatoria. En la práctica, la distribución que se suele utilizar siempre es una Gausiana, como se explicará más adelante en esta misma Sección.

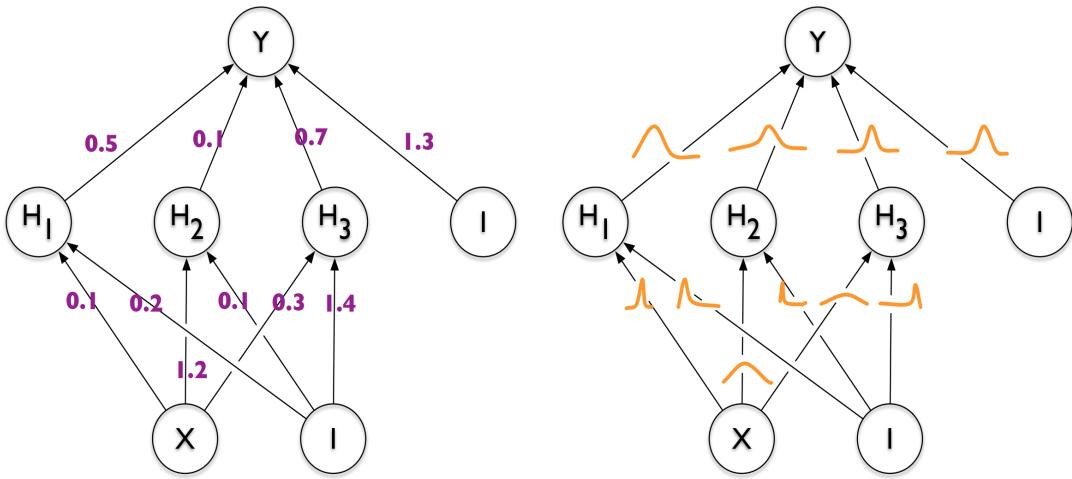


Fig. 2.5. Red Neuronal normal a la izquierda y Red Neuronal Bayesiana (BNN) [11] a la derecha

Por tanto, como se explica en [11], los pasos para entrenar esta clase de modelo serían los siguientes:

1. Muestrea $\epsilon \sim \mathcal{N}(0, I)$.
2. Sea $\mathbf{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$.
3. Sea $\theta = (\mu, \rho)$.
4. Sea $f(\mathbf{w}, \theta) = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w})P(\mathcal{D}|\mathbf{w})$.
5. Calcular el gradiente con respecto a la media

$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}. \quad (3)$$

6. Calcular el gradiente con respecto al parámetro ρ

$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}. \quad (4)$$

7. Actualizar los parámetros:

$$\mu \leftarrow \mu - \alpha \Delta_\mu \quad (5)$$

$$\rho \leftarrow \rho - \alpha \Delta_\rho. \quad (6)$$

En concreto, cada peso es una variable aleatoria y para cada predicción es muestreada de una normal y se calcula el valor del peso en cuestión con el truco de reparametrización, con el que poder realizar el cómputo del gradiente después. Por último, se calcula el gradiente de la función de pérdida, que tiene un nuevo término procedente de la divergencia Kullback-Leibler, con respecto a los parámetros que definen cada peso, en este caso la

media y un parámetro ρ para caracterizar la varianza de la distribución. Además, como se expone en [12] este tipo de modelos se pueden aproximar con una capa de Dropout.

Por otra parte, la combinación de las GNNs y las BNNs es un área inexplorada, con solo unos pocos trabajos como [13], [14] o [15]. En concreto, [13] y [14] solo utilizan esta clase de modelos para intentar mejorar los resultados de clasificación. Por otro lado, [15], aunque genera intervalos de confianza, lo hace solo de longitud fija, sin explorar intervalos de longitud variable generados a partir de las predicciones del modelo, como se realiza en el presente trabajo.

2.4. Explicabilidad

Por último, como se ha añadido al presente trabajo una sección de explicabilidad para adaptarla a las BGNN, se detalla en esta sección una revisión de la literatura de este área. En primer lugar, aunque en la última década el auge del *deep learning* ha tenido un impacto significativo en industrias de toda clase, las redes neuronales siguen teniendo un problema fundamental, y es que son cajas negras en las que no se puede saber qué está sucediendo. Por este motivo, comienzan a surgir artículos, como [16], [17], [18], [19], para intentar comprender lo que está pasando dentro de esta clase de modelos.

Uno de los primeros artículos que se publican aplicados a las arquitecturas de redes neuronales sería [16], que genera visualizaciones como las que se pueden ver en la Figura 2.6, también conocidos como *saliency maps*. En concreto, el objetivo de esta técnica sería obtener un mapa de las zonas que el modelo considera más importantes para la clasificación. Para ello, se realiza el cómputo del gradiente con respecto a la clasificación del modelo.

Sin embargo, uno de los problemas fundamentales en esta línea de investigación es el ruido que generan este tipo de técnicas, tal y como se puede observar en la Figura 2.6. Por este motivo, surgen mejoras como la propuesta en [18], que disminuye el ruido en los *saliency maps* generados, Figura 2.7.

Por otro lado, también hay otra familia de técnicas que intenta crear mapas de calor de las zonas que el modelo considera importantes. En este sentido, los trabajos más importantes son los publicados en [20] y [19]. En concreto, intentan crear dichos mapas con una estructura similar a la que puede observarse en la Figura 2.8, en la que aprovechan la arquitectura del modelo para combinar sus representaciones intermedias.

Ante la aparición de un gran número de técnicas de explicabilidad, también se pueden encontrar los primeros trabajos que estudian cómo evaluar este tipo de métodos. Entre todos ellos destaca [21], donde se cuestiona que alguna de las técnicas más utilizadas hasta el momento podrían no estar ofreciendo verdadera explicabilidad para los modelos, como se ilustra en la Figura 2.9, donde se ve que las visualizaciones que nos proporcionan los métodos no cambian a medida que alteramos los pesos de una red.

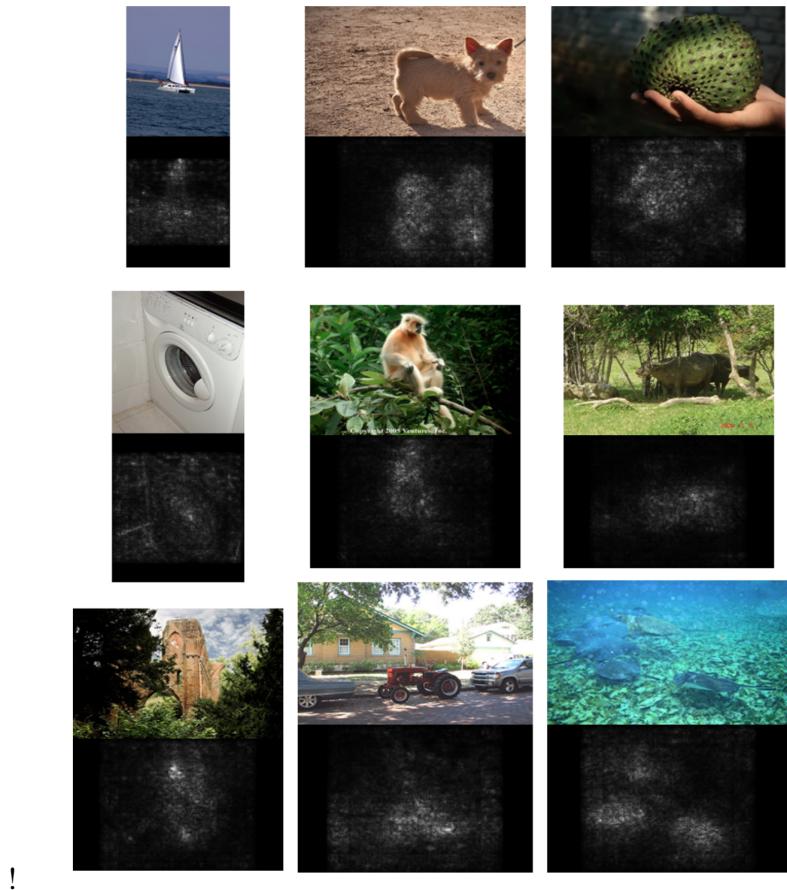


Fig. 2.6. Saliency Maps [16]

Tradicionalmente, la investigación en explicabilidad se ha aplicado al campo de visión artificial, debido seguramente la facilidad para comprobar si una visualización podía ser coherente con los datos de entrada. Sin embargo, en los últimos años también ha comenzado a aplicarse a otros campos, como las GNNs. Por tanto, se encuentran trabajos pioneros, como [22] o [23], en los que se adaptan los métodos al dominio de grafos.

Por último, a raíz de la publicación de [21] también han empezado a surgir trabajos en la línea de [24] y [25] con la intención de evaluar la fiabilidad de las técnicas en el dominio de grafos.

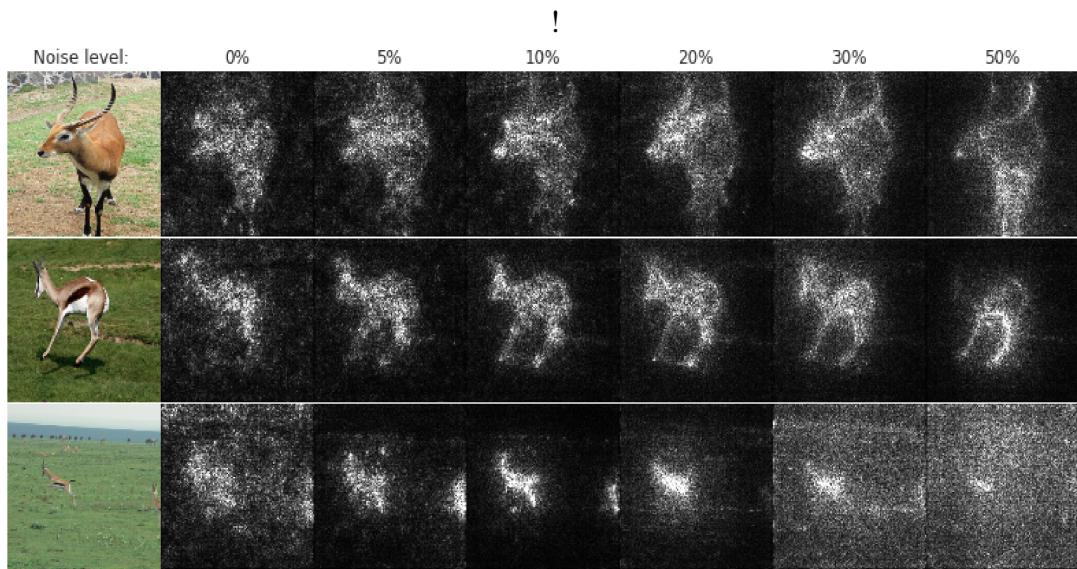


Fig. 2.7. SmoothGrad Saliency Maps [18]

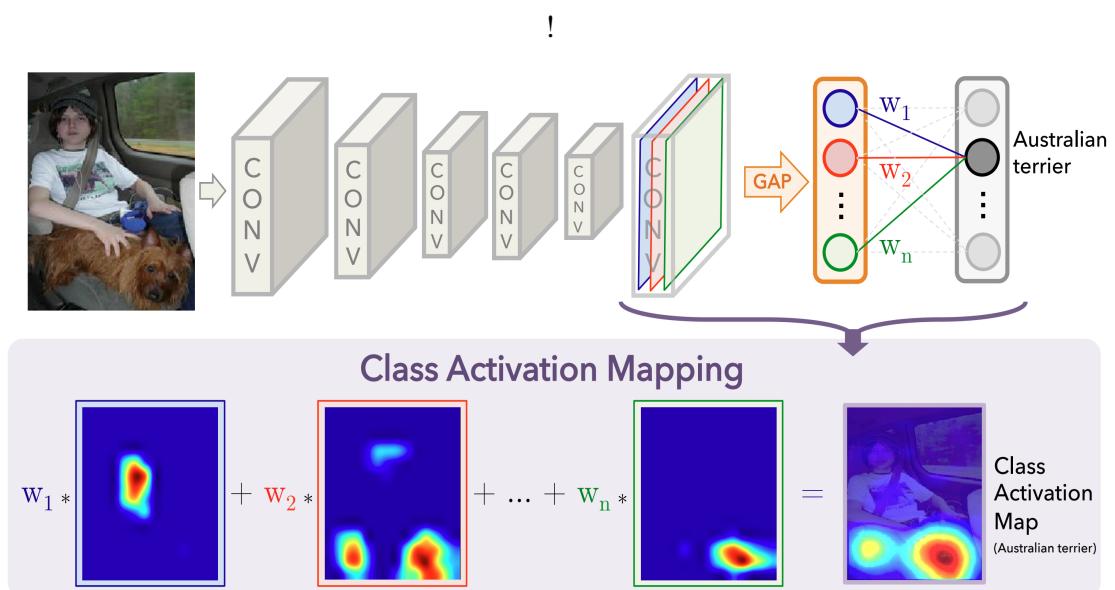


Fig. 2.8. Class Activation Maps [20]

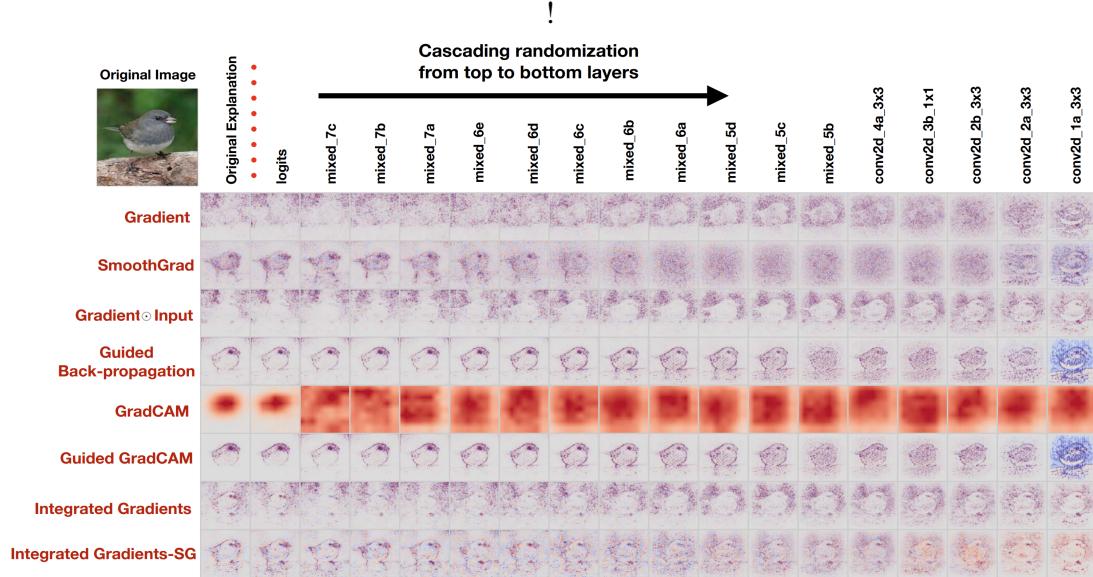


Fig. 2.9. Sanity Checks for Saliency Maps [21]

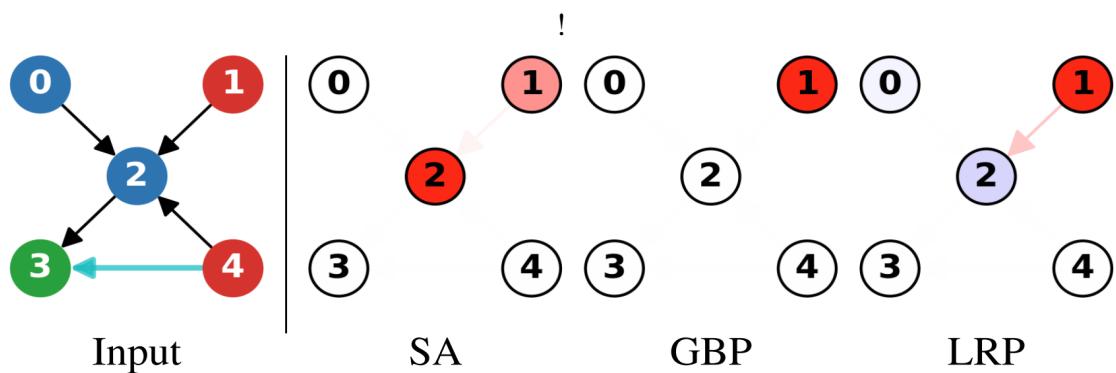


Fig. 2.10. Explicabilidad en GNNs [22]

3. UPINK INTERFERENCE OPTIMIZER

Como se mencionaba en la Introducción, una de las motivaciones del presente trabajo es proporcionar confianzas y una estimación de la incertidumbre en las aplicaciones que intentan modelar el comportamiento de las redes de telecomunicaciones. Por este motivo, en este capítulo se explicará en detalle el caso de uso de la aplicación industrial en la que se ha desarrollado el presente trabajo. En primer lugar, a continuación se hará una introducción al caso de uso en cuestión.

