UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de Fin de Master en Ingeniería Informática

DISEÑO Y MEJORA DE UN PREDICTOR DE TRAFICO BASADO EN GNN

MANUEL FAUVELL MONFORT

Dirigido por: ROCÍO MUÑOZ MANSILLA

Curso: 2022-2023: 1ª Convocatoria

DISEÑO Y MEJORA DE UN PREDICTOR DE TRAFICO BASADO EN GNN

Proyecto de Fin de Master

Realizado por: Manuel Fauvell Monfort

Dirigido por: Rocío Muñoz Mansilla

Tribunal calificador

Presidente: D/D^{a}.

Secretario: D/D^{a}.

Vocal: D/D^{a}.

Fecha de lectura y defensa:

Calificación:

Agradecimientos

Para Eva, que ha soportado mi ausencia y mal humor durante una decada sin quejarse (apenas).

Per a ma mare, ella sempre va creure en mi, encara que jo ho ficara tan difícil i ara per fi està tranquila.

Resumen

En este trabajo se realiza un estudio sobre el efecto de las formas de construcción de grafos y los efectos de las diferentes características disponibles.

La previsión de tráfico es un problema cada vez más importante para la gestión y ordenación de las grandes ciudades. Durante los últimos cincuenta años han ido evolucionando los métodos y las técnicas utilizadas para la tarea de realizar previsiones de tráfico. Desde las primera técnicas puramente estadísticas hasta las actuales basadas en el aprendizaje profundo se ha recorrido un largo camino.

Las *Graph Neural Networks* son la última técnica de moda que se ha aplicado a este problema, siendo un campo que está en rápido y continuo desarrollo. Siendo uno de los principales campos de expansión dentro del aprendizaje profundo.

En este trabajo hemos realizado un estudio que ha estudiado el efecto de la forma de construcción de los grafos necesarios y la influencia de las diversas características adicionales al tráfico que se pueden añadir a los nodos. Presentando un conjunto de ellas y comparando su influencia entre ellas.

Palabras clave

Tráfico

Madrid

Aprendizaje profundo

GNN

Grafo

python

pytorch

DGL

Abstract

In this work, a study is carried out on the effect of the forms of graph construction and the effects of the different available characteristics.

Traffic forecasting is an increasingly important problem for the management and planning of large cities. Over the last fifty years, the methods and techniques used for the task of traffic forecasting have evolved. From the first purely statistical techniques to the current ones based on deep learning, a long way has come.

Graph Neural Networks are the latest fashionable technique that has been applied to this problem, being a field that is in rapid and continuous development. Being one of the main fields of expansion within deep learning.

In this work we have carried out a study that has studied the effect of the way the necessary graphs are constructed and the influence of the various additional traffic characteristics that can be added to the nodes. Presenting a set of them and comparing their influence among them.

Keywords

Traffic

Madrid

Deep Learning

GNN

Graph

python

pytorch

DGL

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general y objetivos

Para la elaboración de este documento se han seguido las pautas marcadas por el reglamento de Trabajos de Fin de Máster de la Universidad Nacional de Educación a Distancia. Se realiza un bitácora de abordo de los pasos preliminares, la metodología utilizada, el ciclo de vida, el diseño y el proceso de implementación del proyecto resultante: *Diseño y mejor de un predictor de tráfico basado en GNN*.

Este documento supone un nexo de unión a todos los documentos básicos del proyecto objeto, conteniendo toda referencia necesaria para entenderlo, aportando también conocimiento sobre cómo se ha logrado su total desarrollo.

Además se informará del porqué, la motivación, las elecciones realizadas durante su concepción y desarrollo y el objetivo final a cubrir.

1.1. Motivación y objetivos

La predicción del tráfico para ajustar los recursos destinados a su gestión y evitar puntos negros y momentos de excesivo tráfico, es una herramienta fundamental en la gestión de las grandes ciudades y Madrid no podía ser diferente.

Madrid dispone de 4000 puntos de medida de tráfico con 7360 sensores que toman diferentes informaciones. Además también dispone de 29 puntos de control meteorológico distribuidos por toda la ciudad. Con esta infraestructura la ciudad está en condiciones de hacer predicciones a futuro en el corto espacio de tiempo para mejorar la gestión de la ciudad[1].

[1] https://acortar.link/Ruzos7

Para aprovechar los datos recopilados y ser capaz de utilizarlos es necesario disponer de las herramientas informáticas adecuadas para ello.

1.1.1. Objetivos

El objetivo de este Trabajo Fin de Máster (TFM) es la realización, continuando en cierta forma con el TFM de la estudiante Elena Sobrini *Diseño y comparación de modelos para la predicción del flujo de tráfico*, de un estudio sobre las diferentes formas de construir grafos de predicción y el efecto de las características utilizadas para la predicción usando un Red neuronal recurrente convolucional de difusión en grafos (GNN). Se pueden definir los objetivos concretos de este TFM de la siguiente forma:

- Generar el software necesario para entrenar modelos GNN y poder comparar estos entre sí.
- Explorar la creación de los grafos necesarios para la predicción y comparar entre las diferentes formas de hacerlo.
- Comparar la efectividad de los diferentes tipos de características que se pueden usar para la predicciones.
- Utilizar herramientas adecuadas y modernas, como pytorch[2] o CUDA[3].

1.1.2. Motivación

Desde hace algunos años la presencia de las diferentes aplicaciones de la herramientas de inteligencia artificial (IA) ha aumentado significativamente. Eso ha aumentado mi interés sobre ellas y las posibles aplicaciones sobre mi vida laboral. Así que cuando surgió la oportunidad de realizar algo mediante predictores con grafos usando una red neuronal recurrente convolucional no lo dudé. Las dos herramientas tienen aplicaciones en la activad industrial y logística donde se desarrolla mi actual vida profesional.

Por otro lado continuar un trabajo tan bueno como el de mi compañera era un reto y un aliciente a su realización.

[2]https://pytorch.org/
[3]https://es.wikipedia.org/wiki/CUDA

1.2. Estructura de la memoria

La memoria de esta proyecto se estructura en los siguientes capítulos:

1. **Introducción general y objetivos.** En esta sección se describe someramente el problema a resolver junto a los objetivos principales y la motivación que lleva a afrontar este TFM. Además también se describen someramente todas la partes de la memoria del trabajo realizado.

2. **Fundamentos teóricos y estado del arte.** Se describe un resumen de la teoría del modelo GNN haciendo hincapié en sus características, puntos fuertes y debilidades. Además se hace un estudio de la situación actual del trabajo en el campo y se especula por el futuro de dicha tecnología.

3. **Análisis y origen de los datos.** En este capítulo se realiza un análisis de los datos de origen, desde donde se han obtenido y sus problemas asociados.

4. **Metodología para el entrenamiento y evaluación del modelo.** Se realiza una presentación de cómo se ha realizado el entrenamiento y su evaluación.

5. **Implementación del modelo e infraestructura utilizada** Se explica cómo se ha implantado el modelo y las diferentes herramientas que se han usado para ello.

6. **Análisis de los resultados.** Se realiza un análisis de los resultados de los diferentes entrenamientos realizados.

7. **Planificación y coste.** En este capítulo se describirá el ciclo de vida utilizado, la planificación de las tareas a realizar junto a un estudio detallado del coste de la aplicación generada.

8. **Conclusiones y trabajos futuros.** Se expondrán las conclusiones derivadas de la realización del trabajo, tanto a nivel puramente técnico como conceptual. Se darán una serie de pautas sobre posibles trabajos futuros para aumentar o complementar la aplicación realizada.

9. **Anexo A: Estructura de la BBDD.** Breve reseña de la estructura de la BBDD.

10. **Anexo B: Manual de instalación.** Manual de instalación y puesta en marcha de todos los servicios necesarios..

11. **Anexo C: Contenido de la entrega.** Relación de los elementos adjuntados en el fichero de entrega.

Capítulo 2

Fundamentos teóricos y estado del arte

En esta sección se presenta un breve fundamento teórico del modelo GNN junto a un Estado del Arte sobre el tema igualmente breve.

2.1. Fundamentos teóricos

Este TFM se basa y continúa el que realizó la estudiante Elena Sobrini [Sobrini, 2022] por lo tanto no hay variaciones respecto al fundamento teórico que ella propuso en su memoria. Lo reproduzco tal cual en esta sección matizando algún dato.

2.1.1. Planteamiento y notación

El problema que nos planteamos consiste en predecir los niveles de tráfico en un conjunto de sensores V, con $|V| = N$, en la ciudad de Madrid, en los T períodos de 15 minutos posteriores a un instante t. Si P es el número de variables (diferentes al tráfico) en cada punto (por ejemplo, variables que indiquen si es un día laborable, que contengan información meteorológica, que indiquen la hora del día, etc), entonces definimos $X \in RN \times P$ como la matriz de variables explicativas. Por otra parte, definimos $Y \in RN$ como el vector con los valores de tráfico. Sean $X(t)$ e $Y(t)$ los valores observados en el momento t.

El objetivo del modelo al ser un modelo dependiente de los instantes anteriores o modelo de predicción de tráfico a corto plazo consiste en aprender una función $h(\cdot)$:

$$\left[X^{(t-T'+1)}, ..., X^{(t)}, Y^{(t-T'+1)}, ..., Y^{(t)}\right] \xrightarrow{h(\cdot)} \left[\hat{Y}^{(t+1)}, ..., \hat{Y}^{(t+T)}\right] \tag{2.1}$$

donde $\hat{Y}^{(t+1)}, ..., \hat{Y}^{(t+T)}$ estén lo más cerca posible de $Y^{(t+1)}, ..., Y^{(t+T)}$. Es decir, el modelo aprende a predecir el tráfico en un momento concreto, dadas todas las otras variables y el tráfico en los T' instantes anteriores.

2.1.2. Planteamiento del problema en un grafo

Las redes neuronales de grafos o *Graph Neural Networks (GNN)* se propusieron en 2009 en [Scarselli et al., 2009]), con el objetivo de extender las redes neuronales clásicas, de manera que utilicen la información estructural de los grafos (nodos y aristas).

Las redes neuronales de grafos pueden resolver dos tipos de tareas:

1. **Tareas centradas en el grafo**: Este tipo de modelos son clasificadores o regresores que se aplican sobre un grafo entero.

2. **Tareas centradas en los nodos**: Estos modelos son clasificadores o regresores de nodos y, por lo tanto, dependen de las propiedades y aristas de cada nodo en particular.

Nuestro problema es una tarea centrada en nodos, ya que queremos predecir el tráfico para cada nodo en la red por separado.

Utilizando la notación de [Li et al., 2017], representamos la red de sensores mediante un grafo dirigido $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, donde $\mathcal{V}$ es el conjunto de nodos, $|\mathcal{V}| = N$, $\mathcal{E}$ es el conjunto de aristas y $\mathbf{W} \in \mathbb{R}^{N \times N}$ es la matriz de adyacencia con pesos, que representa la proximidad entre los nodos.

Continuando con la notación de 2.1.1, sea $Y^{(t)} \in \mathbb{R}^N$ el tráfico observado en el instante t en la red de sensores $\mathcal{G}$, y sea $X^{(t)} \in \mathbb{R}^{N \times P}$ la matriz de variables diferentes al tráfico en el instante t. Entonces el objetivo consiste en aprender una función $h(\cdot)$ como en la ecuación 2.1.

2.1.3. Red neuronal recurrente con proceso de convolución de difusión

El método que utilizaremos estará basado en el descrito en [Li et al., 2017]. La implementación que utilizamos es una modificación de [Chen], que utiliza la librería DGL[1] y Pytorch[2].

[1] https://www.dgl.ai/
[2] https://pytorch.org/

El artículo también tiene una implementación original, pero no es la que utilizó la estudiante Elena Sobrini en [Sobrini, 2022], por lo tanto tampoco es la que se ha utilizado en este trabajo.

Este método modela la dependencia espacial y la dependencia temporal de dos maneras diferentes:

2.1.3.1. Dependencia espacial

El método modela la dependencia espacial relacionando el flujo del tráfico con un proceso de difusión [Ibe, 2013], un tipo de proceso de Markov. Este proceso de difusión se caracteriza por un camino aleatorio en la red $\mathcal{G}$ con matriz de transición $D_O^{-1}\mathbf{W}$, donde $D_O = diag(\mathbf{W1})$ es la matriz de grado *hacia fuera* o de *outdegree* y $1 \in \mathbb{R}^N$ es el vector unidad. Después de muchos pasos, un proceso de Markov de tales características converge a una distribución estacionaria $P \in \mathbb{R}^{N\times N}$, donde la fila $i, P_i \in \mathbb{R}^N$ representa la probabilidad de difusión desde el nodo $v_i \in \mathcal{V}$ o, dicho de otra manera, la proximidad desde el nodo v_i hacia todos los demás nodos.

En lugar de calcular la solución estacionaria, el autor propone una operación de difusión de convolución sobre las características extraídas del grafo, $X \in \mathbb{R}^{N\times P}$, y un filtro f_θ:

$$X_{:,p} \star_{\mathcal{G}} f_\theta = \sum_{k=0}^{K-1} \left(\theta_{k,1}\left(D_O^{-1}\mathbf{W}\right)^k + \theta_{k,2}\left(D_I^{-1}\mathbf{W}^T\right)^k\right) X_{:,p} \tag{2.2}$$

donde $\theta \in \mathbb{R}^{K\times 2}$ son los parámetros para el filtro y $D_O^{-1}\mathbf{W}$ y $D_I^{-1}\mathbf{W}^T$ son las matrices de transición del proceso de difusión y el opuesto.

