Universidad de la República

Facultad de Ingeniería

Estimación de velocidad vehicular mediante análisis predictivo sobre redes

Informe de Proyecto de Grado presentado por

Matias Cikurel

como requisito de graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Pablo Rodriguez-Bocca

Montevideo, 9 de septiembre de 2023

Agradecimientos

A mi madre y mi hermana, quienes convivieron a diario con los efectos secundarios de este proyecto y de esta carrera, que no son pocos. A mi padre, a Carlos y al resto de mi familia, quienes estuvieron ahí todos estos años.

A Sebastián Cortabarría y Sofía Tito Virgilio, con quienes compartí más materias de las que puedo recordar y aun sacándonos chispas logramos llevar a buen puerto todas las entregas que se nos pararon adelante.

A mis amigas Sofía Alberti, Martina Font y Cecilia Toledo, a quienes conocí gracias a una desatención a la hora de inscribirme a un examen, y han sido un soporte emocional clave que me mantuvo a flote, especialmente en el último tramo.

Por último, agradecer a mi tutor, Pablo Rodríguez-Bocca, por guiarme en este proyecto, enseñarme algunos de los secretos del mundo académico y por salvarme de desperdiciar horas investigando algunas cosas que no tenían sentido.

No tengo dudas que sin estas personas no hubiera llegado hasta acá.

Resumen

En este proyecto buscamos estudiar la bondad del modelo Graph WaveNet en la predicción de velocidad de tráfico en un contexto de carencia de datos. Analizando en primer lugar la sensibilidad del modelo frente a la falta de datos, procedemos a plantear una serie de escenarios basados en dos conjuntos de datos vinculados a las ciudades de Los Ángeles y Montevideo. Planteando una serie de variantes con distintos grados de carencia de datos construidas eliminando datos de manera aleatoria, evaluamos una serie de métodos de imputación para observar el impacto en el rendimiento del modelo.

Estos métodos de imputación están vinculados a distintos valores estadísticos como son la media general y la media por ubicación, por mencionar algunos. Además, evaluamos una alternativa de imputación que escapa a esta línea de valores estadísticos: el método *forward-fill*, que en orden cronológico sustituye cada dato faltante con el inmediatamente anterior.

Los resultados obtenidos muestran que con un 20 % de datos faltantes o más, la imputación de datos comienza a producir mejores resultados sobre la opción de no imputar datos. En la evaluación de los métodos de imputación planteados, el que destaca por sobre los demás es el método *forward-fill*. Este comportamiento se acentúa a medida que aumenta la cantidad de datos faltantes en el *dataset*, permaneciendo siempre este método como el mejor. Este resultado puede estar influenciado por la manera en que se eliminan datos para generar los escenarios de prueba, donde al eliminar datos de manera aleatoria, el método *forward-fill* logra reconstruir la señal mejor que el resto.

Palabras clave: aprendizaje automático, series temporales, predicción de velocidades, imputación de datos, Graph WaveNet, *forward-fill*.

Índice general

Capítulo 1

Introducción

Actualmente, la tecnología se ha integrado a prácticamente todos los ámbitos de la vida cotidiana. Con el creciente desarrollo de técnicas de aprendizaje automático de los últimos años, se ha logrado comenzar a explotar el potencial de los grandes volúmenes de datos recolectados, construyendo modelos capaces de realizar tareas de clasificación y predicción sobre estos datos. Una de las áreas que no escapa a esta realidad es la vinculada al transporte urbano, más específicamente el tráfico vehicular.

Mediante la implementación de redes de sensores se ha logrado recabar cantidades importantes de información vinculada a la velocidad y el volumen de tráfico en distintas partes del mundo. Esto permite, en conjunto con técnicas de aprendizaje automático, desarrollar modelos y arquitecturas específicas para atacar el problema de predicción de tráfico.