3.1. Introducción

El problema que se intenta resolver con el presente proyecto sería la optimización de una red celular de telecomunicaciones, que se muestra en la Figura 3.1. Este tipo de red es la que, por ejemplo, da servicio a los terminales móviles en el día a día. La red está dividida en sectores geográficos, denominados celdas, que pueden tener una o varias antenas y dan servicio a los terminales móviles que se encuentran en esa localización. Normalmente, dichas celdas tienen ciertos parámetros de configuración que se pueden modificar para alterar su comportamiento. Sin embargo, el problema de estos cambios es que algunos parámetros pueden ser muy sensibles y afectar al resto de la red, generando un impacto negativo en las celdas vecinas.

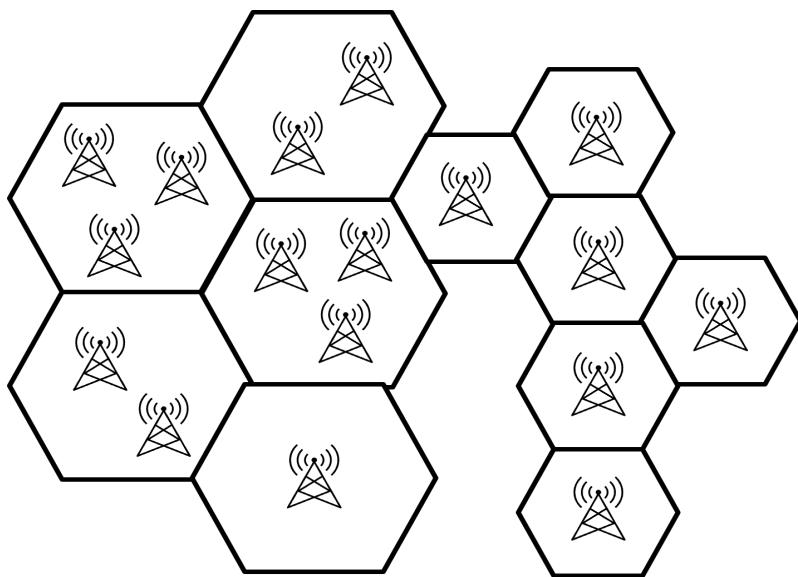


Fig. 3.1. Red Celular

Por tanto, el proyecto que se presenta aquí, se enmarca dentro de la optimización de una red celular. En concreto, el problema que se intenta resolver se llama *uplink interference*, que es la interferencia que surge en el enlace ascendente, es decir, cuando por ejemplo intentamos llamar por teléfono o subir un archivo a la nube. Para ello, se ha creado

un producto denominado Uplink Interference Optimizer, desarrollado para la tecnología 4G [26], que intenta proporcionar recomendaciones de los parámetros de configuración que se pueden modificar para resolverlo.

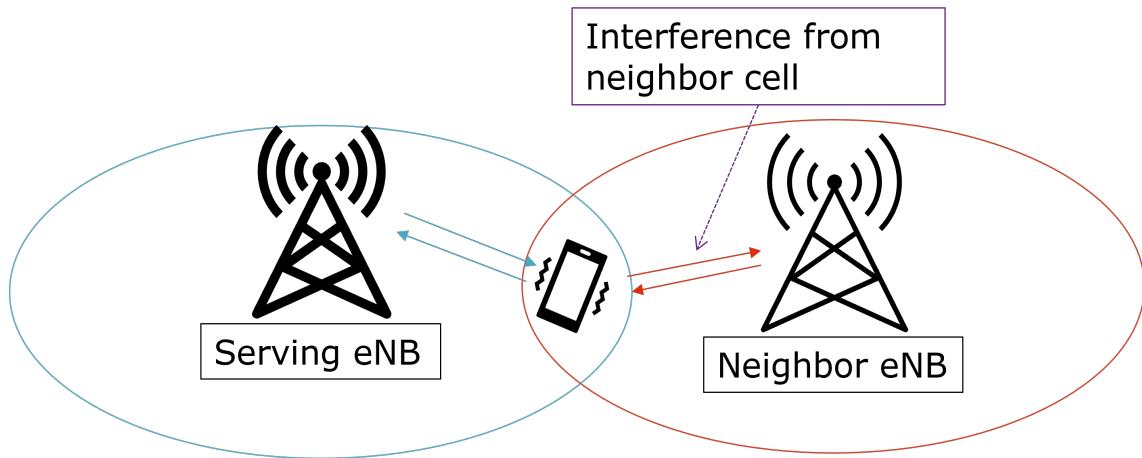


Fig. 3.2. *uplink interference*

3.1.1. Arquitectura

En esta sección se explicará la arquitectura general del Uplink Interference Optimizer. Aunque hay varios esquemas que se pueden plantear para optimizar este tipo de parámetros en las redes celulares, como por ejemplo Aprendizaje por Refuerzo, en este proyecto se plantea la posibilidad de crear un *digital twin* para realizar una optimización offline. Es decir, el objetivo es construir un modelo de la realidad para simular el comportamiento de la red celular, y que la optimización de la simulación se utilice para también optimizar la realidad. Este esquema se ilustra en la Figura 3.3.

El primer paso para desarrollar este esquema sería identificar una variable que refleje el desempeño de la red en el problema que se está intentando resolver. Es decir, esta variable, tendrá que estar correlada con dicho problema. A continuación, para intentar predecir dicha variable, se han de utilizar dos tipos de entradas, variables fijas y parámetros de configuración:

- Las variables fijas serán Key Performance Indicators (KPIs), variables construidas a partir de características del tráfico de la red, que reflejan su estado, que a su vez dependerá del número de usuarios.
- Los parámetros de configuración, que pueden modificar el comportamiento de la red. En este grupo estarán, por un lado, aquellos que se quieren optimizar en la red, y otros que trataremos como variables fijas.

Para el entrenamiento del modelo, se han usado datos recogidos en múltiples redes de todo el mundo (se explica con mayor detalle en la Sección 3.1.2), utilizando datos de

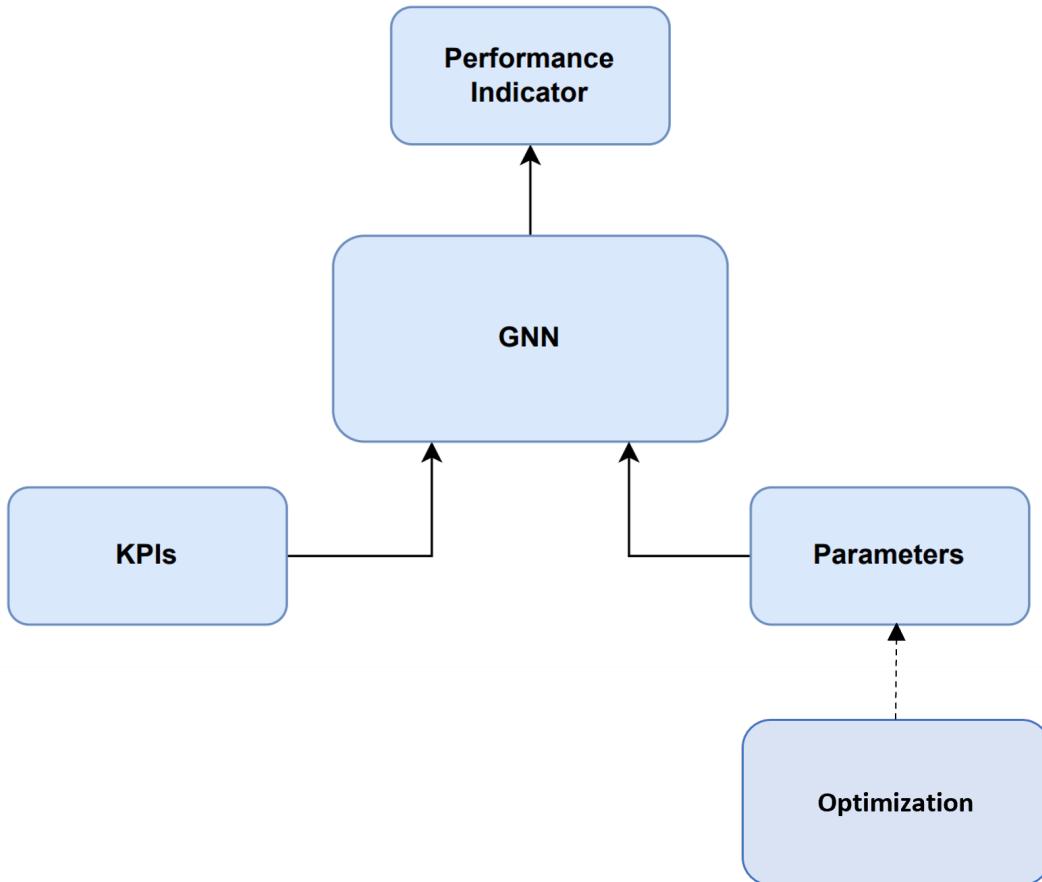


Fig. 3.3. Framework de Optimización

los KPIs y los parámetros sin modificación alguna, ya que será en una etapa posterior (optimización) cuando estos parámetros se modificarán. Por otra parte, el indicador que se intentará predecir en el presente caso de uso será la relación señal/ruido (SINR), que guarda una íntima relación con el problema del *uplink interference*. En concreto, cuanto mayor sea esta variable, menos interferencia habrá.

Posteriormente, una vez el modelo ha sido entrenado, podrá ser usado como un *digital twin* que simula la realidad. Lo que se hará en este caso es recoger datos durante una semana, y utilizar el modelo para encontrar los parámetros que maximizan el indicador que se está prediciendo, en este caso el SINR. Es importante mencionar que los datos se suelen recoger durante una semana de trabajo (lunes a viernes o de domingo a jueves, dependiendo del país en cuestión). Esto se hace de esta forma para evitar patrones de tráfico diferentes que se observan en los fines de semana. Los datos, como se comentará en la Sección 3.1.2, son agregados a nivel diario.

Una vez todos estos datos se recogen, se utiliza un algoritmo que busca parámetros que maximicen el SINR. Esto no se hace para todas las celdas de la red, sino aquellas que han sido identificadas como issue cells (celdas con un problema de *uplink interference*). La identificación de estas celdas se realiza con base en diferentes valores de KPI, aunque esto no se detallará en el presente trabajo, se puede encontrar más información en este

otro proyecto de Ericsson [27]. En concreto, el algoritmo que se utilizará se detalla en la Sección 3.1.2, y se basa en el gradiente.

En cuanto a los parámetros utilizados en el proyecto, son los siguientes:

- pZeroNominalPusch: este parámetro regula la potencia de transmisión de un terminal para el uplink. Un mayor valor en este parámetro conllevará un mayor ratio de transmisión, pero a la vez causará una mayor interferencia en las celdas vecinas.
- alpha: este parámetro es un factor de compensación del *pathloss*, i.e., la reducción de intensidad de potencia. Un mayor nivel de alpha implica una mayor tasa de trasmisión del terminal en los bordes de la celda, pero también incrementará la interferencia en las celdas vecinas.

3.1.2. Datos utilizados

En esta sección se detallarán los datos que han sido utilizados en este proyecto, tanto la naturaleza y cómo se procesan, hasta cuantas redes componen el *training set* y qué número de muestras contiene cada una. Es importante aclarar que, aunque esta sección muestra una descripción del tipo de datos y cómo se crea el grafo en cuestión, no se especificará en el presente documento qué variables concretas se han utilizado, ya que eso forma parte de la información sensible que podría permitir a un competidor de Ericsson replicar esta solución con una mayor facilidad.

En primer lugar, los datos se recogen a nivel horario, y después se agregan para tener un estimador de los KPIs para cada día. Por tanto, en los diferentes datasets encontraremos muestras que pertenecen a una misma celda pero en dos días diferentes. Estas dos muestras serán tratadas de manera independiente. De esta forma, cuando se hable de nodo en este documento, será para referirse a una celda en un día particular, siendo cada nodo un punto en el grafo.

Por otra parte, además de los diferentes KPIs y parámetros para cada celda de telecomunicaciones, se utilizarán variables que modelen los enlaces entre celdas, es decir, atributos de las aristas del grafo. En primer lugar, para definir dicho grafo, se hará uso de esas mismas variables. Es decir, no hay un grafo predefinido, sino que se crea a partir de estas variables. En concreto, se dividen en dos tipos:

- Variables geográficas. Para que una celda tenga una mayor influencia sobre otra, si se encuentra más próxima, esta será una de las variables a tener en cuenta.
- El segundo tipo de variables son las que reflejan la información que se intercambian las distintas celdas, e.g., el número de usuarios que viajan de una celda a otra.

Para la creación del grafo se establecen dos criterios:

- El primero de ellos es que para cada una de las variables que se utilizan, que forman parte de uno de los dos grupos descritos anteriormente, se fijan una serie de umbrales. Una celda solo podrá ser vecina de otra si se superan los mencionados umbrales.
- A continuación, solamente se establecerá una conexión entre las 15 celdas vecinas que menos distancia geográfica y mayor intercambio de información tengan. Esto se ha establecido así por una cuestión de cómputo, para intentar reducir la densidad del grafo. Además, acorde al criterio de expertos de radio en el equipo de Ericsson, como la mayor influencia la suelen tener las celdas más próximas, no sería necesario tener en cuenta más vecinos de los 15 establecidos.

Merece la pena resaltar, que se trata de un grafo dinámico en cuanto a la dimensión temporal se refiere. Es decir, el grafo se define en función de las variables geográficas, que no cambian, pero también en función de la información que se intercambia entre dos celdas, que puede cambiar día a día. Por tanto, aunque no sea lo común, dos celdas podrían ser vecinas un día y no al siguiente.

Una vez se ha establecido la estructura del grafo, las mismas variables que han sido utilizadas para definir las conexiones entre celdas, serán los atributos de las aristas, teniendo que utilizar una GNN que soporte este tipo de variables. Por tanto, el entrenamiento quedaría representado por la Figura 3.4, donde en azul se representan los KPIs, en verde los parámetros y en amarillo los atributos de las aristas.