Una vez definida la operación de convolución en la ecuación 2.2, podemos construir una capa convolucional que transforme P características en vectores de dimensión Q. Sea $\Theta \in \mathbb{R}^{Q\times P\times K\times 2} = [\theta]_{q,p}$, donde $\Theta_{q,p,:,:} \in \mathbb{R}^{K\times 2}$ parametriza el filtro convolucional para la característica de entrada p y de salida q. Entonces, la capa convolucional se pueda escribir como:

$$\mathcal{H}_{:,q} = a\left(\sum_{p=1}^{P} X_{:,p} \star_{\mathcal{G}} f_{\theta_{q,p,:,:}}\right) \tag{2.3}$$

donde $X \int \mathbb{R}^{N\times P}$ son las variables de entrada, $\mathcal{H} \in \mathbb{R}^{N\times Q}$, las de salida, $\theta_{q,p,:,:}$ son los filtros y a es la función de activación

2.1.3.2. Dependencia temporal

El método modela la dependencia temporal con redes *neuronales recurrentes (RNN)*; en concreto, usa *unidades recurrentes cerradas (GRU)* [Cho et al., 2014] . Estas unidades GRU utilizan normalmente multiplicación matricial, la cual sustituiremos por la convolución de difusión de la ecuación 2.2. El autor del método llama por consiguiente a este tipo de unidades: *unidades recurrentes cerradas con convolución de difusión (DCGRU).*

$$\begin{aligned} r^{(t)} &= \sigma\left(\Theta_r \star_{\mathcal{G}} \left[X^{(t)}, H^{(t-1)}\right] + b_r\right) \\ C^{(t)} &= \tanh\left(\Theta_C \star_{\mathcal{G}} \left[X^{(t)}, \left(r^{(t)} \odot H^{t-1}\right)\right] + b_c\right) \\ u^{(t)} &= \sigma\left(\Theta_u \star_{\mathcal{G}} \left[X^{(t)}, H^{(t-1)}\right] + b_u\right) \\ H^{(t)} &= u^{(t)} \odot H^{(t-1)} + \left(1 - u^{(t)}\right) \odot C^{(t)} \end{aligned} \tag{2.4}$$

donde $X^{(t)}$ y $H^{(t)}$ denotan la entrada y salida en el instante t. Las puerta de reset y de update son $r^{(t)}$ y $u^{(t)}$ respectivamente. $\star_{\mathcal{G}}$ denota la difusión de convolución definida en la ecuación 2.2 y Θ_r , Θ_u , Θ_C son los parámetros para los filtros correspondientes. Igual que una unidad GRU, una unidad DCGRU se puede utilizar para construir capas de redes neuronales y entrenarse con retropropagación a través del tiempo. Para la predicción en múltiples pasos, se utiliza la arquitectura *seq2seq* [Sutskever et al., 2014]. Tanto el *encoder* como el *decoder* son redes neuronales recurrentes con DCGRU. El encoder se entrena con series temporales históricas y utiliza su estado final para inicializar el *decoder*. el *decoder* se entrena utilizando observaciones reales. Sin embargo, a la hora de evaluar, las observaciones reales se sustituyen por las predicciones hechas por el propio modelo. Dado que esta discrepancia puede degradar los resultados del modelo, se ha integrado *scheduled sampling* o *muestreo programado* [Bengio et al., 2014], de manera que el modelo se entrena con una observación verdadera con probabilidad ϵ_i, o con una predicción del modelo, con probabilidad $1 - \epsilon_i$, en la iteración i. A lo largo del proceso de entrenamiento, ϵ_i disminuye gradualmente hasta 0, para permitir que el modelo aprenda la distribución de test.

El modelo se esquematiza en la figura 2.1. El *encoder* se entrena con la series temporales. El estado final del encoder se utiliza para inicializar el *decoder*. El *decoder* hace predicciones a partir de, bien las observaciones reales, o bien la salida del modelo. Fuente: [Li et al., 2017].

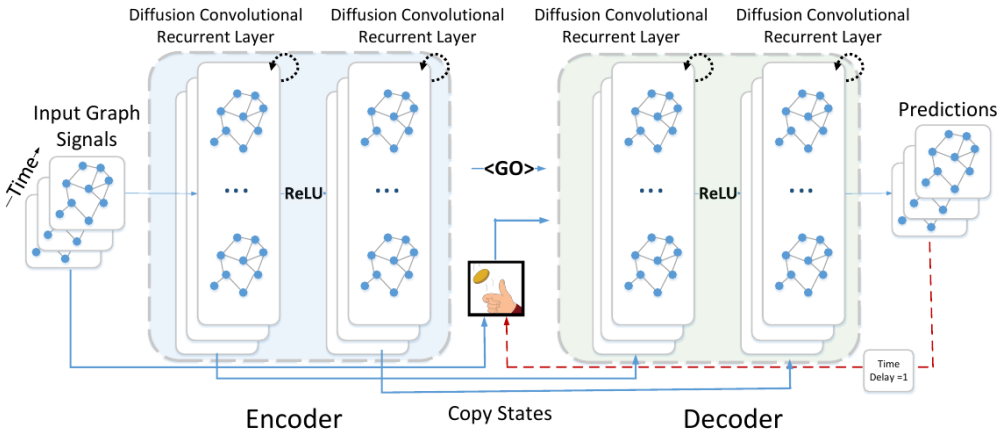

Figura 2.1: Esquema del modelo GNN con convolución de difusión.

2.2. Estado del arte

2.2.1. Previsión del tráfico

Desde el primer momento en que se hizo evidente que el tráfico de las grandes ciudades era un problema se ha intentando descubrir como será su evolución, es decir, predecir el tráfico a más o menos plazo.

A finales de los 70 empezó un creciente interés sobre este tema debido principalmente al aumento de la capacidad de cálculo y el aumento del problema y los costes económicos derivados de la mala gestión del tráfico.

Las herramientas para la previsión del tráfico han ido aumentando y mejorando con el tiempo, en un primer momento se utilizaban métodos estadísticos clásicos para calcular el tráfico en un punto en concreto. Poco a poco esto fue evolucionando hacia métodos no paramétricos (es decir, que no tienen parámetros prefijados) dividiéndose en dos tendencias: la primera se basaba en métodos estadísticos y la segunda en modelos de aprendizaje automático, todavía sin base estadística para sustentarlos.

De los primeros los más populares fueron los modelos autoregresivos integrados de media móvil (ARIMA, [Hyndman and Athanasopoulos, 2021b]). Por ejemplo, [Lee and Fambro, 1999] aplica modelos autoregresivos por subconjuntos y [Voort et al., 1996] introduce un modelo llamado KARIMA, que encadena un clasificador basado en mapas autoorganizados de características [Kohonen, 1982] con un modelo ARIMA (que depende del resultado del clasificador). También

se han evaluado modelos autoregresivos estacionales en [Williams and Hoel, 2003].

Debido a la complejidad de las dependencias del flujo de tráfico, la investigación se ha ido desligando de modelos puramente estadísticos, para concentrarse en modelos de aprendizaje automático.

En [Dong et al., 2018] se utiliza el modelo XGBoost para realizar predicciones a corto plazo sobre el número de coches cada dos minutos. En [Lu et al., 2020] se utilizan procesos de Markov, aprendizaje conjunto o ensemble y métodos basados en regresión y autoregresión para predecir el flujo de tráfico a corto plazo. En [Zhang and Haghani, 2015] se utiliza un modelo de gradient boosting para capturar la dinámica del tráfico y realizar predicciones multipaso.

Multitud de problemas en muchas áreas de investigación se han intentado resolver en los últimos años con redes neuronales y, concretamente, aprendizaje profundo. Esto es debido a su flexibilidad y capacidad de aprendizaje (teóricamente pueden aproximar cualquier función no lineal), así como a que actualmente sí se tiene disponibilidad a la cantidad necesaria de datos para su entrenamiento, algo que antes raramente era posible. Además el aprendizaje profundo, al contrario que los métodos tradicionales de aprendizaje automático, puede teóricamente aprender la jerarquía de las variables de entrada, sin necesidad de realizar transformaciones previas.

En [Liu et al., 2019] se utilizan redes convolucionales para aprender estructuras espacio-temporales y predecir el flujo de tráfico de autobuses. En [Sun et al., 2020] se propone una red profunda llamada TFFNET, que utiliza capas convolucionales para aprender dependencias espaciotemporales en la ciudad de Wuhan y predecir el tráfico.

Con más impacto mediático, en el artículo de prensa [Fisher, 2020] se cuenta como Google colabora con la empresa DeepMind para mejorar las estimaciones de estimación de tiempo de llegada, basándose en redes neuronales de grafos [Lange and Perez, 2020]. En [Tampubolona and Hsiung, 2018] se utiliza una red neuronal completamente conectada, entrenada con técnicas de normalización de batch [Ioffe and Szegedy, 2015] y dropout [Hanson, 1990].

En cuanto las dependencias temporales, en [Osipov et al., 2020] se propone una red neuronal recurrente (RNN) con estructura espiral y que, por lo tanto, puede aprender de manera continua. En [Yang et al., 2019] se utiliza también una red neuronal recurrente, en concreto LSTM, con secuencias largas. En estos casos, la red tiene dificultades para aprender. En el mencionado artículo se utiliza un mecanismo de atención, que da más importancia a las variables explicativas que más influencia tienen para solucionar este problema.

Los enfoques mencionado hasta ahora tienen como objetivo predecir el tráfico a corto plazo, aunque, en ocasiones, los modelos tengan un componente de aprendizaje a largo plazo.

Existen otros modelos cuyo objetivo es la predicción a largo plazo. Es decir, extraer patrones de comportamiento del tráfico, dependientes de otras variables, que nos permitan estimar el tráfico en un instante del futuro, independientemente de la proximidad temporal. Los autores de la publicación [Belhadi et al., 2020] mencionan en ella misma que, según su entendimiento, son los primeros en predecir flujos del tráfico a largo plazo. En ella, utilizan información contextual, como los eventos del día o la información meteorológica, para entrenar varios modelos basados en redes profundas. En [Wang et al., 2021] se menciona el problema que tienen en muchas ocasiones los modelos iterativos con muchos pasos, y es que el error se acumula y amplifica. Por ello, utilizan un modelo *encoder-decoder* con LSTM, llamado *seq2seq* [Sutskever et al., 2014], y que normalmente se utiliza para la traducción automática y otras tareas de procesamiento del lenguaje natural. En [Karim et al., 2019] se utiliza una red neuronal para aprender a predecir el flujo de tráfico a partir de simplemente el mes, el día de la semana, la hora del día y si es festivo.

Otro modelo que también se autodenomina de largo plazo, ya que usa los datos del día anterior es [Zang et al., 2018]. En esta publicación también se menciona un problema recurrente en este tipo de modelos, y es que es frecuente la falta de datos en determinados instantes. El enfoque que siguen es reconstruir los datos que faltan mediante métodos estadísticos y entrenar el modelo con ellos.

Un enfoque más sofisticado a este problema lo presenta [Dong et al., 2022], que encadena la integración de Laplace, para completar los datos que faltan, con una red profunda.

La utilización de GNNs en la predicción de tráfico no ha hecho más que empezar como se puede observar por el creciente número de trabajos sobre este tema, cómo [Sri et al., 2023]. Al mismo tiempo ya es un método suficientemente maduro para que se evalúe su estado [Liu et al., 2023] o se hagan retrospectivas muy completas [Jiang et al., 2023].

2.2.2. Graph Neural Networks

Los grafos son un tipo de dato que modela un grupo de objetos (nodos) y sus relaciones (aristas). Este estructura de datos está recibiendo cada vez más atención por los investigadores trabajando con el aprendizaje automático por su gran expresividad. Se pueden usar para un

gran número de sistemas en varias áreas incluyendo ciencia social, ciencia, bioquímica o, por supuesto, previsión de tráfico.

Como una estructura única no Euclidiana el análisis de grafos se centra en tareas como la clasificación de nodos, predicciones de enlaces y *clustering*. *Graph neural networs (GNN)* son métodos basados en aprendizaje profundo para operar en el dominio de los grafos. GNN se ha convertido en un método de análisis de grafos muy popular.

La primera razón de la existencia de los GNNs se basa en la larga historia de las redes neurales para grafos. En los noventa las Redes Neurales Recursivas se utilizaron en primer lugar en grafos aciclicos [Sperduti and Starita, 1997]. Después de eso las Redes Neurales Recurrentes [Scarselli et al., 2009] y las Redes Neurales Prealimentadas [Micheli, 2009] aparecieron para gestionar los ciclos. Los recientes avances en las redes neuronales profundas, especialmente las Redes Neurales Convolucionadas (CNNs) han resultado en el redescubrimiento de los GNNs. Las CNNs tienen la habilidad de extraer características espaciales multi-escala y con ellas construir representaciones altamente expresivas, que ha conducido a empezar una nueva era de deep learning en todas las áreas del aprendizaje automático [LeCun et al., 2015]. Las principales características de las CNNs son conexiones locales, pesos compartidos y el uso de múltiples capas [LeCun et al., 2015]. Esto es algo de gran importancia en la solución de problemas con grafos. Sin embargo las CNNs solo son operativa en espacios euclidianos como imágenes o textos que pueden representarse como grafos. A pesar de eso es difícil generalizar las CNNs en grafos. Extender los modelos neuronales profundos a dominios no-euclidianos, conocido como aprendizaje profundo geométrico ha sido un campo de investigación emergente.

La otra razón viene desde el aprendizaje de representación de grafos, el cual aprende a representar nodos de grafos, aristas y subgrafos como vectores de baja dimensión. Las aproximaciones tradicionales al aprendizaje automático confiaban en características diseñadas a mano, limitadas por su poca flexibilidad y alto coste. Siguiendo la idea de *aprender la representación* y el éxito de las palabras embebidas [Mikolov et al., 2013], *DeepWalk* [Perozzi et al., 2014], reconocido como el primer método basado en un grafo embebido de aprendizaje de la representación, aplicar el modelo *SkigGram* [Mikolov et al., 2013] en caminos generados aleatoriamente. Estos métodos sufren de dos inconvenientes importantes. Primero, los parámetros no son compartidos entre los nodos, lo cual lleva a ineficiencias ya que le número de parámetro crece linealmente con el número de nodos. Segundo, pierden la capacidad de generalización, lo cual significa que

no pueden lidiar con gráficos dinámicos o generalizar a nuevos grafos.

Basado en CNNs y el embebido de grafos, variante de GNN son propuestas para agregar colectivamente información de la estructura del grafo. Estos pueden modelizar input y output consistente de elementos y sus dependencias.

[Wu et al., 2021] clasifica las GNNs en cuatro grupos: Redes neurales de grafos recurrentes, redes neurales de grafos convolucionales, grafos autocodificados y redes neurales de grafos espacio-temporales. En el caso de este trabajo nos centramos en una red neural convolucional recurrente.