El problema de predicción de tráfico se puede definir a nivel conceptual como la tarea de generar, en base a información histórica, predicciones sobre las velocidades o el volumen de tráfico en un conjunto de ubicaciones dadas. Es un problema de gran relevancia en la actualidad debido a su uso no solo a nivel comercial en aplicaciones de transporte, sino también por su importancia en el área de planeamiento urbano, en particular en sistemas de tráfico inteligentes.

El mayor desarrollo en esta área se ha dado en los últimos cinco años, con la introducción de una serie de arquitecturas diseñadas con el objetivo concreto de mejorar la predicción de velocidad de vehicular. En este trabajo, vamos a enfocarnos en la arquitectura de red neuronal Graph WaveNet (Wu, Pan, Long, Jiang, y Zhang, 2019), y vamos a probar su bondad en distintos escenarios de carencia de datos. Es sabido que las técnicas de aprendizaje automático son particularmente sensibles a la carencia de datos (por ejemplo en (Weerakody, Wong, Wang, y Ela, 2021) se estudia su efecto en problemas de series temporales), sin embargo no encontramos referencias de su estudio para el problema de predicción de tráfico, y en particular para la relativamente joven arquitectura Graph WaveNet.

Para esto, planteamos una serie de experimentos con distintos objetivos. (1) En primer lugar, analizamos la respuesta del modelo Graph WaveNet fren-

te a la carencia de datos. Partiendo de un *dataset* de referencia, construimos variantes eliminando distintas cantidades de datos de forma aleatoria, para finalmente observar que tanto afecta al modelo la cantidad de datos faltantes. (2) Luego, planteamos nuevamente escenarios de datos faltantes y proponemos una serie de métodos de imputación basados en valores estadísticos y estudiamos sus rendimientos en el contexto del *dataset* de referencia, con el objetivo de estudiar si estos métodos agregan valor en un contexto de carencia de datos y así aprovechar de mejor manera los datos existentes. (3) Por último, realizamos una vez más esta experiencia, pero sobre un *dataset* real, construido particularmente para este proyecto en base a información de velocidades obtenida de Montevideo. El objetivo de este último experimento es evaluar si los hallazgos de los métodos de imputación de datos sobre el *dataset* de referencia se mantienen aun en este nuevo *dataset*, el cuál presenta importantes desafíos respecto a la distribución de datos faltantes.

El documento se divide en cinco capítulos. En el Capítulo 2 presentamos la formalización del problema, una descripción de los dos conjuntos de datos utilizados, una serie de fundamentos teóricos necesarios para comprender en forma detallada la arquitectura Graph WaveNet, y finalmente un breve repaso de algunos de los trabajos relacionados más importantes para el contexto de este proyecto. En el Capítulo 3 detallamos el trabajo realizado sobre datos de Montevideo para construir uno de los *datasets*. Además, exponemos el trabajo realizado en términos de análisis descriptivo sobre los *datasets* y las conclusiones extraídas a partir del mismo. Comentamos los aspectos más importantes relativos a la implementación del modelo a nivel de código fuente, y profundizamos en el área de las métricas utilizadas en este trabajo. Por último, presentamos la fase de búsqueda de hiper-parámetros. Luego, en el Capítulo 4, desarrollamos en profundidad los experimentos realizados. Detallamos el planteo de objetivos, ejecución y resultados obtenidos para cada una de las experiencias mencionadas anteriormente. Finalmente en el Capítulo 5, presentamos las conclusiones finales y los resultados clave del proyecto. Enumeramos los aportes realizados y continuamos mencionando algunos aspectos que por distintos motivos no lograron ser ejecutados y que son presentados en última instancia como trabajo futuro y posibles mejoras.

Capítulo 2

Estado del arte

En este Capítulo presentamos los conceptos necesarios que enmarcan este proyecto. Describimos los aspectos más importantes del problema a tratar y explicamos la obtención y el procesamiento de los datos utilizados. Por último, introducimos tanto la arquitectura de red neuronal seleccionada para analizar en este trabajo (Graph WaveNet), así como algunos otros modelos interesantes de mencionar.