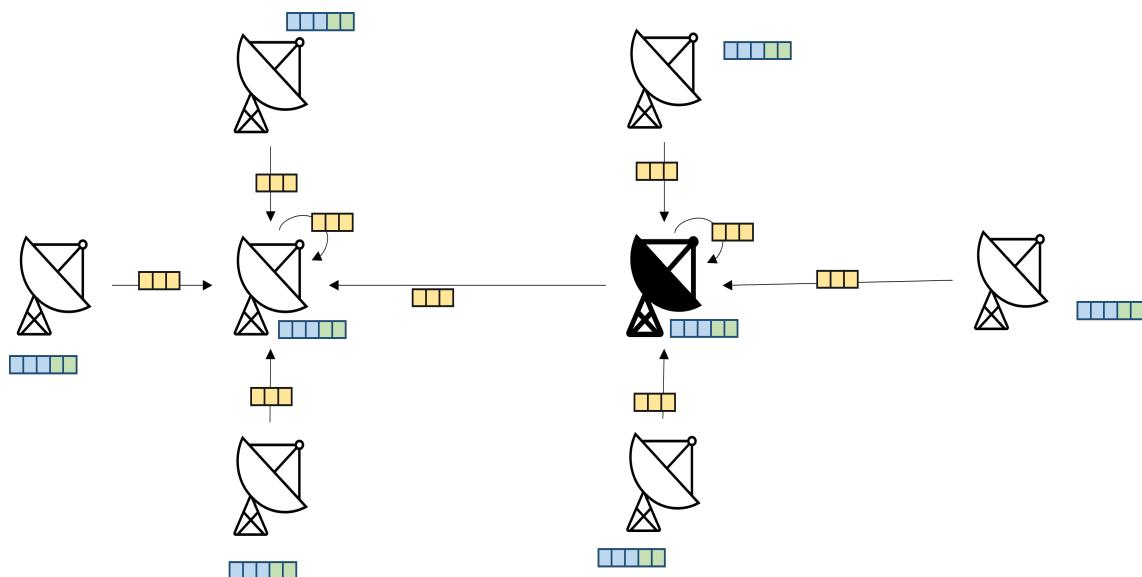


Fig. 3.4. Entrenamiento GNN

Finalmente, aunque los datos no se pueden compartir por acuerdos de privacidad con los diferentes clientes, sí es posible mostrar estadísticas para poder tener una idea del volumen de datos que se han utilizado para entrenar la solución. En concreto, en la Tabla 3.1 se muestra el número de nodos (una celda en un día en específico) para cada uno de los

operadores que se han usado, sin revelar el nombre de los operadores, pero indicando el continente en el que se encuentran.

Operador	Continente	Número de nodos
O1	Europa	60871
O2	Asia	155543
O3	Asia	40613
O4	Europa	32793
O5	América	304997
Global	-	594817

TABLA 3.1. DATOS 4G

3.1.3. Algoritmo de Optimización

Una vez se ha entrenado el modelo en cuestión, comienza la fase de optimización, que es el objetivo final de construir un modelo. La descripción del proceso a alto nivel puede observarse en la Figura 3.5, en la que se aprecia que el modelo solo actúa como un simulador de la realidad.

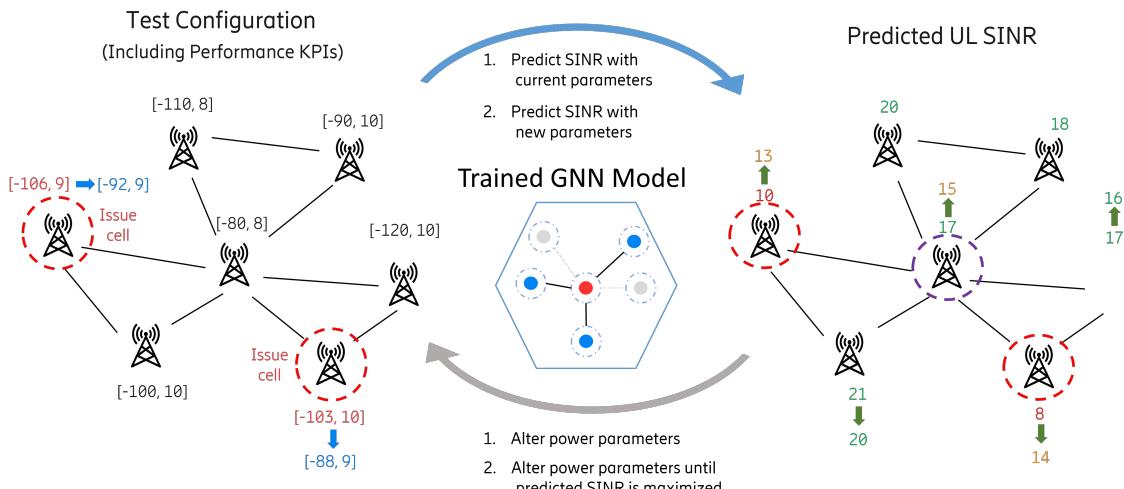


Fig. 3.5. Proceso de Optimización [28]

Por tanto, como muestra la Figura 3.5, el primer paso es identificar las *issue cells*, es decir, las celdas que son el objetivo de la optimización. Como se ha mencionado en la Sección 3.1.1, estas se detectan en función de los valores de los KPIs.

Como se explica en [29], el proceso de optimización se compone de los siguientes pasos:

- El primer paso sería expresar la función de coste a minimizar. En este caso, como el objetivo es maximizar el SINR, se utilizará el negativo de la variable. Además, la

función de pérdida deberá contener cualquier restricción que se quiera aplicar. En este proyecto, aunque se busque maximizar el SINR, se ha tenido en cuenta que si los valores de las configuraciones tienden a ser muy altos, el modelo generará más interferencia en las celdas vecinas. Esto se ha intentado modelar incluyéndolo en la función de coste, lo que nos proporciona la Ecuación 3.1.

$$\text{cost} = \|w_{\text{CELL}} \cdot \Delta\text{SINR}\|_F \quad (3.1)$$

siendo w_{CELL} el peso para cada celda y ΔSINR la diferencia entre el SINR predicho y el SINR objetivo. En concreto, el peso se ha fijado a 0.9 para las celdas que son issue cells, y 0.1 para los vecinos. El motivo de esto es para darle más importancia a las issue cells, sin dejar los vecinos completamente de lado. Por otra parte, el SINR objetivo será siempre 20 dB para las issue cells, y el valor original de SINR para el resto de celdas, de forma que los vecinos tienden a intentar mantener su nivel de SINR.

Sin embargo, como experimentalmente se ha visto en pruebas con clientes (cuyos datos no se pueden revelar en el presente trabajo) que el modelo no es lo suficientemente preciso en las celdas vecinas, se ha propuesto la alternativa de incluir en la función de coste el valor de sus configuraciones como si fuera un factor de regularización.

$$\text{cost} = w_P \cdot \|w_{\text{CELL}} \cdot P\|_F + w_\alpha \cdot \|w_{\text{CELL}} \cdot \alpha\|_F + w_{\text{SINR}} \cdot \|w_{\text{CELL}} \cdot \Delta\text{SINR}\|_F \quad (3.2)$$

- Una vez la función de coste se ha definido, comienza la fase de optimización. En este punto el modelo se utilizará como un simulador y con un algoritmo de optimización estocástica, como Adam [30], se modificarán los parámetros que minimizan la función de coste descrita anteriormente. El principal problema de esta fase es que los parámetros a optimizar son discretos. Por este motivo, los pasos que se seguirán en la optimización serán los siguientes:
 - Optimizar los parámetros como si fueran variables continuas durante un número fijo de iteraciones.
 - Ejecutar clipping de los parámetros para mantenerlos en un rango de valores que tiene sentido para el caso de uso (los valores posibles para el parámetro en concreto).
 - Una vez la optimización ha finalizado, aproximamos los valores continuos a valores discretos. Para ello, tomamos la combinación de los dos parámetros como si fueran vectores de dos dimensiones, aproximando los valores a la combinación que minimiza la distancia entre vectores.

3.2. New Radio - 5G

El proyecto del Uplink Interference Optimizer, como se ha explicado en la Sección 3.1, se ha desarrollado para la tecnología 4G. Sin embargo, las redes se van adaptando a New Radio (NR) o 5G, lo que hace que en el futuro sea necesario este mismo proyecto para 5G, especialmente según el uso de este tipo de redes aumente, lo que hará a su vez que el problema del *uplink interference* en el 5G se haga más frecuente. Por este motivo, una de las líneas principales de este trabajo, además de añadir Bayesian GNNs al proyecto, ha sido replicar el caso de uso para 5G. En esta sección se introducirán las diferencias entre las dos tecnologías y una pequeña introducción a las diferentes redes que se han utilizado para entrenar el modelo de 5G.

3.2.1. Diferencias 4G vs 5G

La mayor y principal diferencia entre las dos tecnologías es que en 5G solamente tendremos disponible uno de los dos parámetros que podíamos configurar en el 4G, el pZeroNominalPusch. En el caso del 5G, el nombre del parámetro será pZeroNomPusch-Grant.

Además, los KPIs que se utilizarán serán similares en su mayoría, aunque habrá alguna diferencia entre la lista de entradas. Como ya se ha mencionado en apartados anteriores, no se describirá la lista específica de KPIs y parámetros por temas de confidencialidad.

3.2.2. Datos 5G

En el caso del 5G, los datos tendrán la misma estructura que los del 4G, tal y como se explica en 3.1.2. En este caso también se utilizan datos de diferentes localizaciones como se puede observar en la Tabla 3.2.

Operador	Continente	Número de nodos
O1	América	1830
O2	Europa	92379
O3	Europa	3361
O4	Asia	34839
O5	Europa	67472
O6	América	1006
Global	-	200887

TABLA 3.2. DATOS 5G

3.3. Bayesian Graph Neural Network

En esta sección se explicará el modelo de Bayesian Graph Neural Network y cómo se ha integrado en la solución del Uplink Interference Optimizer.

3.3.1. Datos para el entrenamiento

Como se ha indicado antes en la Sección 3.1.2, los datos que se utilizarán para entrenar la versión del 4G están compuestos por 5 datasets diferentes procedentes de operadores en Europa, América, África y Asia. Por otra parte, en la versión del 5G, explicada en la Sección 3.2, se utilizarán también datasets provenientes de operadores en Europa, América y Asia.

En concreto, una de las particularidades más importantes que mencionar es cómo se han particionado los datos en conjuntos de entrenamiento, validación y test. Una partición típica sería simplemente dividir el número de nodos que hay entre las proporciones dedicadas a cada conjunto. Sin embargo, teniendo en cuenta que cada celda está representada por varios nodos (uno por cada día) y que el valor del SINR en estos nodos es relativamente similar, en el proyecto se ha utilizado otro modo para dividir los datos. Específicamente, se dividirán las celdas en vez de los nodos, estando todos los nodos que pertenezcan a una sola celda en un mismo conjunto.

3.3.2. Balanceo en el entrenamiento

En esta sección se incluye cómo se han balanceado los datos en el entrenamiento para intentar maximizar el rendimiento del modelo. En primer lugar, hay que comprobar la distribución de los datos en función de los parámetros que se intentarán maximizar, que se pueden observar en las Figuras 3.6, 3.7 y 3.8. Como en estas Figuras no se observa con claridad el número de muestras de las clases menos habituales, también se muestran las Tablas 3.3, 3.4 y 3.5. en las que se puede comprobar dicha información en detalle.

Configuración	Número de muestras
8	79
9	36018
10	558720

TABLA 3.3. DISTRIBUCIÓN ALPHA 4G

Como se puede observar en las figuras y las tablas mencionadas anteriormente, hay una gran diferencia en el número de muestras entre unas configuraciones y otras. Como el objetivo es un modelo que sea fiable en todo tipo de situaciones para poder obtener recomendaciones en el momento de optimizar las configuraciones, a la hora de entrenar los modelos se utilizará un balanceo entre las distintas configuraciones con pesos para

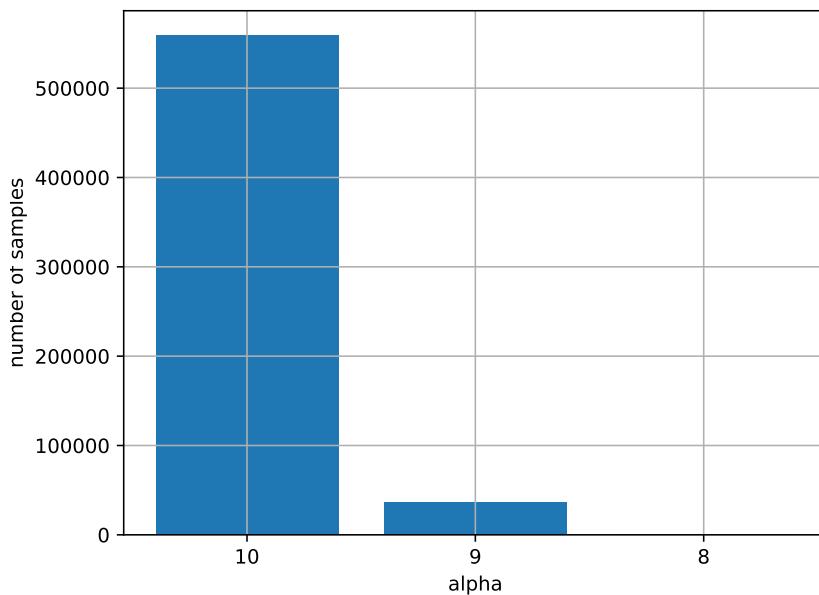


Fig. 3.6. Distribución alpha 4G

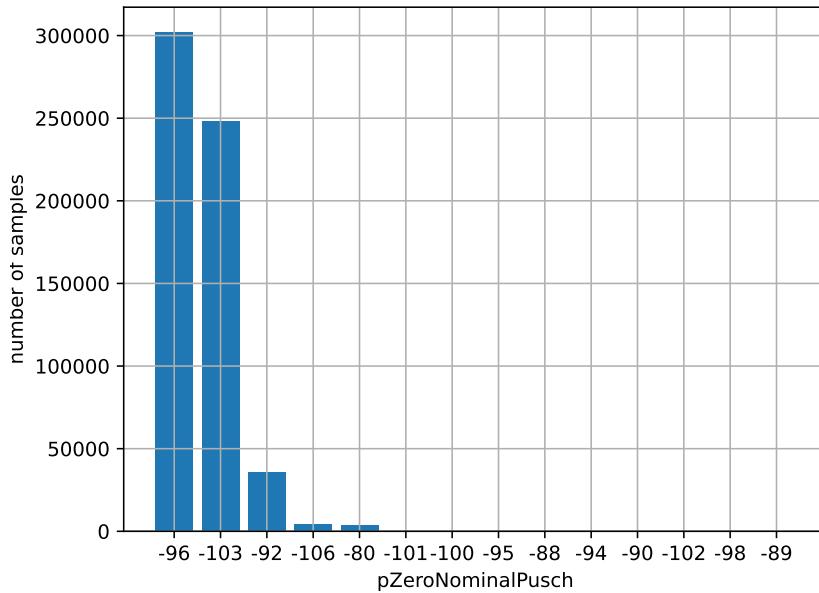


Fig. 3.7. Distribución pZeroNominalPusch 4G

ponderar el cálculo de la función de perdida. Específicamente se realizarán tres balancesos diferentes:

- En las configuraciones de alpha (solo aplica a 4G). Se calcularán unos pesos para que cada configuración tenga un peso equivalente en todo el conjunto de datos.
- En las configuraciones de pZeroNominalPusch para el 4G y pZeroNominalPuschGrant

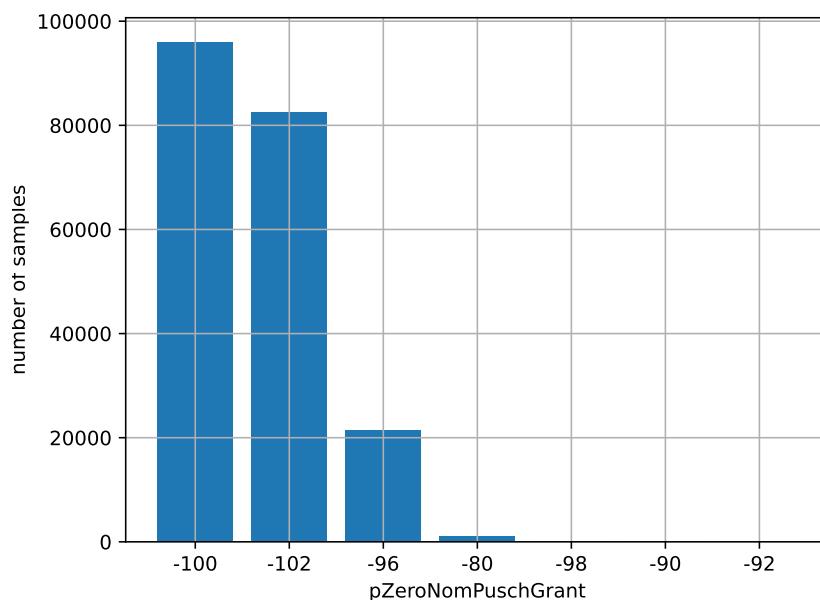


Fig. 3.8. Distribución pZeroNomPuschGrant 5G

Configuración	Número de muestras
-106	4567
-103	247946
-102	27
-101	138
-100	132
-98	11
-96	302070
-95	116
-94	94
-92	35697
-90	38
-89	10
-88	101
-80	3870

TABLA 3.4. DISTRIBUCIÓN PZERONOMINALPUSCH 4G

para el 5G. Al igual que en el caso anterior, se calculan unos pesos para que cada configuración tenga el mismo peso.