Las GNNs se han convertido por derecho propio, especialmente desde que se conoció que google las utilizaba para cálculos de tiempos [Lange and Perez, 2020] en uno de los principales actores de la investigación en predicción del tráfico.

Hay múltiples ejemplos de su uso para predicción del tráfico como el que nos ocupa en este trabajo [Li et al., 2017]. También con GNNs convolucionales tenemos a [Huang et al., 2019] que introduce el concepto de clasificación de los vecinos, [Andreoletti et al., 2019] y [Bai et al., 2020] con enfoques similares al que ocupa este trabajo y después enfocado a un problema similar como es la demanda de taxis en [Luo et al., 2022].

Capítulo 3

Análisis y origen de los datos

En este capítulo se va a realizar una explicación del origen de los datos que se han utilizado para realizar este TFM. Además se presentará un somero análisis previo y una explicación de su tratamiento además de las herramientas que se van a utilizar para realizar la persistencia y los cálculos necesarios.

3.1. Origen de los datos

En primer lugar se va a explicar que tipo de persistencia de datos y porque se ha optado por ella.

El ayuntamiento de Madrid pone a disposición de la ciudadania una serie de datos abiertos dentro de su portal de transparencia[1]. Estos datos se han complementado con una serie de datos disponibles de forma abierta por múltiples fuentes de la forma que se especifica en los siguientes apartados:

3.1.1. Persistencia de los datos

En este trabajo se ha optado por realizar una persistencia de los datos para mejorar la comodidad de su realización utilizando una BBDD. Después de evaluar las diferentes alternativas en el mercados se ha decidido usa un motor de BBDD NoSQL de alto rendimiento como es MongoDB, las razones de esta elección son las siguientes:

- Consume pocos recursos

[1] https://datos.madrid.es/portal/site/egob

- Tiene imágenes de *docker* prehechas.
- Se lleva muy bien con *python*.
- Una gran documentación.
- Interés profesional por el autor del TFM.

Para su uso se ha utilizado un proyecto de *docker-compose* disponible en el siguiente repositorio git[2].

La estructura de esta BBDD así como las queries utilizadas se pueden consultar en el anexo A

3.1.2. Datos de tráfico

El portal de transparencia del ayuntamiento de Madrid ofrece varias colecciones de datos relacionadas con el tráfico pero en nuestro caso se van a utilizar tres colecciones de ellas:

1. Ubicación de los puntos de medida de tráfico. Disponible aquí[3], donde se dispone de la situación exacta de todos los puntos de medida y sus cambios respecto al tiempo.

   De estos datos se aprovechan el id único de cada punto junto a sus coordenadas.

2. Histórico de datos de tráfico desde 2013. Disponible aquí[4], donde se dispone de un histórico de los datos en intervalos de 15 minutos para todos los puntos de medida a lo largo del tiempo.

   De estos datos se aprovechan para cada medida los datos de intensidad y ocupación que se usarán para calcular la carga ya que se ha decidido no usar la que viene en las series de datos.

3. Datos del tráfico en tiempo real. Disponible aquí[5], donde se dispone de una medida instantánea del tráfico en tiempo real. Estos datos se van a usar para obtener la intensidad máxima de cada uno de los puntos de medida.

[2]https://git.dismental.org/fauvell/mongodb-tfm
[3]https://acortar.link/Ruzos7
[4]https://acortar.link/sneF7a
[5]https://acortar.link/eglxZZ

Respecto a los datos de tráfico, durante el estado de alarma estos datos están tremendamente viciados y no ofrecen prácticamente ninguna información, por lo tanto se toma la decisión de no tenerlos en cuenta. Se carga el periodo y se tratan pero luego no se utilizarán.

3.1.2.1. Tratamiento de los datos

Cada uno de los datos obtenidos se han cargado dentro de una colección en la instancia de mongodb creada para tal fin, para después tratarse para obtener una serie de datos cocinados para luego utilizarlos en los entrenamientos que se van a utilizar para hacer el análisis previsto.

Todo el código necesario puede observarse en el repositorio a tal efecto [Fauvell].

En primer lugar y durante una semana se han recogido cada 15 minutos los datos en tiempo real de la fuente 3. Esos datos junto a los datos cargados para cada punto en la colección de la BBDD desde la fuente 1 se han combinado para obtener la intensidad máxima para cada punto de medida. Aquellos puntos que no se ha podido obtener la intensidad máxima o esta ha cambiado en el transcurso de una semana de toma de datos se ha decidido descartarlos.

Por otro lado se ha decidido cargar las medidas de tráfico del punto 2 desde el año 2019 en adelante ya que los datos meteorológicos solo están disponibles desde ese año. Todas aquellas medidas marcadas como erróneas se han descartado. También se han descartado aquellas medidas donde faltara alguno de los datos que se pretendía tratar, intensidad y/u ocupación, además de aquellos que no son puntos de carácter urbano.

En este punto tenemos la primera diferencia importante con el trabajo de Elena Sobrini [Sobrini, 2022] ya que ella decidió eliminar aquellas medidas que resultaran estacionarias durante una hora. En mi opinión esto no significa que sea un error, los flujos de tráfico tienden a ser bastante estacionarios así que no es raro que durante un periodo de tiempo pueda haber datos iguales de intensidad y ocupación. La explicación para esta estacionalidad es que los semáforos crean una estacionalidad intrínseca y que la ocupación es una medida muy engañosa que solo se puede usar como complemento y no como objetivo.

Después para cada punto, y una vez los puntos meteorológicos se han cargado, se ha añadido para cada punto una lista ordenada por distancia creciente de los 29 puntos meteorológicos. Para obtener esta lista se ha usado una instancia docker local de Open Route Service[6] que durante esta misma memoria explicaremos más en profundidad en el capítulo 5

[6]https://openrouteservice.org/

Como resultado final tenemos dos colecciones dentro de la BBDD, en una se han combinado los datos de más de 4434 puntos de medida y en otra colección se disponen más de 552 millones de medidas horarias correspondientes a estos puntos desde el 1 de enero de 2019 hasta el 31 de mayo de 2023.

3.1.3. Datos meteorológicos

El ayuntamiento de Madrid ofrece dentro de su portal de transparencia datos meteorológicos de las 29 estaciones de la ciudad. En esta memoria se van a utilizar dos de estas colecciones:

1. Estaciones de control. Disponible aquí[7], donde se muestran las ubicaciones a lo largo del tiempo de las estaciones de control y sus capacidades.

2. Datos horarios desde 2019. Disponible aquí[8], donde se muestran las medidas horarias de los valores para los sensores disponibles en cada ubicación.

3.1.3.1. Tratamiento de los datos

En este caso también se han cargado en la BBDD de mongodb está información aunque ha sufrido un poco más de proceso que lo propios puntos de tráfico. En primer lugar se han cargado los datos de los 29 puntos incluyendo un dato básico como es la ubicación.

Por otro lado también se han cargado las medidas horarias que se han transformado en medidas con frecuencia de 15 minutos utilizando una interpolación lineal entre las dos medidas horarias. Allí donde no ha sido posible realizar la interpolación se han desestimado los datos.

Como resultado final se crean dos colecciones una con los datos de cada punto y otra con las medidas en frecuencia de 15 minutos.

3.1.4. Datos temporales

Además de los datos obtenibles por el portal de transparencia del ayuntamiento de Madrid también se han usado una serie de datos de muy diferentes fuentes que se han combinado en características de un calendario:

[7] https://acortar.link/ET8rMz
[8] https://acortar.link/L3lWyh

1. Datos horarios. Año, mes, día, hora, minuto, día de la semana, estación y tipo de día (entre semana, sábado y domingo).

2. Festivos laborales. Se ha buscado por varias fuentes los calendarios laborales para cada año.

3. Festivos escolares. Se han buscado los calendarios escolares de cada año.

4. Partidos jugados como local del Real Madrid y del Atlético de Madrid. Se han obtenido de muy diversas fuentes cuando se han jugado, cuanto han tardado y su aforo. Al final solo se han usado tres características derivadas, pre-partido, partido y post-partido y se ha desestimado el uso de la ubicación y del aforo por ser de difícil tratamiento.

3.1.4.1. Tratamiento de los datos

Todos los datos se han combinado en una sola tabla con fechas con frecuencia cada 15 minutos que se ha utilizado para combinarla con el resto de los datos en la fase final ya de entrenamiento.

3.2. Exploración de los datos

Con los datos cargados en la BBDD se puede realizar un estudio de estos mismos datos para ver que datos vamos a utilizar. En este caso debido al proceso anterior se puede garantizar que las medidas del tráfico siempre poseen las dos características susceptibles de poder usarse: intensidad y ocupación. Por otro lado también se puede garantizar que se tiene el dato de la intensidad máxima que soporta el punto de medida.

Respecto a los datos meteorológicos es bastante más complicado porque hay una gran heterogeneidad y se debe tratar ya cuando se realice la preparación final para dejar la información que se va a usar en los entrenamientos.

3.2.1. Selección de variable objetivo

En este punto hay que decidir cual va a ser la variable objetivo del proceso de entrenamiento. Las posibilidades son tres:

- **Intensidad**. La intensidad es cuantos vehículos pasan por un determinado punto en un determinado momento. Es una medida relativa ya que depende de la capacidad máxima del punto en cuestión. Este característica supone que no sea adecuada para comparar la intensidad entre puntos por lo tanto para realizar un entrenamiento efectivo.

- **Ocupación**. Representa el tiempo que el sensor está ocupado con un vehículo. Otra vez es una medida bastante engañosa ya que la misma ocupación puede representar un continuo paso de vehículos a gran velocidad como un paso más comedido pero a menor velocidad. Esto vuelve a suponer que no es la característica ideal para realizar el entrenamiento.

- **Carga**. El ayuntamiento de Madrid ofrece también una medida entre 0 y 1 que da una idea del nivel de saturación del punto de medida. Es una medida sintética que solo se ofrece en los puntos urbanos. Al ser una medida sintética no parece adecuada aunque sea quizá la mejor sobre el papel. Se calcula de la siguiente forma:

$$C = Y + (1 - Y) \cdot TO \cdot KC \tag{3.1}$$

siendo:

$$Y = \frac{I}{I_S} \tag{3.2}$$

donde:

  - Y Relación entre la intensidad y la intensidad de saturación.
  - I Intensidad en el punto de medida.
  - I_S Intensidad de saturación en el punto de medida.
  - TO Tiempo de ocupación en tanto por uno.
  - KC Coeficiente de carga. Es un coeficiente de regulación que por defecto es uno pero que se puede cambiar para cada punto de medida.

Evidentemente no hay ninguno de los tres que sea ideal, la parte que de más reparos es el cambio en el tiempo del factor de regulación.

Así que he decidido utilizar la carga pero eliminando el coeficiente de carga de la formula, es decir, fijando $KC = 1$ en la formula 3.1 y volviéndola a calcular para cada medida. Usando para ello la intensidad de saturación que hemos obtenido de los datos de tráfico en tiempo real.

Evidentemente estaremos introduciendo la posibilidad de un error si se ha cambiado la intensidad máxima de la vía, pero a efectos de entrenamiento se considera que no tendrá efectos negativos ya que el error siempre será el mismo sea cual sea la medida.

Por otro lado todas aquellas medidas donde $I > I_S$ se desestimaran completamente.

3.2.2. Selección de sensores

Una vez cargados todas las medidas y seleccionada la variable objetivo se puede pasar a seleccionar los sensores que se van a utilizar.

Para ello, como se ha comentado antes se pueden garantizar que todas las medidas incluidas tienen los datos de tráfico necesarios para obtener la carga. Siendo más susceptible de fallo la presencia de datos meteorológicos.

Para sortear los fallos de datos meteorológicos se va a suponer que, al menos, los datos más comunes (precipitación, temperatura, presión, humedad y radiación solar) estarán disponibles para cada punto y medida combinando los datos de las 29 estaciones meteorológicas.

Además se ha decidido ignorar el periodo de estado de alarma para realizar los entrenamientos, por lo tanto tampoco se tiene en cuenta para buscar los puntos más idóneos.

Teniendo esto último en cuenta para seleccionar los mejores puntos llegamos a la conclusión que el máximo teórico de medidas para todo el periodo es 110207 medidas.

Teniendo en cuenta eso es sencillo preparar una consulta en la BBDD que añada un contador para cada punto con las medidas y clasificar estos puntos por el porcentaje de medidas que tiene respecto al máximo. Con este criterio se han clasificado los puntos entre los que tienen el 99 % (verde), 95 % (azul), 90 % (naranja) y el resto (rojo) como se puede ver en el mapa de la figura 3.1 que se puede consultar con más detalle aquí[9].

Una vez estudiados los puntos en el mapa se ha buscado una región que tuviera una densidad de puntos verdes y azules con una buena distribución. Se ha elegido la zona delimitada por: El Paseo de la Castellana - Maria de Molina - Francisco Silvela - Doctor Esquerda - O'Donnell - Paseo de la Castellana donde hay 123 puntos entre verdes, azules y naranjas como se puede ver

[9] https://umap.openstreetmap.fr/es/map/mapa-sin-titulo_946530#12/40.4241/-3.7086

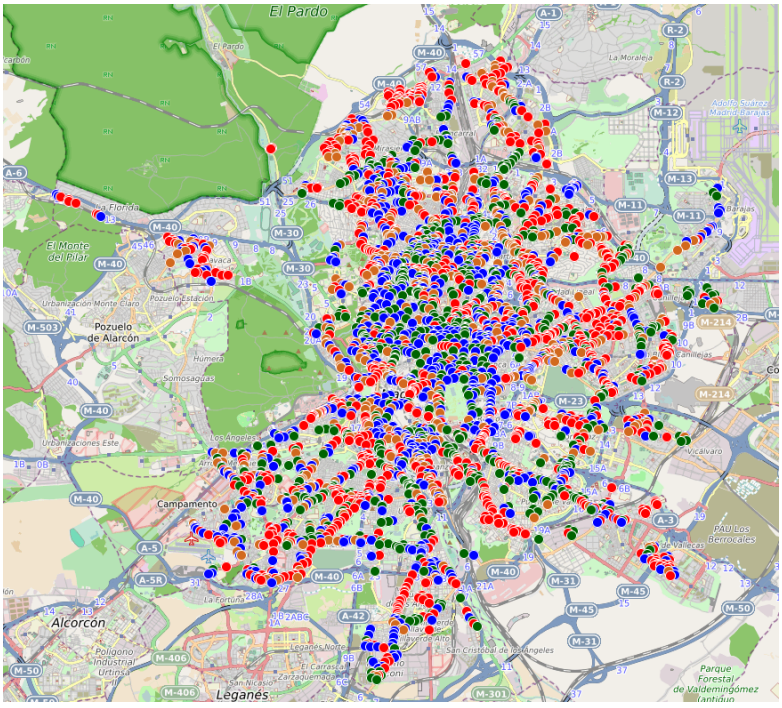

Figura 3.1: Mapa con todos los puntos de medida clasificado por disponibilidad: verde-99 %, azul-95 %, naranja-90 % y rojo el resto.

en la figura 3.2 y se puede ver con más detalle aquí[10].