2.1. Predicción de tráfico

La predicción de tráfico vehicular es una tarea que ha tomado un papel sumamente importante en los últimos años, siendo una pieza clave en el desarrollo de sistemas de tráfico inteligentes y en el planeamiento urbano, como también por su uso en aplicaciones comerciales vinculadas a transporte y movilidad. El objetivo principal de la predicción de tráfico es predecir las condiciones futuras del tráfico (usualmente velocidad o cantidad de vehículos) en una red de calles utilizando un conjunto de observaciones pasadas.

El desarrollo de los sistemas de tráfico inteligentes ha generado un fuerte crecimiento en la cantidad de información de tráfico disponible, permitiendo atacar este problema mediante métodos de aprendizaje automático. La principal dificultad de este problema está dada por la estructura topológica de la red de calles, modelada generalmente mediante grafos, y las complejas dependencias espacio-temporales entre los sensores de la red. Algunas de las aplicaciones más comunes de esta tarea son la planificación de rutas, la estimación de tiempo de viaje y el control de tráfico, entre otras. En (Creß, Bing, y Knoll, 2022) se presenta una recopilación y análisis de distintos sistemas de transporte inteligentes desde 2009 a 2021, que sirve como buena referencia del estado reciente de estos sistemas.

Modelamos una red de calles con N sensores como un grafo ponderado y dirigido, $\mathcal{G} = (V, E)$, donde V representa el conjunto de sensores, $|V| = N$, y E el conjunto de aristas que modelan la conexión entre los nodos. Por lo

general, el grafo $\mathcal{G}$ se representa matemáticamente con la matriz de adyacencia ponderada $\mathbf{A} \in \mathbb{R}^{N \times N}$. Los sensores se encuentran en puntos geográficos de interés, sobre los cuales queremos pronosticar, en un futuro próximo, la velocidad de los vehículos que los transitan. Conocemos un histórico de las velocidades en dichos sensores para distintos pasos (discretos) de tiempo $t \in \mathbb{N}$, simbolizado con $\mathbf{y}_t \in \mathbb{R}^N$, y también conocemos otros posibles atributos predictores (día de la semana, horario, u otros datos relevantes), simbolizados con $\mathbf{X}_t \in \mathbb{R}^{N \times D}$ (N sensores con D atributos). Podemos formalizar matemáticamente la tarea de predicción de velocidad de tráfico mediante la Ecuación 2.1:

$$\mathbf{y}_{(t-T):t}, \mathbf{X}_{(t-T):t}, \mathbf{A} \xrightarrow{f} \mathbf{y}_{(t+1):(t+S)}, \tag{2.1}$$

donde T representa la cantidad de pasos históricos conocidos, y S la cantidad de pasos futuros a predecir. El objetivo es aprender la función $f()$ para poder predecir las velocidades en los siguientes S pasos en todos los sensores: $\mathbf{y}_{(t+1):(t+S)} \in \mathbb{R}^{N \times S}$, conocidas las velocidades anteriores: $\mathbf{y}_{(t-T):t} \in \mathbb{R}^{N \times T}$, los atributos adicionales en los sensores: $\mathbf{X}_{(t-T):t} \in \mathbb{R}^{N \times D \times T}$, y la matriz de adyacencia: $\mathbf{A} \in \mathbb{R}^{N \times N}$

2.2. Imputación de datos de tráfico

La obtención de datos del mundo real suele traer aparejado problemas, tanto técnicos como económicos por mencionar algunos, que suelen resultar en conjuntos de datos incompletos. Esta recolección de datos suele llevarse adelante utilizando sistemas multisensor, que si bien permiten obtener grandes volúmenes de datos, también es habitual que debido a fallas en el funcionamiento o interferencias, se generen mediciones incompletas o con corrupción en algunos valores. Esta situación es problemática ya que la gran mayoría de los modelos basados en aprendizaje automático que utilizan estos datos como insumo generalmente requieren de datos completos. La irregularidad en estos conjuntos de datos también limita la capacidad de estos métodos a la hora de realizar tareas de predicción o clasificación, especialmente cuando nos referimos a tareas en el área de modelado secuencial. Por lo tanto, surge naturalmente la necesidad de desarrollar mecanismos para completar las observaciones de manera artificial intentando mejorar la calidad de los datos usados en el entrenamiento del modelo y trasladar esta mejora hacia las métricas del modelo final.