- En los distintos conjuntos de datos para cada operador. Como hay unos operadores con muchas más muestras que otros, también se calcularán pesos para ponderar este factor en la función de pérdida.

Configuración	Número de muestras
-102	4567
-100	247946
-98	27
-96	138
-92	132
-90	11
-80	302070

TABLA 3.5. DISTRIBUCIÓN PZERONOMPUSCHGRANT 5G

Los pesos calculados por los factores anteriores se multiplicarán entre ellos para calcular el peso final de cada muestra. Es importante mencionar que estos pesos no solo se aplicarán a la función de pérdida, sino también al MAE, MSE y sMAPE que se mostrarán para los diferentes conjuntos.

3.3.3. Normalización

En cuanto a la normalización para los datos, se han aplicado dos tipos diferentes:

- Una primera normalización con límites establecidos por los expertos en radio del equipo de Ericsson. Como ya se ha mencionado anteriormente en situaciones similares, estos valores no serán revelados por criterios de confidencialidad.
- Una segunda normalización en función del máximo y el mínimo que se pueden encontrar en el conjunto de entrenamiento, lo que se puede observar en la Ecuación 3.3.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3.3)$$

Es importante resaltar la importancia de utilizar el mínimo y el máximo que se encuentran en el conjunto de entrenamiento y no en la totalidad de los datos, para replicar unas condiciones reales en evaluación, donde se normalizará con valores que ya hemos definido al no poder depender del conjunto de datos externo.

3.3.4. Creación de Batches

Una de las dificultades al entrenar esta clase de modelos con un conjunto de datos tan grande como el que se describe en la Sección 3.1.2 es como dividirlo para poder crear *batches*. Esto se realiza tanto para entrenar como para realizar las predicciones, al contrario que si el grafo fuera más pequeño, ya que se podría cargar el conjunto de datos entero en memoria y realizar así las operaciones. En concreto, se asume que los recursos

son suficientes para cargar todos los datos en memoria en un primer momento, ya que de otra forma no se podría realizar la partición del grafo, pero los recursos para ejecutar las predicciones exceden a los recursos disponibles. Para esta operación hay varias formas, pero la que se ha utilizado en este proyecto en base a los resultados expuestos en la literatura es el método presentado en [31], que consiste en particionar el grafo y utilizar cada subgrafo como si fuera un *batch*. El mecanismo Message Passing de la GNN bajo este tipo de división se puede observar en la Figura 3.9.

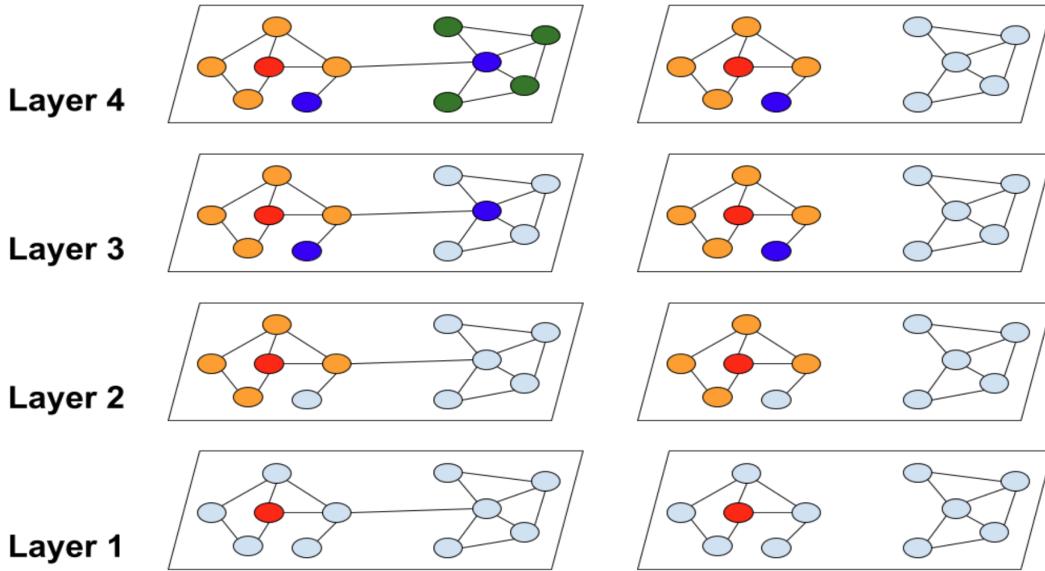


Fig. 3.9. Predicción normal a la izquierda vs predicción con clusters [31] a la derecha

Es importante mencionar que este mecanismo se usa tanto para el entrenamiento como para la evaluación, pero no se utiliza cuando el modelo se despliega para optimizar una serie de celdas. El motivo de esto es que los grafos que se usan en producción suelen ser mucho más pequeños, correspondiendo solo a un pequeño *cluster* que el cliente quiere optimizar en su red, en vez de a la red entera. Por tanto, no sería necesario realizar la partición. Además, tengamos en cuenta que dicha división cambia la topología del grafo en los bordes de los *clusters*, haciendo que las predicciones en dichas celdas puedan ser muy diferentes.

3.3.5. Modelos

En esta sección se explicarán los modelos que se han utilizado en el presente trabajo y una descripción de la arquitectura de los mismos.

En concreto, se ha utilizado una arquitectura compuesta por GNNs y redes lineales y no lineales. Aunque el número de capas no se detallará de forma exacta, de nuevo para evitar que la solución sea fácilmente replicada por competidores, se ha utilizado la

combinación de capas de GNNs y después un MLP para intentar modelar los siguientes patrones:

- En primer lugar, con las GNNs se intenta capturar la influencia de las celdas vecinas, que tendrán un impacto significativo en el problema del *uplink interference*, y, por tanto, en la predicción del SINR.
- Como el comportamiento de las redes celulares puede llegar a ser altamente no lineal, de acuerdo a los expertos de Ericsson, un bloque MLP puede ayudar a aprender patrones más complejos una vez la información de los vecinos ha sido agregada.

En cuanto a las GNNs utilizadas, se han probado las siguientes:

- Crystal Graph Convolution [32]: Este modelo se ha propuesto dado que era el que se usaba en una versión anterior del proyecto, elegido porque en un estudio experimental daba los mejores resultados.
- Graph Attentional Network (GAT) v2 [33]: Dado que es uno de los modelos de GNNs más conocidos y utilizados, se ha considerado que puede ser un buen punto de referencia con el que comparar.

3.3.6. Resultados

Métrica	Global	O1	O2	O3	O4	O5
NMAE	0.005	0.012	0.005	0.004	0.008	0.012
NMSE	0.000	0.000	0.000	0.000	0.000	0.001
sMAPE	0.008	0.022	0.009	0.008	0.015	0.019
MAE	0.122	0.318	0.140	0.110	0.211	0.320
MSE	0.052	0.247	0.061	0.040	0.127	0.367
pi-ci 1 std	0.801	0.562	0.775	0.857	0.615	0.604
pi-ci 2 std	0.939	0.811	0.941	0.949	0.851	0.823
pi-ci 3 std	0.977	0.921	0.983	0.978	0.942	0.911
width-ci 1 std	0.176	0.254	0.189	0.219	0.195	0.241
width-ci 2 std	0.351	0.508	0.377	0.437	0.391	0.483
width-ci 3 std	0.527	0.763	0.566	0.656	0.586	0.724

TABLA 3.6. MÉTRICAS ENTRENAMIENTO CRYSTAL GNN 4G

En esta sección se presentan los resultados de los diferentes modelos utilizados y un breve análisis de los mismos. En primer lugar, se mostrarán las tablas de resultados para los modelos que se han explicado en la Sección 3.3.5. Antes de analizar los resultados es importante conocer las métricas utilizadas en la evaluación:

Métrica	Global	O1	O2	O3	O4	O5
NMAE	0.015	0.021	0.013	0.027	0.010	0.013
NMSE	0.001	0.001	0.000	0.002	0.000	0.000
sMAPE	0.028	0.039	0.022	0.060	0.018	0.022
MAE	0.406	0.570	0.349	0.719	0.260	0.342
MSE	0.388	0.667	0.348	1.167	0.167	0.336
pi-ci 1 std	0.423	0.361	0.486	0.249	0.504	0.497
pi-ci 2 std	0.721	0.666	0.780	0.575	0.808	0.763
pi-ci 3 std	0.850	0.816	0.881	0.772	0.926	0.896
width-ci 1 std	0.240	0.298	0.234	0.285	0.200	0.230
width-ci 2 std	0.480	0.596	0.468	0.570	0.401	0.461
width-ci 3 std	0.721	0.894	0.702	0.855	0.601	0.691

TABLA 3.7. MÉTRICAS VALIDACIÓN CRYSTAL GNN 4G

Métrica	Global	O1	O2	O3	O4	O5
NMAE	0.005	0.013	0.006	0.006	0.010	0.012
NMSE	0.000	0.000	0.000	0.000	0.000	0.000
sMAPE	0.009	0.024	0.010	0.011	0.018	0.019
MAE	0.132	0.349	0.166	0.150	0.265	0.312
MSE	0.069	0.298	0.086	0.065	0.181	0.247
pi-ci 1 std	0.811	0.565	0.755	0.768	0.527	0.559
pi-ci 2 std	0.918	0.763	0.917	0.935	0.785	0.802
pi-ci 3 std	0.961	0.883	0.966	0.973	0.905	0.909
width-ci 1 std	0.177	0.261	0.197	0.235	0.204	0.251
width-ci 2 std	0.354	0.521	0.393	0.470	0.407	0.503
width-ci 3 std	0.531	0.782	0.590	0.706	0.611	0.754

TABLA 3.8. MÉTRICAS ENTRENAMIENTO GAT GNN 4G

- NMAE: MAE normalizado. Los valores que se han utilizado en la normalización son los mismos que se han mencionado en la Sección 3.3.3.
- NMSE: MSE normalizado. Los valores utilizados son los mismos que los de la anterior métrica.
- sMAPE: Como hay múltiples implementaciones de esta métrica, en este trabajo se usará la expresada en la Ecuación 3.4, cuya implementación se puede encontrar en [34].

$$\text{sMAPE} = \frac{2}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{\max(|y_i| + |\hat{y}_i|, \epsilon)} \quad (3.4)$$

- pi-ci 1 std: Porcentaje de muestras en las que el valor real se encuentra dentro

Métrica	Global	O1	O2	O3	O4	O5
NMAE	0.019	0.021	0.019	0.023	0.013	0.014
NMSE	0.001	0.001	0.001	0.001	0.000	0.001
sMAPE	0.035	0.039	0.032	0.049	0.023	0.025
MAE	0.493	0.568	0.495	0.614	0.335	0.383
MSE	0.517	0.623	0.538	0.826	0.285	0.398
pi-ci 1 std	0.339	0.356	0.368	0.316	0.417	0.442
pi-ci 2 std	0.613	0.651	0.603	0.564	0.720	0.719
pi-ci 3 std	0.774	0.806	0.750	0.734	0.864	0.864
width-ci 1 std	0.242	0.315	0.233	0.279	0.208	0.232
width-ci 2 std	0.484	0.629	0.467	0.557	0.416	0.465
width-ci 3 std	0.725	0.944	0.700	0.836	0.625	0.697

TABLA 3.9. MÉTRICAS VALIDACIÓN GAT GNN 4G

Métrica	Global	O1	O2	O3	O4	O5	O6
NMAE	0.01	0	0.02	0.01	0	0.02	0
NMSE	0	0	0	0	0	0	0
sMAPE	0.01	0.004	0.02	0.01	0.01	0.02	0.004
MAE	0.16	0.1	0.42	0.15	0.12	0.45	0.1
MSE	0.09	0.02	0.38	0.07	0.04	0.42	0.02
pi-ci 1 std	0.72	0.64	0.42	0.69	0.81	0.35	0.68
pi-ci 2 std	0.87	0.9	0.64	0.87	0.93	0.57	0.91
pi-ci 3 std	0.93	0.94	0.78	0.94	0.98	0.73	0.95
width-ci 1 std	0.16	0.1	0.23	0.15	0.2	0.22	0.11
width-ci 2 std	0.32	0.2	0.45	0.3	0.39	0.43	0.23
width-ci 3 std	0.48	0.3	0.68	0.45	0.59	0.65	0.34

TABLA 3.10. MÉTRICAS ENTRENAMIENTO CRYSTAL GNN 5G

de los intervalos de confianza generados con 1 desviación típica, es decir, dichos intervalos se han generado sumando y restando una desviación típica a la media de las predicciones.

- pi-ci 2 std: Porcentaje de muestras en las que el valor real se encuentra dentro de los intervalos de confianza generados con 2 desviaciones típicas.
- pi-ci 3 std: Porcentaje de muestras en las que el valor real se encuentra dentro de los intervalos de confianza generados con 3 desviaciones típicas.
- width-ci 1 std: Tamaño medio de los intervalos de confianza generados con 1 desviación típica.
- width-ci 2 std: Tamaño medio de los intervalos de confianza generados con 2 des-

Métrica	Global	O1	O2	O3	O4	O5	O6
NMAE	0.02	0	0.02	0.01	0.03	0.02	0
NMSE	0	0	0	0	0	0	0
sMAPE	0.02	0	0.03	0.02	0.03	0.03	0
MAE	0.4	0.11	0.6	0.35	0.64	0.57	0.12
MSE	0.46	0.02	0.87	0.33	0.82	0.79	0.03
pi-ci 1 std	0.42	0.57	0.29	0.46	0.29	0.28	0.57
pi-ci 2 std	0.61	0.88	0.51	0.65	0.48	0.51	0.85
pi-ci 3 std	0.77	0.96	0.68	0.79	0.66	0.67	0.97
width-ci 1 std	0.17	0.1	0.23	0.16	0.23	0.22	0.11
width-ci 2 std	0.34	0.2	0.46	0.31	0.45	0.43	0.21
width-ci 3 std	0.5	0.29	0.68	0.47	0.68	0.65	0.32

TABLA 3.11. MÉTRICAS VALIDACIÓN CRYSTAL GNN 5G

Métrica	Global	O1	O2	O3	O4	O5	O6
NMAE	0	0	0.01	0	0	0.01	0
NMSE	0	0	0	0	0	0	0
sMAPE	0.01	0	0.01	0.01	0	0.02	0
MAE	0.1	0.11	0.3	0.1	0.06	0.34	0.1
MSE	0.04	0.02	0.21	0.03	0.01	0.25	0.02
pi-ci 1 std	0.81	0.53	0.56	0.8	0.92	0.47	0.62
pi-ci 2 std	0.93	0.85	0.78	0.93	0.97	0.7	0.88
pi-ci 3 std	0.97	0.94	0.89	0.97	0.99	0.83	0.95
width-ci 1 std	0.15	0.1	0.26	0.15	0.16	0.25	0.11
width-ci 2 std	0.31	0.19	0.52	0.33	0.32	0.55	0.33
width-ci 3 std	0.46	0.29	0.78	0.45	0.48	0.75	0.33

TABLA 3.12. MÉTRICAS ENTRENAMIENTO GAT GNN 5G

viaciones típicas.