Los 123 puntos seleccionados eran los sensores objetivo que se iban a utilizar, pero la realidad de la capacidad de computo (se tenía previsto poder disponer de un máquina mucho más potente y adecuada) ha provocado que se utilizaran solo los 49 puntos verdes de la zona acotada que podemos ver su localización en la figura 3.3.

A pesar del revés todavía se puede observar que hay muchos puntos y están bastante bien distribuidos por la zona.

3.2.3. Mezcla y preparación final de los datos

Una vez seleccionados los 123 puntos posible objetivo se construye para cada uno de ellos una colección de medidas donde se combinan todos los datos disponibles y se calcula la carga para cada medida.

En este punto hay una serie de dificultades que hay que comentar:

- Aunque cada punto tenga la mayoría de medidas de tráfico posibles no tienen porque coincidir en todos los puntos. Es decir, si eliminamos una medida para todos los puntos

[10] https://umap.openstreetmap.fr/es/map/mapa-sin-titulo_958774#15/40.4295/-3.6797

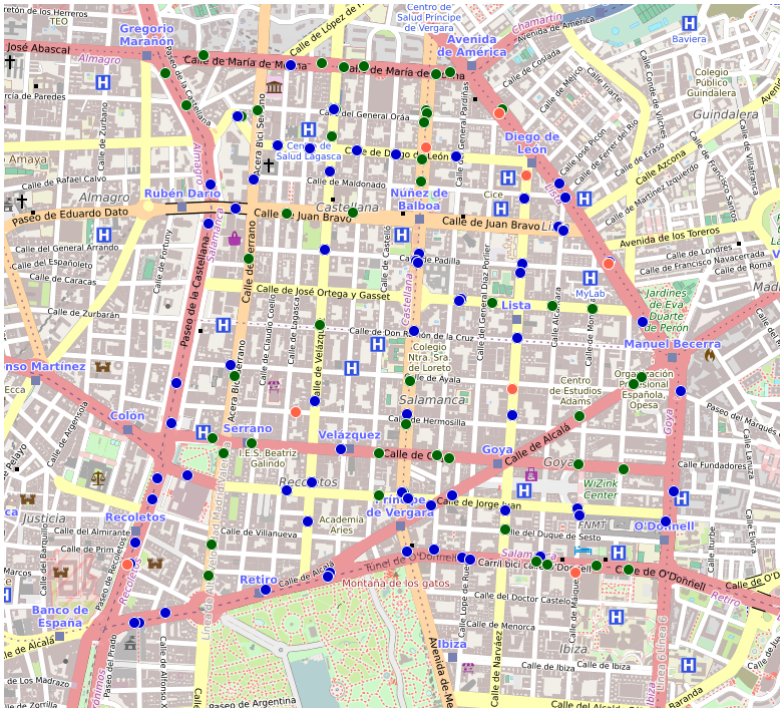

Figura 3.2: Mapa con los puntos de medida seleccionados clasificados por disponibilidad: verde-99 %, azul-95 %, naranja-90 % y rojo el resto.

solo porque uno de los puntos no tenga esa medida se pierden la mayoría de medidas. P.e. si lo hacemos así para los 49 puntos que poseen al menos un 99 % de las medidas perdemos un 40 % de las medidas totales.

Por otro lado tampoco se pueden dejar medidas nulas ya que el modelo que se está evaluando no reacciona bien con los nulos. La solución que se ha tomado es la misma que para ampliar las medidas meteorológicas, es decir, interpolar linealmente las medidas no presentes. Con esta medida consideramos que añadimos cierto error pero es mínimo ya que solo un 1 % de los 5 millones de medidas disponibles son las afectadas.

- En el caso de los datos meteorológicos se hace una mezcla de los 29 puntos de medida ordenados por distancia creciente al sensor en concreto. Si aun así no se han podido obtener las cinco características que se van a tratar, se deja caer esa medida en todos los puntos.

Al final de este proceso se obtiene una colección de momentos para cada punto con todas las características posibles y donde podemos garantizar que no hay nulos para un momento en concreto. P.e. para los 123 puntos seleccionados originalmente obtenemos 13.5 millones de registros para trabajar con ellos.

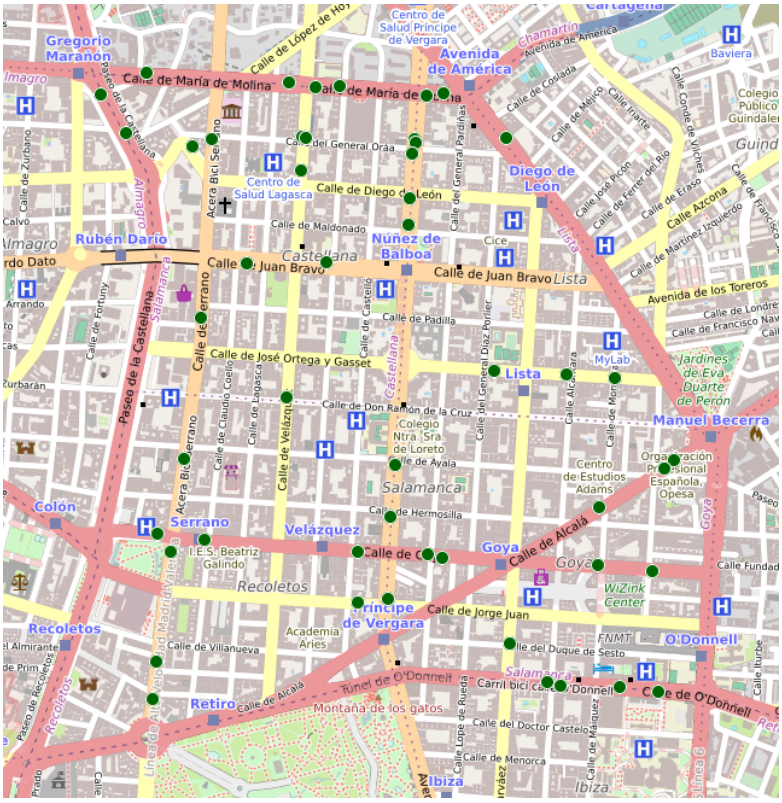

Figura 3.3: Mapa con los puntos finalmente seleccionados

Las medidas que se añaden por medida son las siguientes, aunque algunas no se van a utilizar en los entrenamientos:

- *year*. El año de la medida.
- *month*. El mes de la medida.
- *day*. El día de la medida.
- *hour*. La hora de la medida.
- *minute*. El minuto de la medida (0, 15, 30 y 45).
- *weekday*. El día de la semana.
- *day_type*. El tipo de día de la semana (mon-fri, sat y sun)
- *season*. La estación (spring, summer, autum y winter)
- *prematch*. Si hay un partido de fútbol en menos de hora y media.

- *match*. Si se está jugando un partido de fútbol.
- *postmatch*. Si han pasado menos de 90 minutos desde que ha terminado un partido de fútbol.
- *audience*. La audiencia del partido (al final no se ha usado).
- *place*. El estadio del partido (al final no se ha usado)
- *bank_holiday*. Festivo laboral.
- *work_office_day*. Dia laboral.
- *schoo_day*. Día lectivo.
- *school_holiday*. Dia no lectivo.
- *state_of_alarm*. Si estaba declarado el estado de alarma, al final es irrelevante ya que solo tenemos días sin estado de alarma.
- *intensity*. La intensidad de tráfico.
- *occupation*. La ocupación del sensor.
- *load*. La carga en el sensor.
- *temperature*. La temperatura ambiente.
- *humidity*. La humedad relativa.
- *pressure*. La pressión barométrica.
- *solar_radiation*. La radiación solar.
- *rain*. El nivel de precipitación.

Capítulo 4

Metodología para el entrenamiento y evaluación del modelo.

En esta sección se presenta la metodología utilizada para realizar el entrenamiento de cada modelo y su evaluación.

4.1. Datos de entrenamiento y de test

Los datos que se van a utilizar son los que se han preparado, con las restricciones ya presentadas, en el capítulo 3.1.

El caso presentado en este TFM es muy similar al que tenía Elena Sobrini [Sobrini, 2022] ya que tenemos un número similar de nodos, en su trabajo eran 37 y en este son 49 y un número similar de medidas. En su caso eran tres años incluyendo la pandemia y en este caso es algo más de tres años y medio excluyendo los meses del estado de alarma.

Al igual que en el trabajo de Sobrini, este se basa en entrenamiento con datos a lo largo del tiempo y por lo tanto es importante que los datos de entrenamiento sean anteriores a los datos de test. La diferencia con Sobrini es que en este trabajo no se ha ha hecho una división 80x20 como es habitual sino que al final han quedado para entrenamiento el 75 % de los dato se han reservado para entrenamiento, es decir, hasta el día 31/07/2022 (excluyendo los meses de estado de alarma), por lo tanto se han reservado para test el resto de los datos desde el día 1/08/2022 hasta el día 31/05/2023.

La razón de este cambio de porcentajes es disponer de 10 meses para poder testear los

modelos.

4.2. Validación

Para validar las diferentes configuraciones que se vayan a probar, haciendo hincapié especialmente en poder comparar estas configuraciones entre sí se ha optado por realizar una validación cruzada de tipo temporal. Exactamente igual que la usada en el trabajo de Sobrini [Sobrini, 2022]

4.2.1. Ajuste de parámetros

No se ha realizado ningún ajuste de parámetros ya que no es el objetivo principal la optimización del modelo sino, más bien, ver la influencia de las características y, especialmente, la forma de construir el grafo. Se ha confiado en el trabajo de la compañera, excepto con el aumento del tamaño de *batch* ya que en este trabajo se está usando CUDA y por lo tanto nos podemos permitir un mayor batch, por lo tanto se han se han utilizado los siguientes parámetros:

1. Pasos de difusión: 2
2. Tasa de aprendizaje inicial: 0.01
3. Tamaño de *batch*: 192
4. Tasa de aprendizaje mínima: 2e-6
5. Norma máxima de los gradientes: 5
6. Dirección: *both*

Evidentemente tampoco se ha utilizado el peso mínimo para considerar un arista ya que es uno de los objetivos del estudio comparar entre diversos grafos.

4.2.2. Validación cruzada

Al igual que mi compañera he usado un método de validación cruzada temporal basado en el presentado por [Hyndman and Athanasopoulos, 2021a]. Para las observaciones de test se toma

un solo mes y la de entrenamiento son únicamente las medidas anteriores a ese mes como se puede ver en la figura 4.1.

De esta forma garantizamos que las medidas futuras no interfieren en la validación.

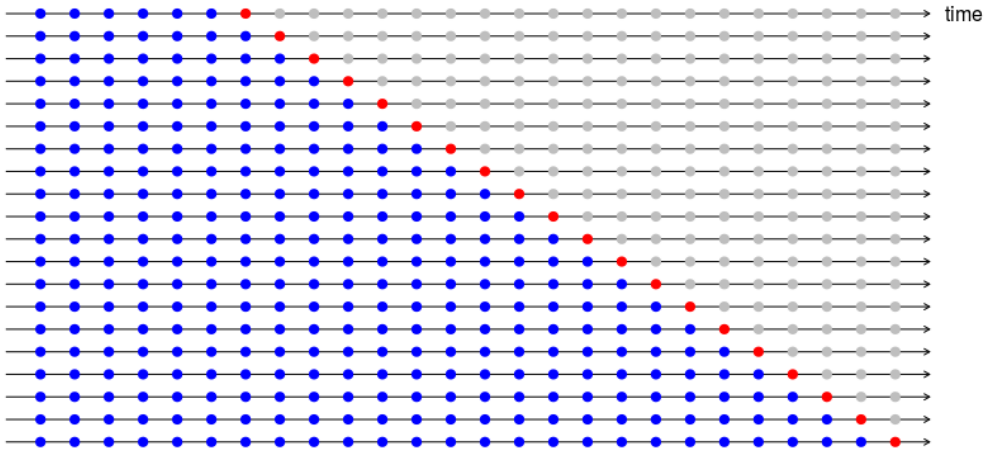

Figura 4.1: El diagrama ilustra las series de conjuntos de entrenamiento y de test. Los puntos azules son observaciones de entrenamiento y, los naranjas, observaciones de test. En nuestro método, cada punto se compone de todas las observaciones de un mes. [Hyndman and Athanasopoulos, 2021a]

4.2.3. Evaluación de modelos

Para la evaluación de los modelos se utilizará la validación cruzada descrita. Para cada validación se va a calcular el error absoluto medio y el error cuadrático medio. Como hemos reservado 10 meses como datos de test se dispondrá del mismo número de medidas para obtener una media y poder comparar entre configuraciones, es decir evaluaremos la media de estas puntuaciones en todos los sensores del modelo por configuración.

Capítulo 5

Implementación y aplicación del modelo

En este capítulo se va a hablar de cómo se ha realizado la Implementación y la aplicación del modelo. Asimismo también se recorren las diferentes configuraciones utilizadas y se comentan las dificultades encontradas.

5.1. Implementación y herramientas

Para realizar la implementación se han usado las herramientas habituales en estos desarrollos utilizando el lenguaje *python*.

5.1.1. Hardware

El hardware utilizado para realizar los entrenamientos ha sido el siguiente:

- Procesador: Intel Pentium I7-6700K
- R.A.M.: 48 GB
- GPU: Nvidia GTX-1070 8GB C.U.D.A capable

5.1.2. Software

Todos los entrenamientos se han realizado en un sistema con una instalación estable de Debian GNU/linux con los paquetes de los repositorios.