Si bien la tarea de imputación de datos se puede generalizar a distintos tipos de conjuntos de datos, en este trabajo nos enfocamos en la imputación de series temporales, más específicamente la imputación de datos de tráfico. Los métodos de imputación varían desde soluciones simples a complejos modelos entrenados con el objetivo de completar los datos faltantes en el conjunto de datos de entrenamiento detectando las dependencias temporales y espaciales subyacentes. Algunas soluciones de imputación sencilla están vinculadas a utilizar valores estadísticos como pueden ser la media, la mediana y la moda. Por otro lado, las soluciones más complejas varían desde enfoques basados en redes neuronales convolucionales (Benkraouda, Thodi, Yeo, Menendez, y Jabari, 2020),

auto-regresiones sobre tensores de bajo rango (Chen, Lei, Saunier, y Sun, 2022), y múltiples enfoques basados en redes neuronales recurrentes como se expone en (Weerakody y cols., 2021). Este último trabajo será explicado con mayor detalle en la Sección 2.5, ya que plantea un análisis muy completo sobre distintos tipos de mecanismos de imputación, desde variantes sencillas a modelos más complejos y planteando también adaptaciones para que algunos algoritmos resuelvan en su propio flujo el problema de la carencia de datos.

2.3. Conjunto de datos

Para este trabajo utilizamos dos conjuntos de datos de velocidades. El primero es un *dataset* público de referencia utilizado en la mayor parte de los trabajos realizados vinculados a predicción de tráfico. Este *dataset*, al que nos referiremos como METR-LA (Li, Yu, Shahabi, y Liu, 2018), es una colección de velocidades a lo largo de 4 meses en 207 ubicaciones en un conjunto de autopistas de la ciudad de Los Ángeles, USA, tomados del trabajo *Big Data and Its Technical Challenges* (Jagadish y cols., 2014). Ver las ubicaciones de este conjunto de datos en la Figura 2.1.

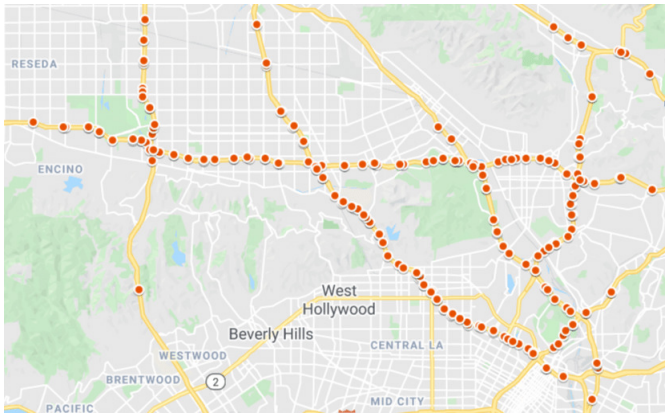


Figura 2.1: Visualización de nodos METR-LA

El segundo *dataset*, al que llamaremos MVD, es un *dataset* construido a partir de una colección de datos publicados por la Intendencia de Montevideo (*Velocidad promedio vehicular en las Principales Avenidas de Montevideo - Catálogo de Datos Abiertos*, 2021). Procesando estos datos, como explicaremos más adelante, construimos un *dataset* de velocidades a lo largo de 7 meses en 124 ubicaciones distribuidas en la ciudad de Montevideo. Ver las ubicaciones de este conjunto de datos en la Figura 2.2.