- width-ci 3 std: Tamaño medio de los intervalos de confianza generados con 3 desviaciones típicas.

En primer lugar, para cada una de las tecnologías se han extraído los resultados para los diferentes tipos de modelos en los conjuntos de entrenamiento y validación. Después, solo para el modelo que obtuviera el mejor resultado en validación, se calcula y muestra el resultado en el conjunto test. Además, el mejor modelo será el utilizado en la optimización en la Sección 3.4. La métrica que se utilizará para medir esto será el MAE.

En el caso del 4G, los resultados los podemos encontrar en las Tablas 3.6, 3.8, 3.7 y 3.9. Como se puede observar en ellas, los resultados de los diferentes modelos son similares, obteniendo ambos un error cercano al 1 %. Además, los intervalos de confianza

Métrica	Global	O1	O2	O3	O4	O5	O6
NMAE	0.02	0	0.03	0.01	0.03	0.02	0.01
NMSE	0	0	0	0	0	0	0
sMAPE	0.02	0.01	0.03	0.02	0.04	0.03	0.01
MAE	0.43	0.12	0.66	0.34	0.71	0.61	0.14
MSE	0.52	0.03	1.02	0.33	0.96	0.89	0.04
pi-ci 1 std	0.38	0.39	0.3	0.47	0.19	0.29	0.45
pi-ci 2 std	0.61	0.82	0.55	0.67	0.42	0.54	0.75
pi-ci 3 std	0.78	0.94	0.72	0.82	0.56	0.71	0.93
width-ci 1 std	0.18	0.09	0.28	0.17	0.22	0.25	0.1
width-ci 2 std	0.36	0.19	0.55	0.33	0.45	0.51	0.21
width-ci 3 std	0.54	0.28	0.83	0.5	0.67	0.76	0.31

TABLA 3.13. MÉTRICAS VALIDACIÓN GAT GNN 5G

Métrica	Global	O1	O2	O3	O4	O5
NMAE	0.021	0.020	0.020	0.029	0.009	0.012
NMSE	0.001	0.001	0.001	0.003	0.000	0.000
sMAPE	0.062	0.037	0.063	0.089	0.015	0.020
MAE	0.566	0.531	0.519	0.776	0.234	0.327
MSE	0.935	0.603	0.791	1.858	0.149	0.340
pi-ci 1 std	0.469	0.404	0.471	0.396	0.558	0.553
pi-ci 2 std	0.698	0.686	0.711	0.596	0.826	0.797
pi-ci 3 std	0.789	0.835	0.815	0.746	0.922	0.902
width-ci 1 std	0.255	0.302	0.245	0.290	0.189	0.232
width-ci 2 std	0.510	0.603	0.490	0.580	0.378	0.463
width-ci 3 std	0.766	0.905	0.735	0.870	0.567	0.695

TABLA 3.14. MÉTRICAS TEST SET CRYSTAL GNN 4G

generados son lo suficientemente amplios como para poder utilizarlos para poder dar estimaciones de incertidumbre en producción junto a las predicciones. En concreto, como no hay forma de asegurar que la salida sea una distribución concreta, en producción se suele dar como valor de confianza de un intervalo el porcentaje de muestras en las que el valor real se encuentra dentro del intervalo en el conjunto de validación. En este caso, como el modelo de Crystal Graph Convolution parece ser algo mejor, este será el modelo que se utilizará en la optimización. Sus resultados en el conjunto de test se muestran en la Tabla 3.14.

Por otro lado, para el 5G, los resultados los encontramos en las Tablas 3.10, 3.12, 3.11 y 3.13. De nuevo se puede observar un error bastante bajo, cercano al 1 % en su versión normalizada, y unos intervalos de confianza que parecen ser lo suficientemente amplios como para poder proporcionar estimaciones de estos a los clientes. Además, el modelo

Métrica	Global	O1	O2	O3	O4	O5	O6
NMAE	0.016	0.005	0.029	0.011	0.025	0.024	0.006
NMSE	0.001	0.000	0.002	0.000	0.001	0.001	0.000
sMAPE	0.019	0.006	0.036	0.014	0.031	0.031	0.006
MAE	0.386	0.136	0.715	0.278	0.619	0.592	0.141
MSE	0.500	0.037	1.245	0.249	0.797	0.792	0.042
pi-ci 1 std	0.419	0.511	0.251	0.501	0.253	0.261	0.482
pi-ci 2 std	0.673	0.804	0.466	0.744	0.502	0.492	0.825
pi-ci 3 std	0.794	0.886	0.640	0.845	0.619	0.657	0.956
width-ci 1 std	0.166	0.093	0.243	0.150	0.211	0.218	0.121
width-ci 2 std	0.332	0.186	0.486	0.300	0.422	0.436	0.243
width-ci 3 std	0.499	0.279	0.729	0.450	0.633	0.653	0.364

TABLA 3.15. MÉTRICAS TEST SET CRYSTAL GNN 5G

de Crystal Graph Convolution vuelve a ser el que mejor resultados obtiene, quedando resumidos sus resultados en el conjunto de test en la Tabla 3.15.

3.4. Optimización Probabilística

En esta sección se explicarán las modificaciones del algoritmo de optimización que se presentó en la Sección 3.1.3. El principal dilema que se plantea es el funcionamiento del modelo durante la optimización de las configuraciones:

- Por un lado, se pueden congelar todas las capas del modelo, generando una muestra de pesos y fijando dichos pesos como los que el modelo utilizará durante la optimización.
- Por otro lado, se podría realizar una optimización sin congelar las capas, ejecutando varias veces el modelo y usando la media de los gradientes de cada ejecución como valor final para realizar la optimización en cada iteración.

Para poder comprobar las ventajas de cada uno se ha ejecutado la optimización con las dos opciones para poder valorar las diferencias, además de una ejecución extra en otra red para la opción que proporcionara mejores resultados.

En el caso del 4G, se pueden observar las diferencias en las Tablas 3.16, 3.17, 3.18, 3.19, 3.20 y 3.21. El operador que se ha utilizado para estas pruebas es el O5. Con base en estos resultados se puede concluir que la opción de no congelar las capas ofrece unos mejores resultados en la ganancia para las *issue cells*, aunque la diferencia es pequeña. Sin embargo, esta pequeña ganancia se obtiene a cambio de un mayor gasto computacional, donde se pasa de 10 segundos a 7 minutos y 18 segundos. Por tanto, es razonable concluir

que la mejor opción para un despliegue en producción sería la primera, teniendo en cuenta que el número de celdas sería mucho mayor que en esta pequeña prueba. En cuanto a los vecinos, también se aprecia que las recomendaciones no les afectan, habiendo conseguido mejorar las *issue cells* sin falta de causar más interferencia en las celdas vecinas. Además, se puede comprobar que las recomendaciones están dispersas entre las diferentes configuraciones, lo que escenifica que el modelo ha aprendido a optimizar en diferentes escenarios y no solamente a subir la configuración a la máxima potencia. Por último, es importante resaltar el factor de ganancia, es decir, que al ser la ganancia predicha mucho mayor que el error del modelo, la predicción de una cierta ganancia es fiable.

Como se ha mencionado anteriormente, para el modelo que ofrece los mejores resultados se ha realizado otra prueba, en esta ocasión simulando un despliegue en una red real. Se ha elegido la opción de congelar el modelo debido a lo expuesto anteriormente. Esta será una red extra a la que llamaremos operador O6. Sus resultados se muestran en las Tablas 3.22, 3.23 y 3.24. Observamos unos resultados similares, con una ganancia menor, pero claramente por encima del error del modelo, además de un impacto nulo en los vecinos y unas recomendaciones distribuidas entre las diferentes configuraciones.

Por otra parte, para el 5G los resultados de la comparación quedan reflejados en las Tablas 3.25, 3.26, 3.27, 3.28, 3.29 y 3.30. El operador que se ha utilizado para estas pruebas es el O1. Se observa en ellas unos resultados similares a los de 4G, con un mejor resultado de la optimización sin congelar, aumentando el gasto computacional de 1 segundo a 6 (en este caso la red utilizada es significativamente más pequeña). La única diferencia reseñable sería el factor de ganancia, ya que en este caso la ganancia es tan pequeña que es menor que el error esperado del modelo. Es por eso que este caso se concluye que no hay una ganancia fiable, ya que al estar dentro del margen de error podría ser que no hubiera ganancia. Esto sucede porque en el conjunto de datos utilizado hay aún poca congestión de tráfico, ya que las redes 5G aún no tienen muchos usuarios, y, por tanto, apenas sufren el problema de *uplink interference*.

De forma análoga al caso del 4G, el modelo que ofrece mejores resultados (también en este caso el modelo congelado debido a la poca diferencia y al aumento del gasto computacional) se ejecuta en un caso real extra. En este caso, el nuevo operador O7 ha sido escogido por tener un mayor volumen de tráfico en su red 5G, por lo que podemos esperar ganancias más altas. Esto se puede comprobar en las Tablas 3.31, 3.32 y 3.33. Queda, por tanto, comprobado el funcionamiento de la solución en el caso del 5G también, donde en una red con suficiente tráfico se observa la ganancia, que se sitúa de nuevo claramente por encima del error del modelo.

3.5. Explicabilidad

En esta sección se explica el resultado de la explicabilidad del modelo. En concreto, el objetivo de la explicabilidad que se expone en el presente trabajo se trata de identificar

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	13.571	11.560	13.876	15.678
Predicted Original SINR	14.301	12.029	15.122	16.616
Error SINR	0.926	0.485	0.847	1.163
Predicted Final SINR	18.255	17.887	18.445	18.981
Real Gain SINR	4.684	3.177	4.536	5.927
Predicted Gain SINR	3.954	2.501	3.852	5.535
Factor: Predicted Gain / Error	13.358	2.344	4.122	10.110

TABLA 3.16. OPTIMIZACIÓN MODELO CONGELADO *ISSUE CELLS 4G*

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	16.281	14.908	17.065	18.850
Predicted Original SINR	16.331	14.910	17.217	18.768
Error SINR	0.533	0.153	0.381	0.722
Predicted Final SINR	16.303	14.819	17.214	18.804
Real Gain SINR	0.022	-0.415	-0.071	0.337
Predicted Gain SINR	-0.029	-0.136	-0.093	-0.056
Factor: Predicted Gain / Error	-3.516	-0.674	-0.272	-0.112

TABLA 3.17. OPTIMIZACIÓN MODELO CONGELADO *FIRST-HOP NEIGHBOR CELLS 4G*

Configuración	Número de Celdas
(10, -103)	4
(10, -80)	60
(10, -82)	8
(10, -85)	4
(10, -86)	4
(10, -89)	4
(10, -93)	4
(10, -96)	4
(9, -80)	12
(9, -81)	4
(9, -82)	4
(9, -83)	8

TABLA 3.18. OPTIMIZACIÓN MODELO CONGELADO CONFIGURACIONES 4G

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	13.571	11.560	13.876	15.678
Predicted Original SINR	14.251	12.012	15.074	16.510
Error SINR	0.890	0.420	0.796	1.152
Predicted Final SINR	18.275	17.877	18.525	19.065
Real Gain SINR	4.705	3.049	4.607	5.876
Predicted Gain SINR	4.024	2.482	3.891	5.539
Factor: Predicted Gain / Error	11.320	2.494	4.328	9.367

TABLA 3.19. OPTIMIZACIÓN MODELO SIN CONGELAR ISSUE
CELLS 4G

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	16.281	14.908	17.065	18.850
Predicted Original SINR	16.285	14.862	17.142	18.730
Error SINR	0.528	0.157	0.374	0.699
Predicted Final SINR	16.293	14.757	17.293	18.802
Real Gain SINR	0.012	-0.430	-0.082	0.339
Predicted Gain SINR	0.008	-0.123	-0.057	-0.023
Factor: Predicted Gain / Error	-2.187	-0.440	-0.176	-0.048

TABLA 3.20. OPTIMIZACIÓN MODELO SIN CONGELAR
FIRST-HOP NEIGHBOR CELLS 4G

los vecinos más importantes para el modelo en el momento de realizar una predicción. Para ello se han utilizado cuatro técnicas diferentes:

- Saliency Maps
- SmoothGrad
- Guided Backpropagation
- Deconvnet

En este caso, para las implementaciones de las diferentes técnicas en el dominio las GNNs, se ha seguido lo expuesto en [25]. Para mostrar un ejemplo de cada una, se han generado las Figuras 3.10, 3.11, 3.12 y 3.13. En estos ejemplos se puede observar que los resultados de muchas de las técnicas del gradiente son muy similares, a diferencia del SmoothGrad que parece que ofrece unos resultados un poco más diferentes.

Es importante resaltar que al ser modelos probabilísticos se ha optado por congelar las capas y ejecutar las técnicas de explicabilidad como se haría en una GNN sin capas

Configuración	Número de Celdas
(10, -80)	52
(10, -81)	4
(10, -82)	4
(10, -85)	8
(10, -86)	8
(10, -87)	4
(10, -88)	4
(10, -89)	4
(10, -90)	8
(10, -96)	4
(9, -80)	4
(9, -82)	4
(9, -84)	4
(9, -85)	4
(9, -86)	4

TABLA 3.21. OPTIMIZACIÓN MODELO SIN CONGELAR
CONFIGURACIONES 4G

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	15.165	13.768	15.536	16.765
Predicted Original SINR	15.408	14.111	15.643	16.784
Error SINR	0.811	0.209	0.461	1.008
Predicted Final SINR	16.937	15.970	17.006	18.212
Real Gain SINR	1.772	0.244	1.379	2.264
Predicted Gain SINR	1.529	0.053	0.885	2.181
Factor: Predicted Gain / Error	5.673	0.118	1.374	4.548

TABLA 3.22. OPTIMIZACIÓN O6 ISSUE CELLS 4G

bayesianas. Un estudio más en profundidad de este tema queda fuera del alcance del presente proyecto, y se dejaría como futuro trabajo.

Como una visualización en el caso de un método de explicabilidad no nos puede revelar información sobre si realmente nos ofrece información verídica, en este trabajo se incluyen unas métricas similares a las expuestas en [25]. En dicho artículo se intenta construir unas métricas, denominadas *loyalty* e *inverse loyalty*, basadas en borrar los vecinos que son más importantes según la técnica de explicabilidad utilizada:

- Loyalty: Los vecinos se ordenan en orden descendente de importancia. Después, se borran en porcentajes fijos. Como los primeros que se eliminan son los más impor-

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	16.331	14.235	16.816	18.873
Predicted Original SINR	16.508	14.549	16.969	18.894
Error SINR	0.432	0.112	0.260	0.536
Predicted Final SINR	16.585	14.622	17.069	18.966
Real Gain SINR	0.255	-0.076	0.142	0.448
Predicted Gain SINR	0.078	-0.025	0.022	0.052
Factor: Predicted Gain / Error	0.897	-0.061	0.064	0.326

TABLA 3.23. OPTIMIZACIÓN O6 FIRST-HOP NEIGHBOR CELLS
4G

tantes, primero el rendimiento del modelo debería cambiar drásticamente, para ir evolucionando de una forma menos brusca al final.

- Inverse Loyalty: El orden se invierte, comenzando ahora por los vecinos menos importantes. Por tanto, la curva debería ser suave al principio y más abrupta al final, que es cuando se borran los vecinos importantes.

Como en [25] solo se tienen en cuenta problemas de clasificación, aquí se ha reformulado para tener en cuenta un caso de regresión. Los resultados se pueden observar en las Figuras 3.14 y 3.15. Se pueden observar resultados parecidos a los expuestos en [25], donde se aprecia lo que se ha explicado anteriormente de la evolución de las curvas. Además, cabe resaltar la poca diferencia que hay entre la predicción original y cuando se han eliminado todos los vecinos, lo que escenifica que una gran parte de la importancia reside en el nodo en cuestión y no en los vecinos.