5.1.3. Librerías

Las principales librerías utilizadas de *python* son las siguientes:

- *PyTorch*. Esta es una de las librerías de referencia para el entrenamiento de redes neuronales con *python*. Soporta múltiples configuraciones y tiene soporte para utilizar C.U.D.A de forma nativa. Su página oficial incluye mucha documentación para su uso, disponible aquí[1].
- *DGL Deep Graph Library* es un desarrollo que comenzó en 2018 para facilitar las implementaciones de redes neuronales con grafos. En este tiempo se ha convertido en referencia y tiene un software cuidado y maduro. Disponible aquí[2].
- *Pandas*. Esta librería no necesita presentación, es posiblemente una de las librerías más populares cuando hablamos de IA y aprendizaje profundo. Facilita el tratamiento de datos masivos. Disponible aquí[3].
- *Numpy*. Otra librería de referencia usada ubicuamente en multitud de proyectos. Acerca la potencia numérica de C y Fortran. Disponible aquí[4].
- *PyMongo*. Es el conector recomendado para conectar BBDD *MongoDB* en *python*.

5.1.4. Otras herramientas

Además de las librerías se han utilizado otras herramientas para la realización del trabajo:

- *MongoDB*. Como ya se ha visto en la sección 3.1.1 se ha usado este motor de BBDD para implementar la persistencia en el trabajo.
- *OpenRouteService*. Como ya hemos comentado anteriormente se ha usado este software para calcular distancias entre localizaciones. Es un software muy potente que se puede instalar localmente para mejorar el rendimiento usando *docker*. Ver aquí[5]. Para los mapas se han usado los que provee OpenSreetMap[6]

[1] https://pytorch.org/
[2] https://www.dgl.ai/
[3] https://pandas.pydata.org/
[4] https://numpy.org/
[5] https://giscience.github.io/openrouteservice/installation/Running-with-Docker.html
[6] https://www.openstreetmap.org/about

- *Docker*. Se ha utilizado este software de contenedores ligeros para implantar el software auxiliar necesario. Disponible aquí[7]

5.2. Construcción de grafos

Para la construcción de los grafos se ha usado la librería DGL como se ha comentado antes, para obtener las distancias entre los nodos se ha usado el servicio de *OpenRouteService*. Este servicio tiene en cuenta la dirección de las ubicaciones, por lo tanto podemos confiar en que construirá nuestros grafos dirigidos correctamente.

Los grafos los podemos definir así:

- Un nodo lo denotamos como $v_i \in \mathcal{V}$ siendo $\mathcal{V}$ el conjunto de todos los nodos.
- Para cada par de nodos $v_i, v_j \in \mathcal{V}$ donde $i \neq j$ calculamos la distancia en segundos en coche sin tener en cuenta el tráfico que denotamos como $d_{i,j}$. Para calcular esta distancia utilizamos una de las estrategias definidas en esta misma sección posteriormente.
- Calculamos la desviación estándar de todas las distancias para luego calcular los pesos con un kernel Gaussiano truncado descrito en [Shuman et al., 2013]. Al peso de la arista de v_i a v_j lo denotamos como $w_{i,j}$ y lo calculamos con la siguiente formula:

$$w_{i,j} = \exp\left(\frac{-d_{i,j}^2}{\sigma_{distancias}^2}\right)$$

- Se puede disminuir la complejidad del grafo se pueden eliminar los pesos que cumplan $w_{i,j} < k$ para algun $k > 0$.
- Por último se utiliza la librería DGL para construir finalmente el grafo.

Para construir las distancias se han usado dos estrategias:

- La primera es la estrategia habitual donde se construye una matriz de distancias entre cada nodo con todos los otros nodos. Esta es la que se usa la mayoría de las ocasiones y la que usó mi compañera. En este caso *OpenRouteService* tiene una herramienta que lo hace en una solo consulta por punto. Aquí[8] se puede consultar la documentación.

[7] https://www.docker.com/
[8] https://openrouteservice.org/dev/#/api-docs/v2/matrix/profile/post

- La segunda es una estrategia un poco más elaborada donde se pretende capturar las relaciones de dependencia entre puntos. La idea es eliminar aquellas aristas que deban pasar por otro nodo para conseguir el mejor tiempo. Solo se pueden añadir opcionalmente aquellas aristas entre nodos que tengan un camino alternativo casi tan bueno como el original controlando como de bueno es este camino dependiendo de un limite pasado como parámetro.

  Para ello se ha seguido un algoritmo iterativo muy simple:

```
aristas
para_cada_nodo:
    matriz_distancias[nodo] := calculamos_matriz_distancias_otros_nodos
para_cada nodo como actual_nodo:
   while (no_esten_tratados_todos_los_nodos):
       nodo_mas_cercano := get_nodo_menor_distancia_matriz()
       add_nodo_distancia_to_aristas(nodo_mas_cercano)
       marcar_procesado(matriz_distancias[actual_nodo][nodo_mas_cercano])
       para cada otro nodo como aditional_nodo:
           si matriz_distancias[actual_nodo][aditional_nodo] igual matriz_distancias[
               actual_nodo][nodo_mas_cercano] + matriz_distancias[nodo_mas_cercano][
               aditional_nodo]:
               distancia_sin_pasar_nodo_mas_cercano =
                   calcular_distancia_entre_nodos_sin_pasar_por_otro(actual_nodo,
                   aditional_nodo, nodo_mas_cercano)
               si distancia_sin_pasar_nodo_mas_cercano <= matriz_distancias[actual_nodo
                   ][aditional_nodo] + limite_adicional:
                   add_nodo_distancia_to_aristas(aditional_nodo)
               marcar_procesado(matriz_distancias[actual_nodo][aditional_nodo])
```

  Al final se obtiene una matriz de distancias donde tenemos aristas que reflejan dependencia directa o secundaría según nuestro limite.

5.3. Implementación del modelo

Al igual que en el trabajo de Elena Sobrini [Sobrini, 2022] no utilizamos la implementación original del artículo original sino la variación de Sabrini que se diferencia en estos aspectos:

- El modelo original se usa para predecir la velocidad media en los nodos de la red, mientras que en este trabajo, por las razones mencionadas en la sección 3.2.1, se está usando para predecir la carga.

- Los datos del modelo original están agregados en períodos de cinco minutos, y, en este trabajo, en períodos de 15 minutos.
- Las dos redes que usa modelo original tienen un mayor número de nodos (207 y 325), al contrario de los 49 que se han seleccionado en la sección 3.2.2.
- Los dos conjuntos de datos que utiliza el modelo original se extienden en períodos de 4 y 6 meses, respectivamente, mientras que este utiliza datos de cuatro años.
- Al contrario que en el modelo original, muchos de los sensores de la red tienen períodos largos en los que no hay disponibilidad de datos, obligando a interpolar datos donde ha sido necesario.
- El modelo original realiza predicciones a 15, 30 y 60 minutos. En este trabajo se van a evaluar períodos más largos de tiempo, de 4 horas (16 períodos de 15 minutos).
- Por último, en este trabajo se ha evaluado el uso de otras variables diferentes al tráfico, mientras que el modelo original solo utiliza como características de los nodos las propias variables de tráfico.

Por estas razones, ha sido preciso adaptar el modelo en los siguientes aspectos:

- Para que la red neuronal use también características de los nodos.
- Se ha corregido la función que calculaba el error para adaptarla a este caso.
- La implementación no utiliza las fórmulas originales de una red recurrente cerrada (GRU), que son, además las que se utilizan en el artículo, ya que intercambia en un caso la función sigmoide y la función de tangente hiperbólica. Esto se ha corregido.
- La implementación original tiene un error en la asignación de grafos a la red neuronal. Este error también se ha tenido que corregir.

5.4. Entrenamiento

Una vez implementado el modelo y preparados los datos se ha pasado a realizar los entrenamientos necesarios para estudiar la influencia de la forma del grafo y de las características en las predicciones. Se pasa a describir acto seguido el proceso y las configuraciones probadas.

5.4.1. Elementos comunes

Como se ha señalado en la sección 4.2.1 hay una serie de parámetros que son comunes a todos los entrenamientos. En primer lugar los parámetros del modelo como se ha señalado, pero no únicamente estos.

También se ha optado para que todos los test se hagan en vista a 16 periodos.

Como se ha indicado en la sección 4.2.2 se van a producir 10 secuencias de validación, para cada una de ellas se van a realizar un máximo de 10 *epochs*, cortando el entrenamiento si se producen uno de ellos peor que los tres anteriores.

La razón de esto es que el proceso de un solo epoch es bastante costoso aun usando C.U.D.A. Cada *epoch* supone entre 4 y 6 minutos dependendiendo del número de medidas de entrenamiento y test. La preparación de los datos supone unos 4 minutos adicionales por lo que una validación entera supone entre 16 y 66 minutos, por tanto la prueba de una configuración supone entre 160 minutos y 660 minutos, esto se agrava por un problema con C.U.D.A. que se expondrá en la sección 5.4.4.

En la fase de pruebas se ha observado que el modelo en general no tiende al sobreajuste u *overfitting*. Es decir, que normalmente la *epoch* en la que el modelo tiene un mejor resultado para entrenamiento, tiene un resultado similar para los datos de test, Por ello, de entre los modelos guardados en cada iteración vamos a escoger aquel que tiene un menor error absoluto medio en los datos de entrenamiento. De esta manera, no estamos utilizando los datos de test para realizar la elección del modelo y, por lo tanto, la evaluación de los modelos va a ser correcta.

5.4.2. Estudio de construcción de grafos

Combinando las dos estrategias de obtención de distancias vistas en la sección 5.2 junto con los limites por peso o por distancia nos permite jugar con varias combinaciones que son las que se han testeado para estudiar esta influencia de la forma del grafo en el entrenamiento. Para ello se han definido tres configuraciones:

- `graph_threshold`. Es el limite de sobrecarga de la distancia alternativa en la segunda estrategia.

- `graph_limit_distance`. Es el limite en los pesos habitual.

- `combine_graph`. Este valor booleano desactiva `graph_threshold` pero usa todavía la segunda estrategia. Es decir combina las dos estrategias obteniendo un grafo completo que tiene en cuenta las relaciones directas y los resultados alternativos y todavía se puede aplicar el limite a los pesos de la misma forma habitual.

Para probar estas configuraciones se ha construido una combinación base de características y se ha jugado con las tres características anteriores. La configuración por defecto se muestra en la tabla 5.1. Las configuraciones diferenciadas para realizar las nueve configuraciones que se ensayan se muestran en la tabla 5.2.

5.4.3. Ensayo de características

Al igual que con el estudio de grafos en este estudio se van a probar una serie de configuraciones diferentes con el objetivo de ver que influencia tienen ciertas características en los resultados finales del entrenamiento. Es decir, si influye su presencia y como es esta influencia.

Al igual que en la sección anterior se han probado nueve configuraciones diferentes que mostramos en las tablas 5.3, 5.4 y 5.5.

5.4.4. Comentarios y dificultades en el entrenamiento

Como se ha comentado en secciones anteriores el entrenamiento es muy lento y además se ha detectado un problema con *pyTorch* y C.U.D.A. y es que a pesar de borrar los modelos no reinicia el contexto provocando que la validación cruzada no funcionara correctamente. Para que no pasara eso había que esperar a que todos los procesos de torch se decargaran de la memoria de la GPU.

Esto se ha conseguido lanzando los entrenamientos desde un cron que va comprobando cada 5 minutos si tiene alguna configuración que lanzar siempre y cuando no este ya en ejecución el proceso de entrenamiento.

Tabla 5.1: Configuración por defecto de las características para el estudio de construcción de grafos.

Característica	**Configuración**
year	passthrough
month	passthrough
day	passthrough
hour	passthrough
minute	passthrough
weekday	passthrough
day_type	drop
season	drop
prematch	passthrough
match	passthrough
postmatch	passthrough
bank_holiday	passthrough
work_office_day	passthrough
school_day	passthrough
school_holiday	passthrough
state_of_alarm	passthrough
temperature	passthrough
humidity	passthrough
pressure	passthrough
radiation	passthrough
rain	passthrough

Tabla 5.2: Diferencias entre las nueve configuraciones para realizar el estudio de creación de grafos.

Configuración	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
graph_threshold	0	5	10	-1	-1	-1	-	-	-
graph_limit_distance	-	-	-	0.05	0.1	0.15	0.05	0.1	0.15
combine_graph	False	False	False	False	False	False	True	True	True

Tabla 5.3: Configuraciones 1 a 5 del estudio de características.

Característica	**C1**	**C2**	**C3**	**C4**	**C5**
year	drop	passthrough	passthrough	passthrough	passthrough
month	drop	passthrough	passthrough	passthrough	passthrough
day	drop	passthrough	passthrough	passthrough	passthrough
hour	drop	passthrough	passthrough	passthrough	passthrough
minute	passthrough	passthrough	passthrough	passthrough	passthrough
weekday	passthrough	passthrough	passthrough	passthrough	passthrough
day_type	drop	drop	drop	drop	drop
season	drop	drop	drop	drop	drop
prematch	drop	drop	passthrough	drop	drop
match	drop	drop	passthrough	drop	drop
postmatch	drop	drop	passthrough	drop	drop
bank_holiday	drop	drop	passthrough	drop	drop
work_office_day	drop	drop	passthrough	drop	drop
school_day	drop	drop	passthrough	drop	drop
school_holiday	drop	drop	passthrough	drop	drop
state_of_alarm	drop	drop	passthrough	drop	drop
occupation	drop	drop	drop	passthrough	drop
intensity	drop	drop	drop	passthrough	drop
temperature	drop	drop	drop	drop	passthrough
humidity	drop	drop	drop	drop	passthrough
pressure	drop	drop	drop	drop	passthrough
radiation	drop	drop	drop	drop	passthrough
rain	drop	drop	drop	drop	passthrough

Tabla 5.4: Configuraciones 6 a 9 del estudio de características.