Para ambos *dataset*s, las lecturas de velocidad de cada sensor son agrupadas en ventanas de 5 minutos y finalmente promediadas.

Respecto a los datos, otro elemento importante para nuestro trabajo es la matriz de adyacencia dirigida. La matriz de adyacencia es la representación

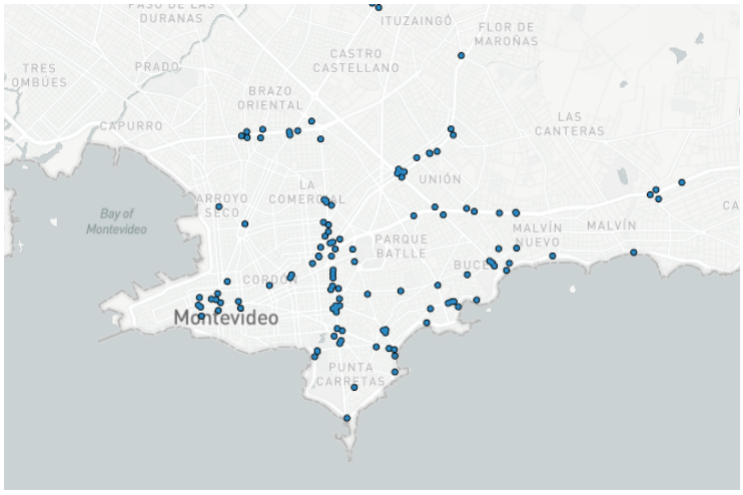


Figura 2.2: Visualización de nodos MVD

elegida para el grafo que modela la red de calles. Aquí se define el peso de las aristas utilizando la distancia por calle entre cada par de sensores. Siguiendo la metodología presentada en (Wu y cols., 2019) basada en (Li y cols., 2018), la definición de $\mathbf{A}$ está dada por la aplicación de un núcleo Gaussiano umbralizado:

$$\mathbf{A}_{ij} = \text{máx}\{e^{-\frac{dist(i,j)^2}{\sigma^2}} - \kappa, 0\}. \tag{2.2}$$

Donde $\mathbf{A}_{ij}$ es el peso de la arista entre los nodos i y j, $dist(i,j)$ es la distancia por calle entre los nodos i y j, σ es la desviación estándar de todas las distancias entre cada par de nodos y κ es un umbral definido de manera arbitraria. El valor del umbral se mantiene como está presentado en (Wu y cols., 2019), $\kappa = 0{,}1$.

En el caso del *dataset* METR-LA, esta matriz de adyacencia ya se encontraba construida y disponible para ser utilizada. Para el caso del *dataset* MVD, utilizamos la API de *Google Maps Distance Matrix*[1], para obtener la distancia por calle entre cada par de sensores ubicados en Montevideo.

2.3.1. Obtención de datos de Montevideo

Los datos obtenidos del catálogo abierto de datos de la Intendencia de Montevideo presentan el siguiente formato:

- `id_detector`: identificador del sensor,
- `id_carril`: identificador del carril en el que se monitorea la velocidad,

[1] Distance Matrix API overview - Google for Developers `https://developers.google.com/maps/documentation/distance-matrix/overview`. Accedido el 2023-06-15.

- `fecha`: día en que se toma la muestra,
- `hora`: hora en que se toma la muestra,
- `dsc_avenida`: nombre de la vía sobre la que se mide el tránsito,
- `dsc_int_anterior`: nombre de la vía que forma el cruce desde donde vienen los vehículos,
- `dsc_int_siguiente`: nombre de la vía que forma el cruce hacia donde circulan los vehículos,
- `latitud`: latitud de la ubicación del sensor,
- `longitud`: longitud de la ubicación del sensor,
- `velocidad_promedio`: promedio de las velocidades de los autos que circularon por el carril durante los últimos 5 minutos.