Configuración	Número de Celdas
(10, -101)	3
(10, -102)	3
(10, -103)	6
(10, -104)	3
(10, -105)	9
(10, -106)	3
(10, -108)	6
(10, -109)	3
(10, -110)	6
(10, -80)	42
(10, -81)	3
(10, -82)	3
(10, -83)	3
(10, -84)	3
(10, -85)	6
(10, -88)	6
(10, -89)	3
(10, -90)	3
(10, -92)	3
(10, -93)	5
(10, -95)	6
(10, -96)	9
(10, -97)	3
(10, -99)	3
(8, -80)	3
(8, -83)	3
(8, -87)	3
(9, -85)	5
(9, -87)	1

TABLA 3.24. OPTIMIZACIÓN O6 CONFIGURACIONES 4G

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	15.998	13.453	16.059	17.892
Predicted Original SINR	15.926	13.215	16.146	17.710
Error SINR	0.994	0.264	0.632	1.405
Predicted Final SINR	16.324	13.516	16.590	18.066
Real Gain SINR	0.326	-0.557	0.262	1.173
Predicted Gain SINR	0.398	-0.140	0.214	0.805
Factor: Predicted Gain / Error	-3.264	-0.148	0.337	1.258

TABLA 3.25. OPTIMIZACIÓN MODELO CONGELADO *ISSUE CELLS 5G*

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	20.214	18.439	20.209	22.081
Predicted Original SINR	20.179	18.381	20.428	21.870
Error SINR	0.586	0.183	0.416	0.772
Predicted Final SINR	20.206	18.395	20.415	21.874
Real Gain SINR	-0.008	-0.487	-0.003	0.417
Predicted Gain SINR	0.027	-0.012	0.016	0.050
Factor: Predicted Gain / Error	0.150	-0.026	0.032	0.135

TABLA 3.26. OPTIMIZACIÓN MODELO CONGELADO *FIRST-HOP NEIGHBOR CELLS 5G*

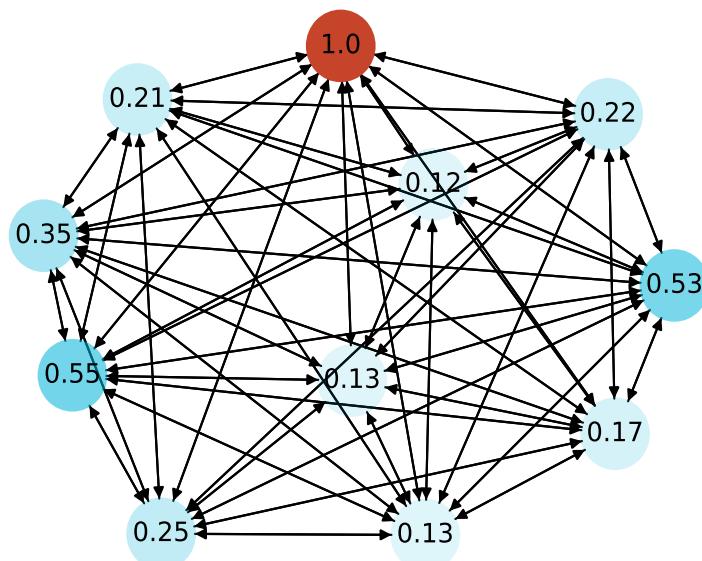


Fig. 3.10. Example Saliency Map

Configuración	Número de Celdas
-99	10
-97	5
-96	18
-95	5
-94	10
-93	15
-90	15
-89	10
-88	5
-86	5
-85	3
-83	17
-82	14
-81	5

**TABLA 3.27. OPTIMIZACIÓN MODELO CONGELADO
CONFIGURACIONES 5G**

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	15.998	13.453	16.059	17.892
Predicted Original SINR	15.946	13.177	16.172	17.756
Error SINR	1.009	0.232	0.639	1.446
Predicted Final SINR	16.407	13.994	16.417	18.044
Real Gain SINR	0.409	-0.387	0.249	1.019
Predicted Gain SINR	0.461	-0.115	0.220	0.889
Factor: Predicted Gain / Error	0.054	-0.139	0.381	1.091

**TABLA 3.28. OPTIMIZACIÓN MODELO SIN CONGELAR ISSUE
CELLS 5G**

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	20.214	18.439	20.209	22.081
Predicted Original SINR	20.191	18.358	20.419	21.869
Error SINR	0.584	0.199	0.411	0.778
Predicted Final SINR	20.189	18.413	20.423	21.868
Real Gain SINR	-0.025	-0.498	-0.021	0.398
Predicted Gain SINR	-0.002	-0.030	-0.004	0.023
Factor: Predicted Gain / Error	-0.124	-0.079	-0.008	0.065

**TABLA 3.29. OPTIMIZACIÓN MODELO SIN CONGELAR
FIRST-HOP NEIGHBOR CELLS 5G**

Configuración	Número de Celdas
-102	5
-101	10
-100	5
-99	7
-97	5
-95	15
-94	5
-93	10
-90	7
-89	10
-88	5
-87	8
-86	10
-85	10
-84	5
-83	5
-82	5
-81	10

**TABLA 3.30. OPTIMIZACIÓN MODELO SIN CONGELAR
CONFIGURACIONES 5G**

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	18.29	16.82	18.84	20.26
Predicted Original SINR	18.76	17.49	18.88	20.32
Predicted Final SINR	20.56	19.65	20.76	21.62
Real Gain SINR	2.27	0.64	1.76	3.11
Predicted Gain SINR	1.80	0.51	1.58	2.83
Factor: Predicted Gain / Error	4.28	1.21	3.74	6.72

TABLA 3.31. OPTIMIZACIÓN O7 *ISSUE CELLS 5G*

KPI	Media	25th-percentil	50th-percentil	75th-percentil
Original SINR	22.13	21.39	22.34	23.18
Predicted Original SINR	22.15	21.43	22.36	23.09
Predicted Final SINR	22.24	21.59	22.47	23.15
Real Gain SINR	0.11	-0.07	0.11	0.30
Predicted Gain SINR	0.09	0.04	0.07	0.09
Factor: Predicted Gain / Error	0.21	0.11	0.17	0.22

TABLA 3.32. OPTIMIZACIÓN O7 *FIRST-HOP NEIGHBOR CELLS 5G*

Configuración	Número de Celdas
-100	6
-98	1
-97	1
-96	1
-95	1
-92	2
-90	2
-89	1
-87	1
-85	1
-81	1
-80	1