Característica	C6	C7	C8	C9
year	passthrough	passthrough	passthrough	passthrough
month	passthrough	passthrough	passthrough	passthrough
day	passthrough	passthrough	passthrough	passthrough
hour	passthrough	passthrough	passthrough	passthrough
minute	passthrough	passthrough	passthrough	passthrough
weekday	passthrough	passthrough	passthrough	passthrough
day_type	drop	drop	drop	one_hot
season	drop	drop	drop	one_hot
prematch	passthrough	drop	drop	drop
match	passthrough	drop	drop	drop
postmatch	passthrough	drop	drop	drop
bank_holiday	drop	drop	drop	drop
work_office_day	drop	drop	drop	drop
school_day	drop	drop	drop	drop
school_holiday	drop	drop	drop	drop
state_of_alarm	drop	drop	drop	drop
occupation	drop	drop	drop	drop
intensity	drop	drop	drop	drop
temperature	drop	passthrough	passthrough	drop
humidity	drop	drop	drop	drop
pressure	drop	drop	drop	drop
radiation	drop	drop	drop	drop
rain	drop	passthrough	ordinal	drop

Tabla 5.5: Configuraciones 10 a 11 del estudio de características.

Característica	**C10**	**C11**
year	passthrough	passthrough
month	passthrough	passthrough
day	passthrough	passthrough
hour	passthrough	passthrough
minute	passthrough	passthrough
weekday	passthrough	passthrough
day_type	drop	drop
season	drop	drop
prematch	drop	drop
match	drop	drop
postmatch	drop	drop
bank_holiday	passthrough	drop
work_office_day	passthrough	drop
school_day	drop	passthrough
school_holiday	drop	passthrough
state_of_alarm	drop	drop
occupation	drop	drop
intensity	drop	drop
temperature	drop	drop
humidity	drop	drop
pressure	drop	drop
radiation	drop	drop
rain	drop	drop

Capítulo 6

Análisis de los resultados

6.1. Análisis creación de grafos

Tal y como se ha explicado en la sección 5.4.1 y la sección 5.4.2 se han construido nueve grafos diferentes construidos de tres formas distintas y con diferentes parámetros. Estas nueve configuraciones están pensadas para tener una muestra para poder comparar las diferentes estrategias presentadas, las características de cada configuración son:

- **Configuración 1**. Esta configuración utiliza la segunda estrategia que intenta hacer más dispersos los grafos con un sentido. Con su configuración resulta un grafo con solo las influencias primarias, es decir solo las aristas que representan un camino directo han sido añadidas. Tiene un total de 1254 aristas.

- **Configuración 2**. Esta configuración utiliza la segunda estrategia que intenta hacer más dispersos los grafos con un sentido. Con su configuración resulta un grafo con las influencias primarias sumando también aquellas que solo añaden un 10 % adicional.Tiene un total de 1274 aristas.

- **Configuración 3**. Esta configuración utiliza la segunda estrategia que intenta hacer más dispersos los grafos con un sentido. Con su configuración resulta un grafo con las influencias primarias sumando también aquellas que solo añaden un 20 % adicional.Tiene un total de 1295 aristas.

- **Configuración 4**. Esta configuración utiliza la estrategia habitual para formar el grafo y recorta los pesos menos significativos con el limite de 0.05. Tiene un total de 1317 aristas.

Tabla 6.1: Error absoluto medio por configuración en el estudio de grafos.

Configuración	**Maes medio**
1	0.283637
2	0.314777
3	0.292443
4	0.286052
5	0.255173
6	0.265404
7	0.318273
8	0.316512
9	0.278164

- **Configuración 5**. Esta configuración utiliza la estrategia habitual para formar el grafo y recorta los pesos menos significativos con el limite de 0.1. Tiene un total de 1098 aristas.
- **Configuración 6**. Esta configuración utiliza la estrategia habitual para formar el grafo y recorta los pesos menos significativos con el limite de 0.15. Tiene un total de 934 aristas.
- **Configuración 7**. Esta configuración utiliza la segunda estrategia para formar el grafo y recorta los pesos menos significativos con el limite de 0.05. Tiene un total de 888 aristas.
- **Configuración 8**. Esta configuración utiliza la segunda estrategia para formar el grafo y recorta los pesos menos significativos con el limite de 0.1. Tiene un total de 770 aristas.
- **Configuración 9**. Esta configuración utiliza la segunda estrategia para formar el grafo y recorta los pesos menos significativos con el limite de 0.15. Tiene un total de 680 aristas.

Una vez realizado los entrenamientos y seleccionado para cada partición el mejor resultado tenemos 10 datos por cada configuración que nos permite comparar una media del error medio para cada uno de las configuraciones para obtener aquella que ha dado mejor resultado que podemos observar en la tabla 6.1.

Podemos ver que la mejora configuración es la 5 y aunque la segunda mejor es la 9, en general son mejores las generadas con el método tradicional que las generadas teniendo en cuenta más factores.

Para poder obtener más conclusiones se presentan en la figura 6.1 el error medio para cada partición de entrenamiento y validación. Todas juntas se observa que más o menos todas han quedado dentro de los mismos valores pero con grandes diferencias en las formas entre ellas. Se presenta un grafo con cada configuración individualizada en la figura 6.2 que permite observar mejor cada forma de cada configuración.

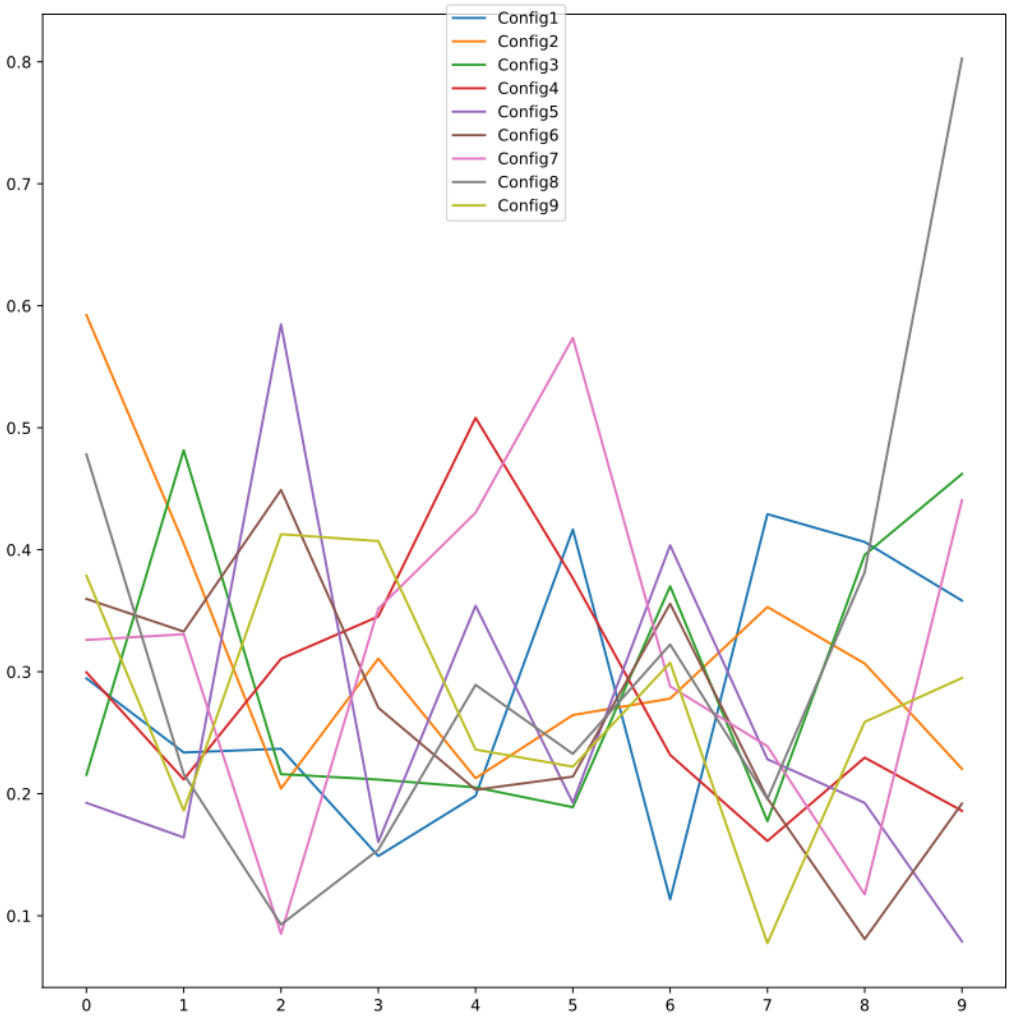

Figura 6.1: Evolución del entrenamiento y validación de cada configuración del estudio de grafos en la misma figura

Observando la forma de los entrenamientos se observa que a partir de la configuración 5 se observan valores muy bajos de error medio absoluto en todas las configuraciones, para observarlo se obtienen los cinco mejores valores junto a la configuración que los ha generado y se muestran en la tabla 6.2.

Viendo que el mejor valor pertenece a uno de estas configuraciones queda evidente que algún efecto positivo se ha conseguido ya que grafos con casi la mitad de aristas están funcionando

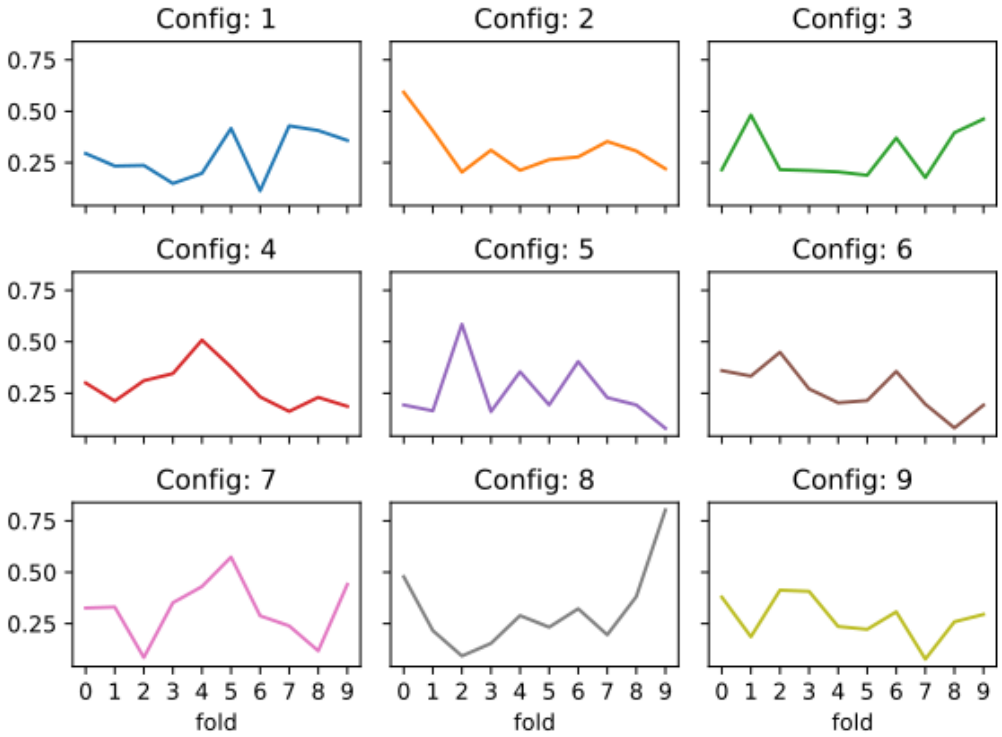

Figura 6.2: Evolución del entrenamiento y validación de cada configuración del estudio de grafos en distintos gráficos

de forma similar a grafos menos dispersos. Por otro lado cuando se usa la segunda estrategia en solitario se obtiene grafos menos dispersos y mas coherentes pero menos eficaces.

Todavía se puede mantener que la forma tradicional es superior a una forma de construir los grafos que tenga en cuenta más cosas. Pero algunos de los resultados parece que son suficientemente interesantes para continuar con este camino.

Tabla 6.2: 5 mejores valores.

Maes medio	**Configuración**
0,076411	9
0,076566	5
0,077006	6
0,088819	7
0,093972	8

6.2. Análisis de características

Tal como se ha explicado en 5.4.1 y en 5.4.3 se han probado 11 configuraciones diferentes con diferentes caracteristicas y diferentes formas de tratarlas:

- **Configuración 1**. Esta configuración es la base, ya que solo tiene en cuenta la variable objetivo *carga*.
- **Configuración 2**. Esta configuración solo utiliza las variables temporales.
- **Configuración 3**. Esta configuración utiliza las variables temporales junto a las de calendario.
- **Configuración 4**. Esta configuración utiliza las variables temporales junto a las de tráfico.
- **Configuración 5**. Esta configuración utiliza las variables temporales junto a las meteorológicas.
- **Configuración 6**. Esta configuración utiliza las variables temporales junto a las de calendario relacionadas con eventos.
- **Configuración 7**. Esta configuración utiliza las variables temporales junto a temperatura y lluvia tal cual.
- **Configuración 8**. Esta configuración utiliza las variables temporales junto a temperatura y lluvia, pero esta última transformada ordinalmente.
- **Configuración 9**. Esta configuración utiliza las variables temporales a las que se les ha añadido las variables tipo de día y estación del año tratadas de forma *one_hot*.
- **Configuración 10**. Esta configuración utiliza las variables temporales junto a las de calendario relacionadas con días festivos y laborales.
- **Configuración 11**. Esta configuración utiliza las variables temporales junto a las de calendario relacionadas con días lectivos y no lectivos.

Una vez realizado los entrenamientos y seleccionado para cada partición el mejor resultado tenemos 10 datos por cada configuración que nos permite comparar una media del error medio

Tabla 6.3: Error absoluto medio por configuración en el estudio de características.

Configuración	**Maes medio**
1	0.273904
2	0.260386
3	0.310472
4	0.389927
5	0.295733
6	0.222050
7	0.276874
8	0.236367
9	0.266300
10	0.262496
11	0.205367

para cada uno de las configuraciones para poder comparar los resultados y ver como influyen las características. Estos resultados los tenemos presentes en la tabla 6.3.

Para completar el valor medio del error absoluto adjuntamos la figura 6.3 con la forma de los entrenamientos.

Se va a utilizar la primera configuración como tope mínimo, cualquier configuración menor a ella se considerará errónea. Como todas las otras configuraciones tienen en común al menos la características horarias (excepto tipo de día y estación del año) que en solitario se han probado en la configuración 2. Como la configuración 2 es mejor que la 1 queda como el tope con el que se va a comparar el resto de las nueve configuraciones.

Hay seis configuraciones que han resultado peores que la configuración horaria por si misma:

- **Configuración 3** En esta configuración se habían añadido a las variables horarias todas las variables de calendario dando un resultado significativamente peor que incluso la primera configuración. Todavía resulta más curioso que probadas por grupos funcionales (calendario lectivo, festivo y eventos) todas han sido significativamente mejores que juntas. Hay una característica que tiene siempre el mismo valor, que es el estado de alarma. La lógica apunta a una sinergia negativa entre todas las características. Algo que podría entrar dentro de lo posible cuando alguna de estas características no mejorara las características

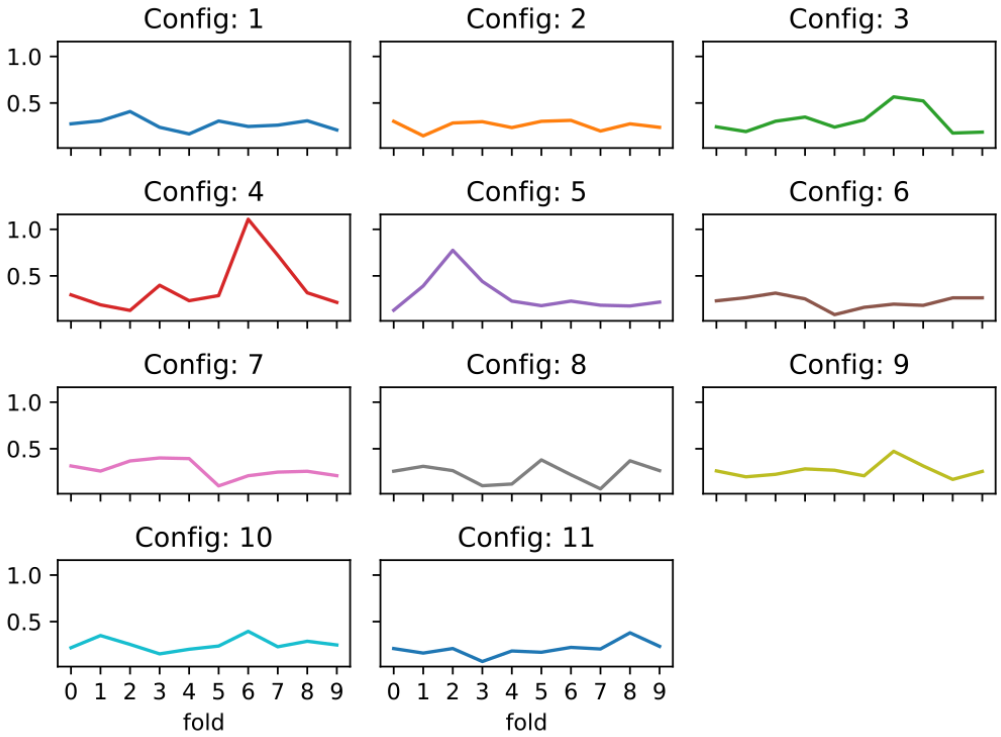

Figura 6.3: Evolución del entrenamiento y validación de cada configuración del estudio de características en distintos gráficos

horarias creando un *engaño* al predictor.

- **Configuración 4** En esta configuración se han añadido las características del tráfico, ocupación e intensidad. En esta configuración se esperaba un buen resultado ya que la variable objetivo es una combinación de estas dos junto a la intensidad de saturación. Sin embargo ha sido la peor configuración, superando en una de sus particiones incluso los valores enteros que no ha pasado en ninguna otra configuración durante todo el estudio.

- **Configuración 5** En esta configuración se han añadido todas las características meteorológicas sin tratarlas de ninguna forma, incluidas características como la radiación solar o la presión que, a priori, no parece que vayan a tener una importancia en el tráfico directa. Efectivamente el desempeño ha sido peor que incluso la configuración 1, cosa que parece lógica porque además de los problemas de idoneidad, la característica lluvia tiene la mayor parte del tiempo valores 0, por lo que igual hubiera sido mejor convertirla.

- **Configuración 7** En esta configuración se ha probado añadir solo las variables meteorológicas de temperatura y lluvia sin tocar, el desempeño ha sido mejor que en la configuración

5 pero aun así continuaba siendo peor que la configuración 1 incluso.

- **Configuración 9** En esta configuración se han añadido las características tipo de día y estación de año convertidas a *one_hot*. Como se esperaba han empeorado el resultado sin ellas. Parece lógico porque las estaciones del año cada vez significan menos y, en una ciudad como Madrid, el impacto entre ser un día entre semana, sábado o domingo puede estar atenuado o capturado este efecto con otras característica, como por ejemplo el día de la semana.

- **Configuración 10** En la configuración 10 se han añadido las características de días laborables y festivos. Aunque ha mejorado la configuración 1 no ha sido capaz de mejorar la configuración 2. Es un tanto sorprendente ya que los días festivos reducen el tráfico con mucha intensidad, quizá que sean tan pocos y que, es posible, que esta atenuación de tráfico no sea tan significativa como se tiende a pensar ya que el tráfico diario al centro de trabajo sea sustituido por otro tipo de tráfico.

Por otro lado hay tres configuraciones que han conseguido ser mejores que la configuración horaria por sí mismas:

- **Configuración 6** En esta configuración se ha probado añadir solo las características relacionadas con los eventos deportivos. El resultado ha sido sorprendentemente bueno, mejorando mucho la media de la configuración horaria por si misma. Intentando buscar una lógica a este buen desempeño se podría creer que al ser una característica muy especificas en su escala temporal y muy repetitivas respecto a la hora del día y el día de la semana funcionara muy bien para predecir sobre esos momentos dando ese gran resultado.

- **Configuración 8** En esta configuración se han añadido las características temperatura y lluvia como en la configuración 7. La única diferencia es que la lluvia ha sido codificada como un ordinal en cinco categorías. Esta diferencia ha hecho que se haya mejorado dramáticamente respecto a la otra configuración pareciendo demostrar que la categorización es mejor para esta característica.

- **Configuración 11** En esta configuración se han añadido los días lectivos y los no lectivos. Ha resultado ser la mejor configuración. Este buen desempeño parece indicar que pesa más el calendario escolar que el festivo. Es cierto que la diferencia entre los dos es muy grande

respecto a días de aplicación y que en Madrid por determinadas características de su configuración de distrito escolar y distribución por renta de los estudiantes, provoque más desplazamientos en vehículos de los que sería previsible.

Realmente es curiosa esta distribución de resultados. Por lo que parece puede haber sinergias negativas entre características, eso parece apreciarse en las características del calendario donde el los días festivos, lectivos y los eventos por si mismos son mejores que cuando se combinan. También parece ser que se está premiando las características en los dos extremos: características muy específicas como los eventos deportivos y características aplicables a una gran cantidad de medidas como el calendario escolar, donde los días no lectivos y lectivos no tienen la disparidad de aplicación que tienen los festivos y laborales.

La gran sorpresa ha sido que las características de tráfico fueran tan malas respecto al resto, en un principio se habían añadido por complitud pero no se utilizaban por qué la variable objetivo era una variable sintética calculada a partir de las otras posibilidades y parecía que fuera a distorsionar los resultados.

Por otro lado no parece sorprendente que tanto la temperatura, como la lluvia sean los factores meteorológicos más determinantes, tampoco es sorprendente que el pretratamiento de la variable a algo más acotado haya mejorado los resultados. Es bastante seguro que en una ciudad con mucha más precipitación no sería el caso, pero Madrid tiene un clima bastante árido y es lógico que este tipo de variables necesite un pretratamiento.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

El objetivo de este trabajo era sencillo de explicar pero difícil de ejecutar. Comprobar la influencia de la forma de construir grafos sobre el entrenamiento final o la verdadera influencia de cada característica sobre este mismo entrenamiento ha sido difícil y con conclusiones no definitivas.

Aun así hay indicios y muestras que pueden iluminar nuestra búsqueda. En la construcción de grafos ha quedado acreditado que influye la forma en que estos se generan, también ha quedado acreditado que no supone una diferencia abrumadora y que no supone tampoco una mejora en cualquier caso. Pero, dicho esto, abre el camino a cambiar cómo se utilizan los grafos y crearlos de forma más inteligente, también cambiando cómo se utilizan en el entrenamiento para aprovechar esas formas diferentes de construir el grafo. Por ejemplo utilizando modificadores para *premiar* las relaciones directas y modular las secundarias sin perder su influencia.

Por otro lado, las características han dejado todavía más dudas, ya que parece evidente que hay influencia cruzada entre ellas. A pesar de ello podemos apreciar que tienen influencia sobre el resultado y que en algunas cosas apunta ciertas impresiones.

- Las características concretas del tipo evento tienen una influencia cierta sobre el resultado final.

- Las características con poca incidencia temporal y poca concreción, como los festivos, pueden ser contraproducentes.

- Las características con valores equilibrados tienen más efecto que aquellas que tienen valores estacionarios.

Con estos datos que se infieren del trabajo hecho se puede concluir que tan importante como la información que se esta añadiendo al modelo es como esta información se combina con los datos ya existentes. Parece claro que la mera acumulación no mejora los resultados, quizá hay que darle un enfoque más personalizado.

El camino parece prometedor pero es evidente que muy preliminar, especialmente la creación de grafos más personalizados. Por esto debemos ir perseverando en estas ideas y ver si realmente consiguen aportar soluciones al problema que estamos tratando.

7.2. Trabajos futuros

Como hemos visto en las conclusiones hay una evidente relación en los resultados con dos cuestiones, como se forman los grafos y como se representan las características. Para abordar estos temas creo que se deberían enfocar las futuras acciones en dos direcciones:

1. Hacer que los grafos sean más representativos de la realidad que representan, para ello debería ser capaces de diferenciar entre relaciones primarias y secundarias pero modulando su influencia sin perderla. También el grafo debería cambiar durante el entrenamiento. En la vida real las relaciones entre nodos (puntos de una ciudad) cambian de forma dinámica, una manifestación, una obra o un accidente cambian estas relaciones y los grafos deberían capturar estos cambios.

2. Otra pelea es hacer más útil la información adicional que introducimos en el sistema. Parece evidente que la información temporal se hace omnipresente y tiene una grandísima influencia sobre el resultado final. También creo que se hace evidente que las relaciones secundarias entre las características no acaba de funcionar. Igual habría que plantearse que tener p.e. horas y festivos no captura realmente la relación y la sinergia que hay entre ellas, igual sería más efectivo tener horas festivas y horas laborales en lugar de la clasificación actual. Algo que debería, quizá, expandirse a una serie de categorías mucho más rica. P.e. horas nocturnas y festivas pero también hora lectiva y laboral.

Bibliografía

D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore. Network traffic prediction based on diffusion convolutional recurrent neural networks. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 246–251, 2019.

L. Bai, L. Yao, C. Li, X. Wang, and C. Wang. Adaptive graph convolutional recurrent network for traffic forecasting, 2020.

A. Belhadi, Y. Djenouri, D. Djenouri, and J. C.-W. Lin. A recurrent neural network for urban long-term traffic flow forecasting. *Applied Intelligence*, 50(10):3252–3265, 2020.

S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc, 2014.

E. Chen. Discrete temporal dynamic graph with recurrent structure. `https://github.com/dmlc/dgl/tree/master/examples/pytorch/dtgrnn`.

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. *On the properties of neural machine translation: Encoder-decoder approaches*. 2014.

H. Dong, F. Ding, H. Tan, and H. Zhang. Laplacian integration of graph convolutional network with tensor completion for traffic prediction with missing data in inter-city highway network. *Physica A*, 586(126474), 2022.

X. Dong, T. Lei, S. Jin, and Z. Hou. Short-term traffic flow prediction based on xgboost. *In 2018 IEEE 7th Data Driven Control and Learning Systems Conference*, pages 854–859, 2018.

M. Fauvell. Traffic. `https://github.com/mfauvell/traffic`.

C. Fisher. Google maps is improving travel etas with deepmind ai. *Engadget*, 2020.

S. J. Hanson. A stochastic version of the delta rule. *Physica D: Nonlinear Phenomena*, 42(1): 265–272, 1990.

Y. Huang, Y. Weng, S. Yu, and X. Chen. Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 678–685, 2019.

R. J. Hyndman and G. Athanasopoulos. *Time series cross-validation. In Forecasting: Principles and Practice*. OTexts, Australia, 3ª edition, 2021a.

R. J. Hyndman and G. Athanasopoulos. *Time series cross-validation. In Forecasting: Principles and Practice*. OTexts, Australia, 3ª edition, 2021b.

O. C. Ibe. *Diffusion processes. In Elements of random walk and diffusion processes, Wiley series in operations research and management science*. John Wiley & Sons Inc, Hoboken, N.J., 2013.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. `https://arxiv.org/abs/1502.03167`, 2015.

W. Jiang, J. Luo, M. He, and W. Gu. Graph neural network for traffic forecasting: The research progress. *ISPRS International Journal of Geo-Information*, 12(3), 2023. ISSN 2220-9964. doi: 10.3390/ijgi12030100. URL `https://www.mdpi.com/2220-9964/12/3/100`.

A. M. Karim, A. M. Abdellah, and S. Hamid. Long-term traffic flow forecasting based on an artificial neural network. *Advances in Science, Technology and Engineering Systems Journa*, 4, 2019.

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

O. Lange and L. Perez. Traffic prediction with advanced graph neural networks. https://www.deepmind.com/blog/traffic-prediction-with-advanced-graph-neural-networks, 2020.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, (521):436–444, 2015.

S. Lee and D. B. Fambro. Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting. *Transportation Research Record*, 1678(1): 179–188, 1999.

Y. Li, R. Yu, C. Shahabi, and Y. Liu. *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.* 2017.

P. Liu, Y. Zhang, D. Kong, and B. Yin. Improved spatio-temporal residual networks for bus traffic flow prediction. *Applied Sciences*, 9(4), 2019.

X. Liu, Y. Liang, C. Huang, H. Hu, Y. Cao, B. Hooi, and R. Zimmermann. Do we really need graph neural networks for traffic forecasting?, 2023.

Z. Lu, J. Xia, M. Wang, Q. Nie, and J. Ou. Short-term traffic flow forecasting via multi-regime modeling and ensemble learning. *Applied Sciences*, 10(1), 2020.