TABLA 3.33. OPTIMIZACIÓN O7 CONFIGURACIONES 5G

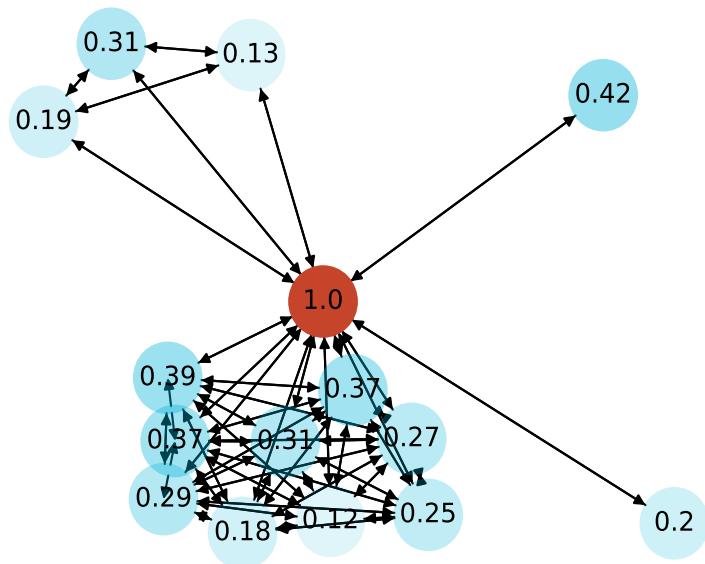


Fig. 3.11. Example SmoothGrad

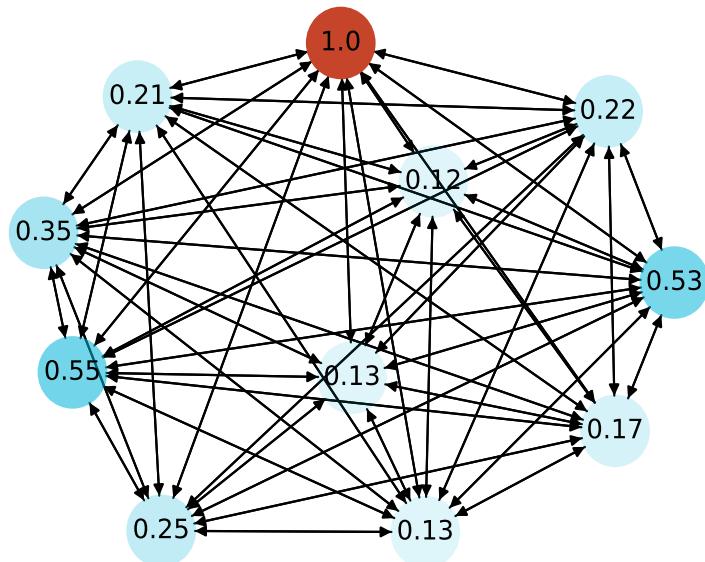


Fig. 3.12. Example Guided Backpropagation

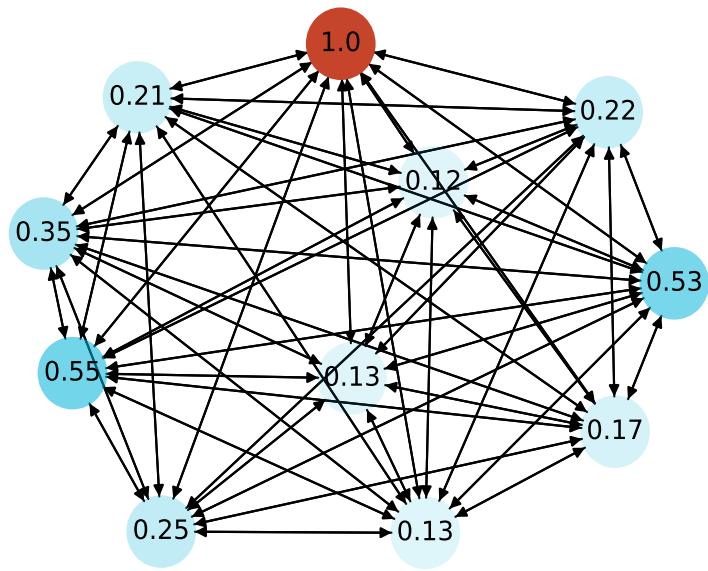


Fig. 3.13. Example Deconvnet

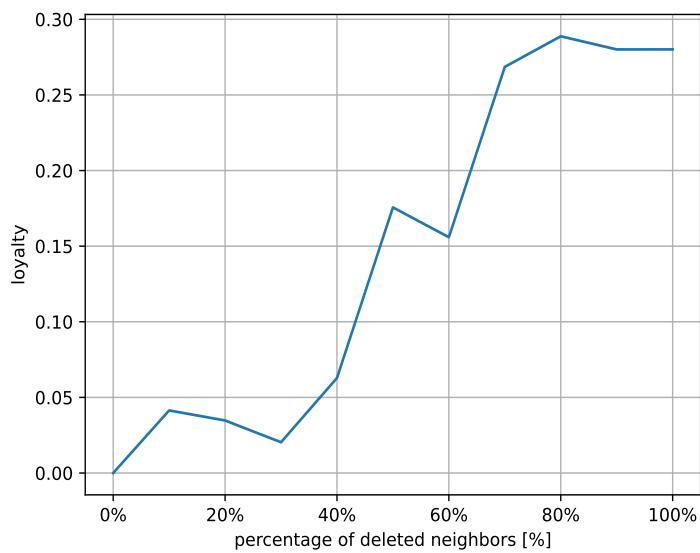


Fig. 3.14. Loyalty Saliency Map

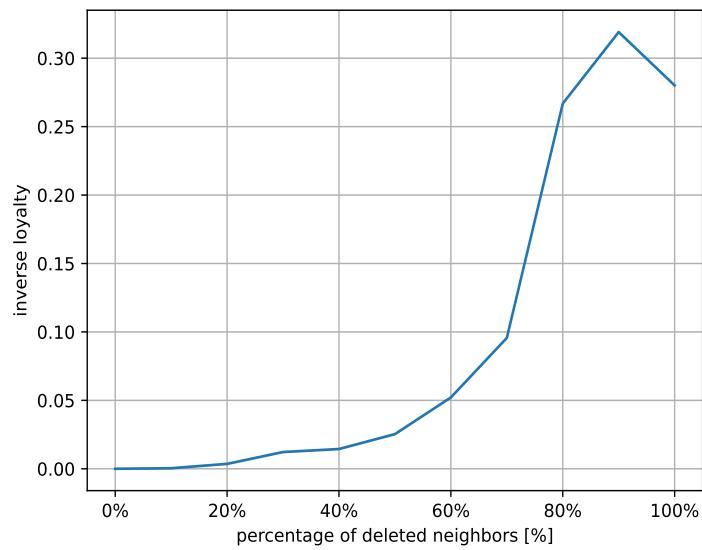


Fig. 3.15. Inverse loyalty Saliency Map

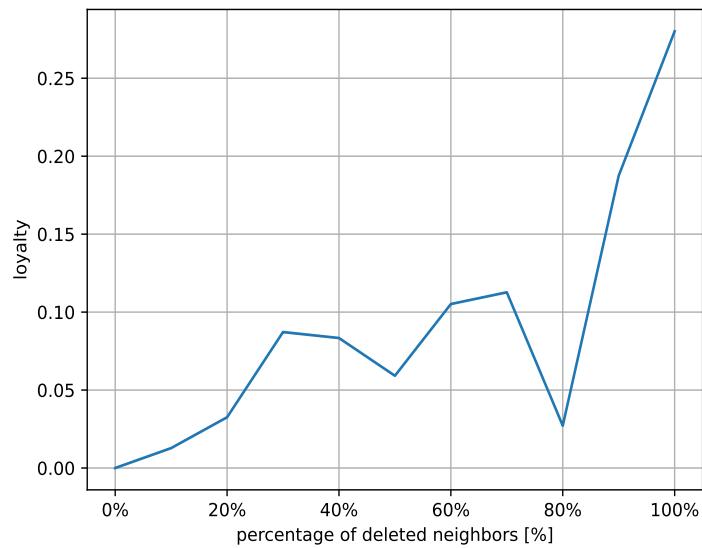


Fig. 3.16. Loyalty SmoothGrad

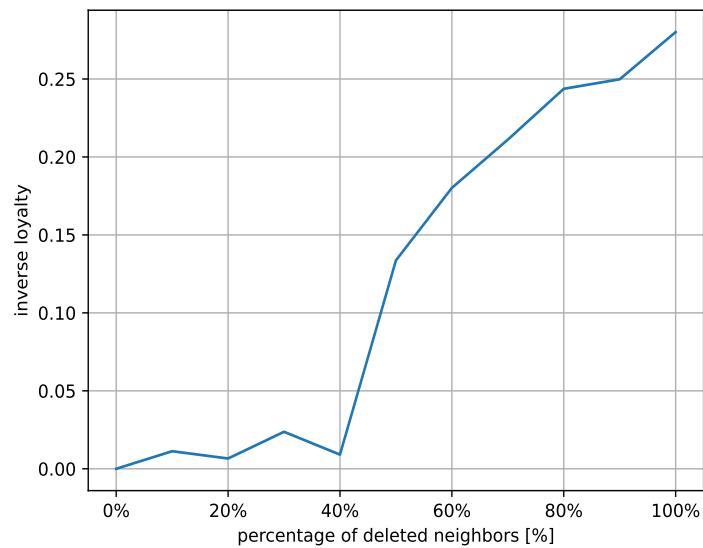


Fig. 3.17. Inverse loyalty SmoothGrad

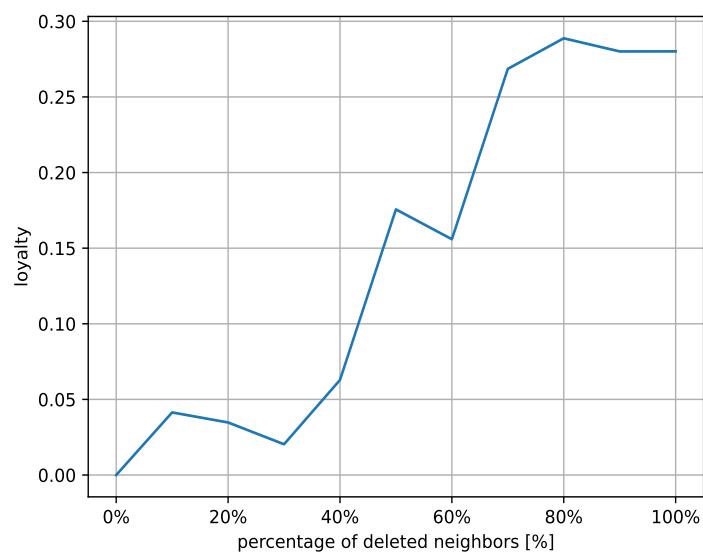


Fig. 3.18. Loyalty Guided Backpropagation

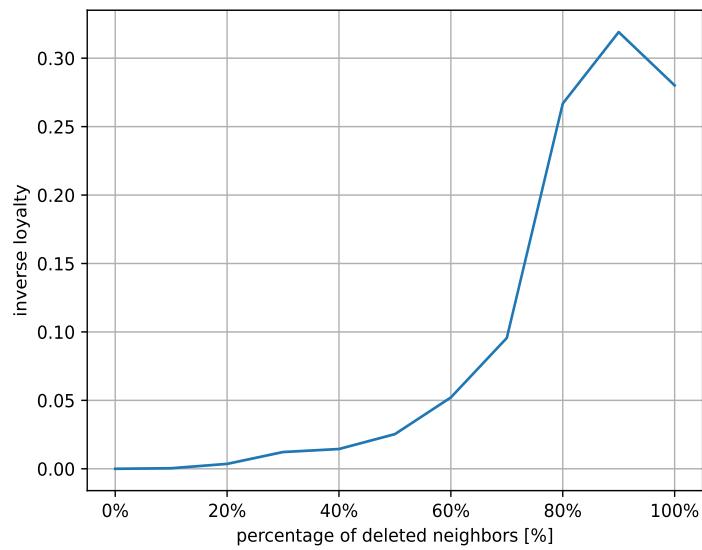


Fig. 3.19. Inverse loyalty Guided Backpropagation

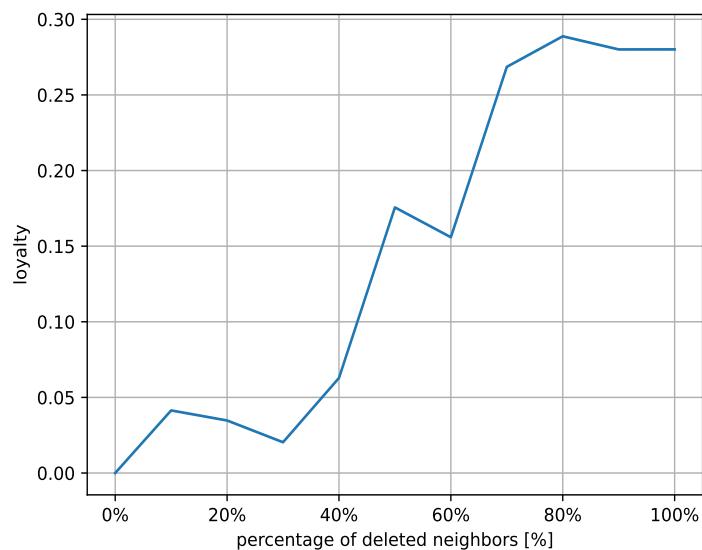


Fig. 3.20. Loyalty Deconvnet

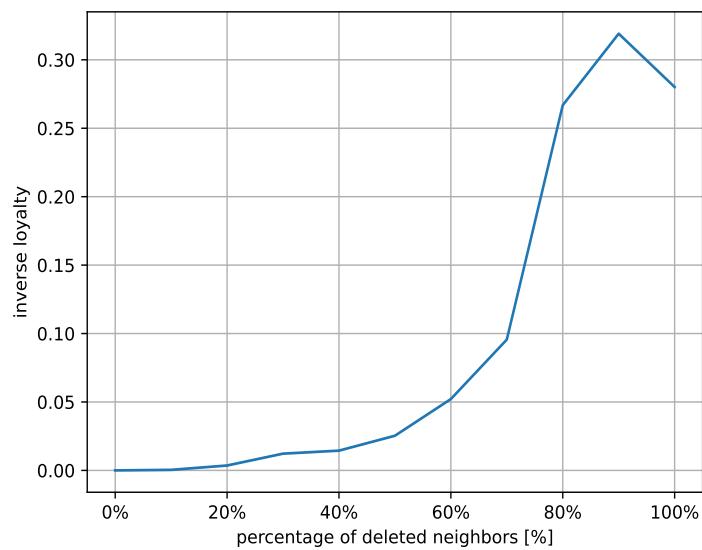


Fig. 3.21. Inverse loyalty Deconvnet

4. INVESTIGACIÓN OPEN-SOURCE

Como la mayoría de los datos y el código de la solución de Ericsson son confidenciales, en este capítulo se mostrará la investigación realizada con un conjunto de datos open-source. Además, para asegurar la reproducibilidad de esta parte, todo el código se hará público en <https://github.com/oslllogon/bayesian-gnn>.

4.1. Datos

En primer lugar, los datos que se utilizarán en este capítulo serán los del conjunto QM9 [6]. En este caso, será un problema de regresión con múltiples variables a predecir, lo que se puede observar en la Tabla 4.1.

Target	Property	Description	Unit
0	μ	Dipole moment	D
1	α	Isotropic polarizability	a_0^3
2	ϵ_{HOMO}	Highest occupied molecular orbital energy	eV
3	ϵ_{LUMO}	Lowest unoccupied molecular orbital energy	eV
4	$\Delta\epsilon$	Gap between ϵ_{HOMO} and ϵ_{LUMO}	eV
5	$\langle R^2 \rangle$	Electronic spatial extent	a_0^2
6	ZPVE	Zero point vibrational energy	eV
7	U_0	Internal energy at 0K	eV
8	U	Internal energy at 298.15K	eV
9	H	Enthalpy at 298.15K	eV
10	G	Free energy at 298.15K	eV
11	c_v	Heat capacity at 298.15K	$\frac{\text{cal}}{\text{mol}\cdot\text{K}}$
12	U_0^{ATOM}	Atomization energy at 0K	eV
13	U^{ATOM}	Atomization energy at 298.15K	eV
14	H^{ATOM}	Atomization enthalpy at 298.15K	eV
15	G^{ATOM}	Atomization free energy at 298.15K	eV
16	A	Rotational constant	GHz
17	B	Rotational constant	GHz
18	C	Rotational constant	GHz

TABLA 4.1. DATOS QM9 FEY/LENSSEN/2019

En este caso el problema a solucionar será una regresión, pero a nivel de grafo en vez de a nivel de nodo. Se ha intentado buscar un problema que fuera similar al expuesto en el

caso de Ericsson, pero ante la falta de conjuntos de datos públicos de cierta calidad para la tarea de regresión a nivel de nodo se ha optado por esta alternativa.

4.2. Modelos

En este caso, los experimentos tendrán una arquitectura similar, comparando diferentes capas GNNs y diferentes implementaciones para las capas bayesianas. En concreto, el modelo estará formado por los siguientes bloques:

- Dos capas GNN, para capturar la influencia de los vecinos en cada nodo del grafo.
- Una capa de Average Pooling para combinar la información de todos los nodos de cada grafo.
- Una capa lineal para realizar la predicción final utilizando como entrada el tamaño de representación de las capas GNNs.

Por otra parte, como se ha mencionado antes, se realizarán diferentes experimentos sobre esta arquitectura común. El primero de ellos será el tipo de GNN que se utilizará:

- Graph Convolutional Network [8]. Elegida debido a su popularidad por ser uno de los primeros modelos que implementaba la arquitectura *Message Passing*.
- Graph Attention Networks v2 [33]. Escogida también debido a su popularidad.

Además, también se compararán tres tipos de implementaciones de las redes neuronales:

- Redes normales, sin ninguna implementación bayesiana. Se utilizarán como punto de referencia para la métrica del MAE calculada los otros modelos.
- Redes bayesianas con la implementación de *bayes by backprop* [11]. En el caso de las GNNs cada capa lineal en su interior se sustituirá por una lineal bayesiana, sin modificar su estructura o el resto de capas.
- MC Dropout. Se utilizará el dropout como aproximación de una red bayesiana, tal y como se detalla en [12]. En este caso se utilizará una probabilidad de 0.5 y para las GNNs el enfoque será equivalente al de la opción anterior.

4.3. Resultados

En esta sección mostraremos los resultados obtenidos en diferentes tablas. En concreto, solo se mostrarán los del conjunto de validación, y el conjunto de test para el modelo que demuestre un mayor rendimiento. Las métricas que se utilizarán en este caso son similares a las de la Sección 3.3.6:

- MAE.
- pi-ci 1 std.
- pi-ci 2 std.
- pi-ci 3 std.
- pi-ci 0.1 std.
- pi-ci 0.2 std.
- pi-ci 0.3 std.

Los resultados se pueden analizar en las Tablas 4.2 4.3, 4.4, 4.5, 4.6 y 4.7. De dichas tablas se pueden sacar dos conclusiones principales. En este caso, se aprecia una diferencia significativa en el rendimiento de los modelos probabilísticos y sus versiones normales. Aunque esto puede ser normal, ya que optimizar estos modelos es una tarea más complicada porque no solo están intentando minimizar el error en la predicción, es importante comentar la diferencia con el caso de uso de Ericsson en el que modelos probabilísticos obtenían un resultado mucho mejor (menos del 1 % de error). Una explicación plausible puede ser la cantidad de datos, ya que en el caso de Ericsson era un conjunto de un gran tamaño (medio millón de muestras).

El otro resultado que se extrae al analizar es que entre las versiones probabilísticas, aunque el dropout era en principio una aproximación, los resultados de los modelos bayesianos son considerablemente mejores, aunque no se ha hecho un estudio exhaustivo de la probabilidad del dropout para estos casos. Como el mejor modelo probabilístico es el GAT bayesiano, se ha ejecutado el conjunto test con este modelo, Tabla 4.8, ofreciendo resultados similares a los que se podían apreciar en validación.

Por último, como los intervalos eran demasiado amplios, con las medidas normales de 1, 2 y 3 desviaciones estándares, i.e., contenían a todas las muestras del conjunto de datos, se han añadido el porcentaje de los intervalos de 0.1, 0.2 y 0.3 desviaciones estándar. Aunque para comprobar el motivo de la diferencia con el caso de Ericsson habría que realizar un estudio exhaustivo, lo que queda fuera del alcance del presente trabajo, una razón plausible sería una diferente distribución de los datos en los objetivos a predecir.

Objetivo	MAE	pi-ci 1 std	pi-ci 2 std	pi-ci 3 std	pi-ci 0.1 std	pi-ci 0.2 std	pi-ci 0.3 std
0	1.025	0.000	0.000	0.000	0.680	0.950	0.997
1	3.703	0.000	0.000	0.000	0.680	0.950	0.997
2	0.399	0.000	0.000	0.000	0.680	0.950	0.997
3	0.779	0.000	0.000	0.000	0.680	0.950	0.997
4	0.857	0.000	0.000	0.000	0.680	0.950	0.997
5	191.607	0.000	0.000	0.000	0.680	0.950	0.997
6	0.215	0.000	0.000	0.000	0.680	0.950	0.997
7	476.376	0.000	0.000	0.000	0.680	0.950	0.997
8	455.298	0.000	0.000	0.000	0.680	0.950	0.997
9	459.056	0.000	0.000	0.000	0.680	0.950	0.997
10	457.075	0.000	0.000	0.000	0.680	0.950	0.997
11	2.154	0.000	0.000	0.000	0.680	0.950	0.997
12	3.213	0.000	0.000	0.000	0.680	0.950	0.997
13	3.225	0.000	0.000	0.000	0.680	0.950	0.997
14	3.228	0.000	0.000	0.000	0.680	0.950	0.997
15	2.977	0.000	0.000	0.000	0.680	0.950	0.997
16	296.696	0.000	0.000	0.000	0.680	0.950	0.997
17	0.347	0.000	0.000	0.000	0.680	0.950	0.997
18	0.358	0.000	0.000	0.000	0.680	0.950	0.997

TABLA 4.2. RESULTADOS GCN

Objetivo	MAE	pi-ci 1 std	pi-ci 2 std	pi-ci 3 std	pi-ci 0.1 std	pi-ci 0.2 std	pi-ci 0.3 std
0	1.177	0.996	0.999	1.000	0.316	0.049	0.003
1	5.995	0.998	1.000	1.000	0.318	0.050	0.003
2	0.615	0.944	0.999	1.000	0.264	0.049	0.003
3	1.375	0.700	0.973	1.000	0.025	0.023	0.003
4	1.367	0.915	1.000	1.000	0.235	0.050	0.003
5	186.801	0.953	0.995	1.000	0.273	0.045	0.003
6	0.592	0.912	0.999	1.000	0.232	0.049	0.003
7	764.342	0.992	0.999	1.000	0.312	0.049	0.003
8	806.929	0.990	0.999	1.000	0.310	0.049	0.003
9	771.167	0.991	0.999	1.000	0.311	0.049	0.003
10	810.046	0.989	0.999	1.000	0.309	0.049	0.003
11	2.860	0.963	0.999	1.000	0.283	0.049	0.003
12	7.409	0.955	0.999	1.000	0.275	0.049	0.003
13	7.532	0.941	0.998	1.000	0.261	0.048	0.003
14	7.245	0.958	0.999	1.000	0.278	0.049	0.003
15	6.950	0.940	0.998	1.000	0.260	0.048	0.003
16	5538.003	1.000	1.000	1.000	0.320	0.050	0.003
17	7.214	1.000	1.000	1.000	0.320	0.050	0.003
18	3.946	1.000	1.000	1.000	0.320	0.050	0.003

TABLA 4.3. RESULTADOS BAYESIAN GCN

Objetivo	MAE	pi-ci 1 std	pi-ci 2 std	pi-ci 3 std	pi-ci 0.1 std	pi-ci 0.2 std	pi-ci 0.3 std
0	1.782	0.300	0.603	0.810	0.380	0.347	0.187
1	41.070	0.016	0.960	0.999	0.664	0.010	0.002
2	2.848	0.130	0.994	1.000	0.550	0.044	0.003
3	1.616	0.974	0.999	1.000	0.294	0.049	0.003
4	1.669	0.994	1.000	1.000	0.314	0.050	0.003
5	532.563	0.734	0.979	0.998	0.054	0.029	0.001
6	1.510	0.973	0.996	0.998	0.293	0.046	0.001
7	4282.539	0.387	0.996	1.000	0.293	0.046	0.003
8	4255.400	0.454	0.997	1.000	0.226	0.047	0.003
9	4290.455	0.370	0.996	1.000	0.310	0.046	0.003
10	4267.202	0.427	0.996	1.000	0.253	0.046	0.003
11	11.726	0.744	0.999	1.000	0.064	0.049	0.003
12	22.263	0.026	0.889	0.986	0.654	0.061	0.011
13	22.206	0.031	0.895	0.987	0.649	0.055	0.010
14	22.340	0.032	0.897	0.987	0.648	0.053	0.010
15	20.319	0.030	0.892	0.986	0.650	0.058	0.011
16	89.248	1.000	1.000	1.000	0.320	0.050	0.003
17	0.687	0.349	0.721	0.921	0.331	0.229	0.076
18	0.399	0.772	0.973	0.994	0.092	0.023	0.003

TABLA 4.4. RESULTADOS DROPOUT GCN

Objetivo	MAE	pi-ci 1 std	pi-ci 2 std	pi-ci 3 std	pi-ci 0.1 std	pi-ci 0.2 std	pi-ci 0.3 std
0	1.020	0.000	0.000	0.000	0.680	0.950	0.997
1	18.822	0.000	0.000	0.000	0.680	0.950	0.997
2	0.559	0.000	0.000	0.000	0.680	0.950	0.997
3	1.170	0.000	0.000	0.000	0.680	0.950	0.997
4	1.711	0.000	0.000	0.000	0.680	0.950	0.997
5	170.837	0.000	0.000	0.000	0.680	0.950	0.997
6	0.213	0.000	0.000	0.000	0.680	0.950	0.997
7	458.576	0.000	0.000	0.000	0.680	0.950	0.997
8	453.102	0.000	0.000	0.000	0.680	0.950	0.997
9	469.544	0.000	0.000	0.000	0.680	0.950	0.997
10	450.945	0.000	0.000	0.000	0.680	0.950	0.997
11	2.035	0.000	0.000	0.000	0.680	0.950	0.997
12	3.101	0.000	0.000	0.000	0.680	0.950	0.997
13	3.163	0.000	0.000	0.000	0.680	0.950	0.997
14	3.155	0.000	0.000	0.000	0.680	0.950	0.997
15	2.888	0.000	0.000	0.000	0.680	0.950	0.997
16	212.454	0.000	0.000	0.000	0.680	0.950	0.997
17	0.698	0.000	0.000	0.000	0.680	0.950	0.997
18	0.273	0.000	0.000	0.000	0.680	0.950	0.997

TABLA 4.5. RESULTADOS GAT

Objetivo	MAE	pi-ci 1 std	pi-ci 2 std	pi-ci 3 std	pi-ci 0.1 std	pi-ci 0.2 std	pi-ci 0.3 std
0	1.083	0.927	0.996	0.998	0.247	0.046	0.001
1	5.649	0.979	0.999	1.000	0.299	0.049	0.003
2	0.596	0.752	0.952	0.993	0.072	0.007	0.004
3	1.153	0.527	0.834	0.987	0.153	0.116	0.010
4	1.322	0.556	0.926	0.997	0.124	0.025	0.001
5	190.806	0.803	0.953	0.986	0.123	0.004	0.011
6	0.339	0.903	0.996	1.000	0.223	0.046	0.003
7	685.571	0.939	0.994	0.999	0.259	0.044	0.002
8	724.504	0.919	0.991	0.999	0.239	0.041	0.002
9	697.056	0.926	0.992	0.999	0.246	0.042	0.002
10	719.653	0.922	0.992	0.999	0.242	0.042	0.002
11	2.610	0.750	0.976	0.997	0.070	0.026	0.001
12	4.383	0.903	0.993	0.999	0.223	0.043	0.002
13	4.600	0.892	0.994	0.999	0.212	0.044	0.002
14	4.491	0.883	0.987	0.997	0.203	0.037	0.001
15	3.886	0.911	0.993	0.999	0.231	0.043	0.002
16	2656.354	1.000	1.000	1.000	0.320	0.050	0.003
17	2.400	1.000	1.000	1.000	0.320	0.050	0.003
18	0.704	1.000	1.000	1.000	0.320	0.050	0.003

TABLA 4.6. RESULTADOS BAYESIAN GAT

Objetivo	MAE	pi-ci 1 std	pi-ci 2 std	pi-ci 3 std	pi-ci 0.1 std	pi-ci 0.2 std	pi-ci 0.3 std
0	1.484	0.558	0.850	0.976	0.122	0.100	0.021
1	34.184	0.498	1.000	1.000	0.182	0.050	0.003
2	2.780	0.182	0.996	1.000	0.498	0.046	0.003
3	1.584	0.950	1.000	1.000	0.270	0.050	0.003
4	2.334	0.953	0.999	1.000	0.273	0.049	0.003
5	584.223	0.585	0.970	0.997	0.095	0.020	0.001
6	1.802	0.480	0.995	0.998	0.200	0.045	0.001
7	4128.574	0.705	0.998	1.000	0.025	0.048	0.003
8	4098.914	0.739	0.998	1.000	0.059	0.048	0.003
9	4116.223	0.720	0.998	1.000	0.040	0.048	0.003
10	4110.240	0.724	0.998	1.000	0.044	0.048	0.003
11	12.766	0.479	1.000	1.000	0.201	0.050	0.003
12	18.607	0.624	0.985	0.998	0.056	0.035	0.001
13	18.841	0.604	0.985	0.998	0.076	0.035	0.001
14	18.942	0.615	0.985	0.998	0.065	0.035	0.001
15	17.102	0.617	0.984	0.998	0.063	0.034	0.001
16	53.863	1.000	1.000	1.000	0.320	0.050	0.003
17	0.572	0.621	0.931	0.985	0.059	0.019	0.012
18	0.365	0.972	0.998	0.999	0.292	0.048	0.002

TABLA 4.7. RESULTADOS DROPOUT GAT

Objetivo	MAE	pi-ci 1 std	pi-ci 2 std	pi-ci 3 std	pi-ci 0.1 std	pi-ci 0.2 std	pi-ci 0.3 std
0	1.076	0.929	0.996	0.998	0.249	0.046	0.001
1	5.563	0.983	0.999	1.000	0.303	0.049	0.003
2	0.435	0.823	0.968	0.997	0.143	0.018	0.001
3	0.862	0.508	0.879	0.996	0.172	0.071	0.002
4	0.925	0.658	0.979	0.999	0.024	0.029	0.002
5	190.622	0.793	0.950	0.986	0.113	0.005	0.011
6	0.338	0.897	0.996	1.000	0.217	0.046	0.003
7	682.853	0.934	0.994	1.000	0.254	0.044	0.003
8	726.112	0.915	0.992	0.999	0.235	0.042	0.002
9	691.626	0.923	0.992	0.999	0.243	0.042	0.002
10	707.220	0.929	0.993	0.999	0.249	0.043	0.002
11	2.641	0.739	0.974	0.997	0.059	0.024	0.001
12	4.352	0.904	0.993	0.999	0.224	0.043	0.002
13	4.573	0.896	0.995	0.999	0.216	0.045	0.002
14	4.443	0.883	0.987	0.998	0.203	0.037	0.001
15	3.897	0.909	0.994	0.999	0.229	0.044	0.002
16	2491.028	1.000	1.000	1.000	0.320	0.050	0.003
17	2.472	1.000	1.000	1.000	0.320	0.050	0.003
18	0.652	1.000	1.000	1.000	0.320	0.050	0.003

TABLA 4.8. RESULTADOS BAYESIAN GAT CONJUNTO TEST

5. CONCLUSIONES

En este trabajo se han conseguido alcanzar resultados significativos en cuanto al campo de las GNNs se refiere. En concreto, los avances presentados se pueden resumir en las siguientes contribuciones:

- Conseguir generar a través de estos métodos intervalos de confianza lo suficientemente amplios como para poder utilizarlos para proporcionar estimaciones de incertidumbre.
- La implementación del primer producto industrial en utilizar la tecnología de BGNNs, combinando estas redes neuronales con algoritmos de optimización para mejorar la eficiencia de la red de telecomunicaciones.
- Aplicación de la Explicabilidad a las BGNNs, siendo la primera vez que esto se realiza en la literatura.
- Reformulación de las métricas de *loyalty* e *inverse loyalty* para poder utilizarlas en problemas de regresión.

En primer lugar, aunque las BNNs surgieron hace unos años, prácticamente no se han utilizado en comparación con otras muchas tecnologías en el campo del *deep learning*. Una de las posibles explicaciones es su dificultad para entrenarlas o generar intervalos de confianza que puedan ser útiles, lo que podría ser un posible motivo, aunque no queda claro en base a la literatura publicada. Sin embargo, el hecho de que las arquitecturas GNNs tiendan a ser mucho menos profundas proporciona una facilidad en el entrenamiento que puede ser aprovechada para intentar entrenar esta clase de modelos.

Por otra parte, una de las grandes limitaciones en los modelos de Inteligencia Artificial es la cantidad de los datos. Esto se ve intensificado en esta clase de modelos, donde hemos podido ver como al utilizar un conjunto de datos más limitado (el conjunto público en nuestra investigación open-source) los intervalos de confianza que generamos son mucho más estrechos y los resultados en las predicciones tienden a dar peores resultados en métricas como el MAE. Sin embargo, se demuestra que al utilizar un conjunto de datos a gran escala, con más de medio millón de muestras, conseguimos un error menor al 1 % y unos intervalos de confianza mucho más amplios. De hecho, aunque como se explica en la Sección 2, hay cierta literatura publicada, esta es la primera aplicación industrial que se consigue con este tipo de tecnología.

Además, aunque en los recientemente, como ya se ha mencionado en la Sección 2, ha habido una intensa labor de investigación acerca de la explicabilidad en redes neuronales, aún no se habían publicado artículos aplicando explicabilidad a BNNs o BGNNs. Por tanto, este trabajo es pionero, al aplicar por primera vez la explicabilidad a este nuevo de

tipo de redes, además de reformular las métricas existentes para poder evaluar los métodos de explicabilidad en un escenario de regresión.

Por último, el presente trabajo abre varias líneas de investigación prometedoras para el futuro, entre ellas las más importantes podrían ser las siguientes:

- Establecer unas métricas que permitan comparar la capacidad de los modelos para generar intervalos de confianza fiables, cuando no se puede establecer qué distribución tiene la salida que se está prediciendo.
- Investigar y comparar las diferentes formas que pueden existir de aplicar explicabilidad a las BNNs y BGNNs, como hacer la media, congelar las capas o alguna otra estrategia.
- Investigar acerca de intervalos de confianza en la explicabilidad, donde con las muestras generadas se podría definir una incertidumbre en las explicaciones.

BIBLIOGRAFÍA

- [1] A. Kirillov et al., “Segment Anything,” *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3992-4003, 2023. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:257952310>.
- [2] T. Brown et al., “Language Models are Few-Shot Learners,” en *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan y H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877-1901. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.
- [3] M. Dikmen y C. Burns, “Trust in autonomous vehicles: The case of Tesla Autopilot and Summon,” en *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 1093-1098. doi: [10.1109/SMC.2017.8122757](https://doi.org/10.1109/SMC.2017.8122757).
- [4] H. Gholamalinezhad y H. Khosravi, “Pooling methods in deep neural networks, a review,” *arXiv preprint arXiv:2009.07485*, 2020.
- [5] L. Wu, P. Cui, J. Pei y L. Zhao, *Graph Neural Networks: Foundations, Frontiers, and Applications*. Singapore: Springer Singapore, 2022, p. 725.
- [6] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals y G. E. Dahl, “Neural Message Passing for Quantum Chemistry,” en *International Conference on Machine Learning*, 2017. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:9665943>.
- [7] M. Fey y J. E. Lenssen, “Fast Graph Representation Learning with PyTorch Geometric,” en *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [8] T. N. Kipf y M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” en *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. [En línea]. Disponible en: <https://openreview.net/forum?id=SJu4ayYgl>.
- [9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò e Y. Bengio, “Graph Attention Networks,” *6th International Conference on Learning Representations*, 2017.
- [10] C. K. Williams y C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.
- [11] C. Blundell, J. Cornebise, K. Kavukcuoglu y D. Wierstra, “Weight Uncertainty in Neural Network,” en *International Conference on Machine Learning*, 2015. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:39895556>.

- [12] Y. Gal y Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” en *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan y K. Q. Weinberger, eds., ép. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 20–22 Jun de 2016, pp. 1050-1059. [En línea]. Disponible en: <https://proceedings.mlr.press/v48/gal16.html>.
- [13] Y. Zhang, S. Pal, M. J. Coates y D. Üstebay, “Bayesian graph convolutional neural networks for semi-supervised classification,” en *AAAI Conference on Artificial Intelligence*, 2018. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:53786372>.
- [14] A. Hasanzadeh et al., “Bayesian Graph Neural Networks with Adaptive Connection Sampling,” en *International Conference on Machine Learning*, 2020. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:219530761>.
- [15] G. Lamb y B. Paige, “Bayesian graph neural networks for molecular property prediction,” *arXiv preprint arXiv:2012.02089*, 2020.
- [16] K. Simonyan, A. Vedaldi y A. Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps,” *CoRR*, vol. abs/1312.6034, 2013. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:1450294>.
- [17] J. T. Springenberg, A. Dosovitskiy, T. Brox y M. A. Riedmiller, “Striving for Simplicity: The All Convolutional Net,” *CoRR*, vol. abs/1412.6806, 2014. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:12998557>.
- [18] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas y M. Wattenberg, “SmoothGrad: removing noise by adding noise,” *ArXiv*, vol. abs/1706.03825, 2017. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:11695878>.
- [19] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh y D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” *International Journal of Computer Vision*, vol. 128, pp. 336-359, 2016. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:15019293>.
- [20] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva y A. Torralba, “Learning Deep Features for Discriminative Localization,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2921-2929, 2015. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:6789015>.
- [21] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt y B. Kim, “Sanity Checks for Saliency Maps,” en *Neural Information Processing Systems*, 2018. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:52938797>.
- [22] F. Baldassarre y H. Azizpour, “Explainability Techniques for Graph Convolutional Networks,” *ArXiv*, vol. abs/1905.13686, 2019. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:173188615>.

- [23] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin y H. Hoffmann, “Explainability Methods for Graph Convolutional Neural Networks,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10764-10773, 2019. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:198904065>.
- [24] C. Agarwal, O. Queen, H. Lakkaraju y M. Zitnik, “Evaluating explainability for graph neural networks,” *Scientific Data*, vol. 10, 2022. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:251710449>.
- [25] O. Llorente et al., “Evaluating Neighbor Explainability for Graph Neural Networks,” en *Explainable Artificial Intelligence*, L. Longo, S. Lapuschkin y C. Seifert, eds., Cham: Springer Nature Switzerland, 2024, pp. 383-402.
- [26] *Ericsson Uplink Interference Optimizer rApp - Ericsson*. [En línea]. Disponible en: <https://www.ericsson.com/en/portfolio/cross-portfolio/rapp-directory/network-optimization/ericsson-uplink-interference-optimizer-rapp> (Acceso: 07-09-2024).
- [27] *Ericsson Uplink Anomaly Detector rApp - Ericsson*. [En línea]. Disponible en: <https://www.ericsson.com/en/portfolio/cross-portfolio/rapp-directory/network-optimization/ericsson-uplink-anomaly-detector-rapp> (Acceso: 07-09-2024).
- [28] *Graph Neural Networks for optimized networks — ericsson.com*, <https://www.ericsson.com/en/blog/2023/11/pioneering-within-graph-rural-networks-for-increased-optimization-of-networks>, [Accessed 30-08-2024].
- [29] O. Llorente, G. V. Buenestado, R. Fawzy, A. Falci y C. R. Martin, *Optimising values of discrete input parameters*, EP 24382662.5, June 2024.
- [30] D. P. Kingma y J. Ba, “Adam: A Method for Stochastic Optimization,” *CoRR*, vol. abs/1412.6980, 2014. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:6628106>.
- [31] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio y C.-J. Hsieh, “Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks,” *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:159042192>.
- [32] T. Xie y J. C. Grossman, “Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties.,” *Physical review letters*, vol. 120 14, p. 145301, 2017. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:13838309>.
- [33] S. Brody, U. Alon y E. Yahav, “How Attentive are Graph Attention Networks?” En *International Conference on Learning Representations*, 2022. [En línea]. Disponible en: <https://openreview.net/forum?id=F72ximsx7C1>.

- [34] N. S. Detlefsen et al., “TorchMetrics - Measuring Reproducibility in PyTorch,” *Journal of Open Source Software*, vol. 7, n.º 70, p. 4101, 2022. doi: [10.21105/joss.04101](https://doi.org/10.21105/joss.04101). [En línea]. Disponible en: <https://doi.org/10.21105/joss.04101>.