A. Luo, B. Shangguan, C. Yang, F. Gao, Z. Fang, and D. Yu. Spatial-temporal diffusion convolutional network: A novel framework for taxi demand forecasting. *ISPRS International Journal of Geo-Information*, 11(3), 2022. ISSN 2220-9964. doi: 10.3390/ijgi11030193. URL https://www.mdpi.com/2220-9964/11/3/193.

A. Micheli. Neural network for graphs: a contextual constructive approach. *IEEE TNN*, 20: 498–511, 2009.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *Proceedings of ICLR*, 2013.

V. Osipov, V. Nikiforov, N. Zhukova, and D. Miloserdov. Urban traffic flows forecasting by recurrent neural networks with spiral structures of layers. *Neural computing and applications*, 32(18):14885–14897, 2020.

B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: online learning of social representations. *Proceedings of KDD*, pages 701–710, 2014.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2009.

D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

E. Sobrini. *Diseño y comparación de modelos para la predicción del flujo de tráfico*. UNED, 2022.

A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE TNN*, 8:714–735, 1997.

L. S. V. Sri, A. Karthick, S. Akash, and R. Anuradha. Traffic prediction using graph neural network. In *2023 IEEE Guwahati Subsection Conference (GCON)*, pages 1–6, 2023. doi: 10.1109/GCON58516.2023.10183504.

S. Sun, H. Wu, and L. Xiang. City-wide traffic flow forecasting using a deep convolutional neural network. *Sensors*, 20(2), 2020.

I. Sutskever, O. Vinyals, and Q. V. Le. *Sequence to sequence learning with neural networks*. 2014.

H. Tampubolona and P.-A. Hsiung. Supervised deep learning based for traffic flow prediction. *In 2018 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS)*, pages 95–100, 2018.

M. V. D. Voort, M. Dougherty, and S. Watson. Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research. Part C, Emerging technologies,*, 4 (5):307–318, 1996.

Z. Wang, X. Su, and Z. Ding. Long-term traffic prediction based on lstm encoder-decoder architecture. *IEEE Transactions on Intelligent Transportation Systems*, 22(10):6561–6571, 2021.

B. M. Williams and L. A. Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

B. Yang, S. Sun, J. Li, and X. L. andYan Tian. Traffic flow prediction using lstm with feature enhancement. *Neurocomputing*, 332:320–327, 2019.

D. Zang, Y. Fang, Z. Wei, K. Tang, and J. Chent. Long term traffic flow prediction using residual net and deconvolutional neural network. In J.-H. Lai, C. L. Liu, X. Chen, J. Zhou, T. Tan, N. Zheng, and H. Zha, editors, *Pattern Recognition and Computer Vision*, pages 62–74. Curran Associates, IncSpringer International Publishing, 2018.

Y. Zhang and A. Haghani. A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, 58:308–324, 2015.

Anexo A

Estructura de la BBDD

A.1. Colecciones

La estructura de la BBDD en mongoDb es muy sencilla.

Las colecciones creadas son las siguientes:

- **calendar**. En esta colección se encuentra una tabla calendario que empieza en la primera fecha que vamos a trabajar 2019-01-01 y termina la última que vamos a trabajar el 31-05-2023 en intervalos de 15 minutos. Cada entrada incluye las características propias del calendario y las temporales.
- **calendar_aux**. Tabla auxiliar donde se han añadido las reglas para formar el calendario.
- **meteo_measures**. Las medidas ampliadas a frecuencia horaria de 15 minutos para cada estación meteorológica.
- **meteo_measures_aux**. La carga inicial horaria que se ha transformado en la colección anterior.
- **meteo_points**. Los datos de situación de las estaciones meteorológicas.
- **selected_points**. Los puntos seleccionados con su situación y su clasificación.
- **selected_points_aux**. Carga inicial con el id y la clasificación de cada punto seleccionado. Se ha usado para obtener la anterior.

- **selected_points_prepared_data**. Colección con todas las características unidas para cada punto e instante del calendario tratado de todos los puntos seleccionados.
- **traffic_measures**. Los datos brutos de medidas de tráfico para todos los puntos urbanos de Madrid en el periodo considerado.
- **traffic_points**. Las situación de cada punto de medida urbano de tráfico en la ciudad de Madrid.
- **traffic_points_measures_no_pandemic_count**. Número de medidas sin error y sin contar el estado de alarma para cada punto urbano de Madrid.

Además de estas colecciones se han ido creando colecciones temporales para tratar resultados sin visos de permanencia.

A.2. Queries

Se muestran ejemplos de las queries realizadas para apoyar la obtención de resultados.

A.2.1. Contador de medidas por punto

Con esta *query* se crea una colección donde a cada punto se le asocia el número de medidas que posee excluyendo el tiempo en que estuvo activo el estado de alarma.

```
[
  {
    $match:
      /**
       * exclude pandemic
       */
      {
        $and: [
          {
            error: {
              $eq: "N",
```

```
          },
        },
        {
          $or: [
            {
              date: {
                $gt: "2021-06-21 00:00:00",
              },
            },
            {
              date: {
                $lt: "2020-03-14 00:00:00",
              },
            },
          ],
        },
      ],
    },
},
{
  $group:
    /**
     * _id: The id of the group.
     * fieldN: The first field name.
     */
    {
      _id: "$point_id",
      count: {
        $sum: 1,
      },
    },
```

```
},
{
  $sort:
    /**
     * Provide any number of field/order pairs.
     */
    {
      count: -1,
    },
},
{
  $lookup:
    /**
     * from: The target collection.
     * localField: The local join field.
     * foreignField: The target join field.
     * as: The name for the results.
     * pipeline: Optional pipeline to run on the foreign collection.
     * let: Optional variables to use in the pipeline field stages.
     */
    {
      from: "traffic_points",
      localField: "_id",
      foreignField: "point_id",
      as: "tp",
    },
},
{
  $unwind:
    /**
     * path: Path to the array field.
```

```
      * includeArrayIndex: Optional name for index.
      * preserveNullAndEmptyArrays: Optional
      *   toggle to unwind null and empty values.
      */
    {
      path: "$tp",
    },
},
{
  $match:
    /**
     * query: The query in MQL.
     */
    {
      $and: [
        {
          "tp.capacity": {
            $ne: -1,
          },
        },
        {
          "tp.modified": false,
        },
      ],
    },
},
{
  $project:
    /**
     * specifications: The fields to
     *   include or exclude.
```

```
      */
      {
        count: "$count",
        latitude: "$tp.latitude",
        longitude: "$tp.longitude",
      },
  },
  {
    $out:
      /**
       * Provide the name of the output collection.
       */
      "traffic_points_measures_no_pandemic_count",
  },
]
```

A.2.2. Creación de la colección de puntos seleccionados

Con esta *query* se transforma la carga disponible de puntos seleccionadas realizada con un CSV en la colección base *selected_points*.

```
[
  {
    $group:
      /**
       * _id: The id of the group.
       * fieldN: The first field name.
       */
      {
        _id: "$point_id",
        count: {
          $sum: 1,
        },
```

```
      point_id: {
        $first: "$point_id",
      },
      tipus: {
        $first: "$tipus",
      },
      posicio: {
        $first: "$posicio",
      },
    },
},
{
  $lookup:
    /**
     * from: The target collection.
     * localField: The local join field.
     * foreignField: The target join field.
     * as: The name for the results.
     * pipeline: Optional pipeline to run on the foreign collection.
     * let: Optional variables to use in the pipeline field stages.
     */
    {
      from: "traffic_points",
      localField: "point_id",
      foreignField: "point_id",
      as: "tp",
    },
},
{
  $unwind:
    /**
```

```
      * path: Path to the array field.
      * includeArrayIndex: Optional name for index.
      * preserveNullAndEmptyArrays: Optional
      *   toggle to unwind null and empty values.
      */
     {
       path: "$tp",
     },
  },
  {
    $project:
     /**
      * specifications: The fields to
      *   include or exclude.
      */
     {
       point_id: "$point_id",
       latitude: "$tp.latitude",
       longitude: "$tp.longitude",
       tipus: "$tipus",
       posicio: "$posicio",
     },
  },
  {
    $out:
     /**
      * Provide the name of the output collection.
      */
     "selected_points",
  },
]
```

A.2.3. Máxima intensidad por punto según las medidas

En esta *query* se obtiene la máxima intensidad para un punto concreto (siempre excluyendo el estado de alarma), en el ejemplo el 3395.

```
[
  {
    $match:
      /**
       * exclude pandemic
       */
      {
        $and: [
          {
            error: {
              $eq: "N",
            },
          },
          {
            point_id: "3395",
          },
          {
            $or: [
              {
                date: {
                  $gt: "2021-06-21 00:00:00",
                },
              },
              {
                date: {
                  $lt: "2020-03-14 00:00:00",
                },
```

```
              },
            ],
          },
        ],
      },
  },
  {
    $group:
      /**
       * _id: The id of the group.
       * fieldN: The first field name.
       */
      {
        _id: "$point_id",
        maxIntensity: {
          $max: "$intensity",
        },
      },
  },
]
```

Anexo B

Instrucciones de instalación

Para la instalación del proyecto y de varios de los servicios es necesario tener instalado en el sistema una versión funcional de *git*.

B.1. Servicios

Para el correcto desarrollo de las pruebas y tal y como esta diseñado el conjunto de herramientas para ello son necesarios algunos servicios. Para ellos se ha utilizado un servicio de contenedores ligeros como es *docker*. Los contenedores son una forma magnifica de instalar diferentes servicios en el mismo hardware y que no interfieran entre sí.

Esta guiá no va a cubrir la instalación de *docker* ya que es diferente para cada sistema pero presupone que sea cual sea el método el sistema está funcionando correctamente. Además, aunque en las versiones recientes va incluido por defecto, también es necesario tener instalado *docker-compose*.

B.1.1. MongoDB

MongoDB es uno de los motores `NoSQL` más extendidos y maduros. En este proyecto es necesaria una instancia funcional de nuestro propio servidor.

Para hacerlo más sencillo se encuentra a disposición un repositorio con todo lo necesario para instalar y configurar una instancia del servicio. El repositorio está disponible aquí.

La instalación es muy sencilla y la puesta en marcha también. Solo es necesario clonar el repositorio y después ejecutar el script `start` para que construya e inicie el proyecto.

B.1.2. OpenRouteService

Para calcular distancias se ha utilizado el servicio de Open Route, si bien este servicio tiene una versión web que se puede utilizar directamente, debido al alto número de peticiones que se van a tener es más eficiente tener una instancia local.

Para montar una instancia local que facilita la tarea pero es un poco más complicado que el servicio *MongoDb*. El primer paso es clonar el repositorio disponible aquí.

Una vez clonado debemos crear una red dentro de nuestro sistema *docker*. Podemos hacerlo como queramos pero la manera más sencilla es crearlo vía cli:

```
docker network create openroute-net
```

Una vez creada la red ya podemos entrar dentro del directorio *docker* en el directorio donde hemos clonado el proyecto. Ya solo sería necesario, como en el caso anterior, ejecutar el script `start` para que construya e inicie el proyecto.

En este caso todavía no está listo, *openrouteservice* es una *API-rest* escrita en *Java*. Para usarla eficazmente hay que montar un servidor web que nos permita utilizar una url más cómoda. En nuestro caso vamos a usar un proxy reverso creado con nginx. Se detalla la configuración en la sección B.1.2.1.

B.1.2.1. Nginx

Nginx es un servidor web ligero que destaca especialmente como proxy reverso. Para facilitar la configuración usaremos este repositorio git para la configuración necesaria.

Una vez clonado nuestro repositorio crearemos un fichero `docker-compose.yml` con el siguiente contenido:

```
version: '3'
services:
    reverseproxy:
        build:
            context: ./.docker
            dockerfile: Dockerfile
        # image: reverseproxy
```

```
        ports:
            - 80:80
        restart: always
        volumes:
            - './sites-enabled:/etc/nginx/sites-enabled'
        networks:
            - openroute-net

networks:
    openroute-net:
        external: true
```

Una vez realizado esto crearemos dentro de la carpeta `sites-enabled` (si no existe la crearemos) un fichero, da igual el nombre, con el siguiente contenido:

```
server {
    listen 80;
    server_name openroute.local;

    access_log /var/log/nginx/www_openroute_access.log;
    error_log /var/log/nginx/www_openroute_error.log info;



    location / {
        resolver 127.0.0.11 valid=30s;
        set $upstream_api http://ors-app.openroute-net:8080;
        proxy_pass $upstream_api;
        include /etc/nginx/proxy_params;
    }
}
```

Ya podemos arrancar el servicio con la ejecución del script *start*.

Por último para que nuestro nodo local responda a la dirección *openroute.local* solo tenemos

que editar el fichero `hosts` (ver como hacerlo en cada sistema operativo) para añadir una linea como esta:

```
127.0.0.1   openroute.local
```

B.2. Proyecto

Una vez tenemos todos los servicios instalados y ejecutándose ya podemos instalar el proyecto. Este proyecto se encuentra disponible en [Fauvell].

Antes de continuar nos aseguraremos que tenemos *python3* con soporte para entornos virtuales en nuestro sistema. Así que una vez hayamos clonado el proyecto, p.e. en el directorio `traffic`, crearemos un entorno virtual en él:

```
python3 -m venv traffic
```

Pasamos al interior del directorio y activamos el entorno virtual:

```
source bin/activate
```

Una vez activado, ya dentro del entorno, realizamos la instalación:

```
pip install torch pymongo xmltodict tqdm numpy requests pandas scikit-learn
pip install pymongoarrow matplotlib
pip install  dgl -f https://data.dgl.ai/wheels/cu118/repo.html
pip install  dglgo -f https://data.dgl.ai/wheels-test/repo.html
```

Es importante hacer notar que la librería *dgl* depende de nuestro sistema, si tiene CUDA y de que versión estamos usando, para más información ver aquí.

Con esto ya tendremos el proyecto preparado y podremos ya colocar los datos en el lugar adecuado y empezar a ejecutar los diferentes scripts.

Anexo C

Contenido de la entrega

En la entrega adjunta se incluye el siguiente contenido:

- Memoria.
- Declaración jurada de autoría.
- Licencia de uso.
- Documento texto plano con enlace a repositorio con datos adicionales.