En la Sección 3.1 detallamos el procesamiento de los datos para Montevideo, explicando las decisiones tomadas en la transformación desde el formato aquí presentado, hasta su forma final para ser tomados como entrada de Graph WaveNet.

2.4. Arquitectura Graph WaveNet

En esta sección explicaremos algunos conceptos fundamentales sobre aprendizaje profundo. Introduciremos a nivel general elementos como redes convolucionales clásicas, redes convolucionales de grafos y redes temporales convolucionales necesarias para posteriormente explicar la arquitectura Graph WaveNet en detalle.

2.4.1. Fundamentos

La inteligencia artificial es un área de la computación que tiene como objetivo simular, con medios computacionales, los procesos de aprendizaje y las capacidades cognitivas humanas para llevar adelante tareas complejas de manera automatizada y eficiente. Dentro de este amplio campo podemos definir aprendizaje automático como el subcampo que utiliza algoritmos que aprenden de manera automática mediante un proceso de entrenamiento a partir de importantes volúmenes de datos, generando modelos computacionales para resolver distintos tipos de tareas y sin estar programados explícitamente para resolverlas. Una famosa definición formal de aprendizaje automático es la brindada por Tom Mitchell en su libro *Machine Learning* (Mitchell, 1997), en la que se dice que un algoritmo aprende de la experiencia E respecto a una clase de tarea T y una medida de rendimiento P, si el rendimiento al realizar una tarea de clase T, medido utilizando P, mejora con la experiencia E.

Podemos dividir el aprendizaje automático en tres categorías distintas, de acuerdo a los datos de entrada del algoritmo y el objetivo que se desea cumplir. En primer lugar tenemos el aprendizaje supervisado. Aquí los conjuntos de datos de entrada se encuentran completamente etiquetados, en muchos casos como resultado de un importante y costoso proceso de etiquetado manual. El objetivo de estos datos etiquetados es guiar el aprendizaje del algoritmo para que este aprenda a retornar el resultado deseado de acuerdo a la tarea y luego evaluar el rendimiento del modelo. En general, este método es utilizado en problemas de clasificación y regresión. Siguiendo esta línea, en segundo lugar tenemos el aprendizaje semi-supervisado. Este tipo de aprendizaje es similar al supervisado pero difiere en que los datos se encuentran parcialmente etiquetados. Esto es útil en casos donde el conjunto de datos es extremadamente grande y permite aliviar el costo del etiquetado manual completo. Además, existen casos en los que este tipo de aprendizaje muestra un mejor rendimiento, como sucede con algunas tareas sobre grafos. Por último, mencionamos la técnica de aprendizaje no supervisado. En este escenario los datos de entrada no se encuentran etiquetados y el objetivo es que el propio modelo detecte estructuras, patrones y correlaciones ocultas en los datos. En nuestro caso trabajaremos con aprendizaje automático supervisado.

Existe una gran cantidad de métodos en el área de aprendizaje automático. La familia que nos interesa destacar son los métodos de aprendizaje profundo, que tienen como elemento medular las redes neuronales de múltiples capas. Dentro del aprendizaje profundo se han desarrollado una gran cantidad de métodos basados en distintos tipos de redes neuronales. Sin embargo mencionaremos algunos de los tipos más importantes para la comprensión de este trabajo.

Para una mejor introducción sobre el aprendizaje automático y los métodos relacionados, recomendamos el libro *Deep Learning* (Goodfellow, Bengio, y Courville, 2016) para explorar algunos conceptos fundamentales partiendo del álgebra lineal hasta modelos complejos de aprendizaje profundo. Por otro lado, la revisión presentada en (LeCun, Bengio, y Hinton, 2015) resume de muy buena manera algunas de las arquitecturas más importantes del área.

Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN) son una familia de redes neuronales artificiales ampliamente utilizadas para resolver problemas de información secuencial, como por ejemplo el procesamiento de series temporales. Su estructura le permite a estas redes trabajar con entradas de largo variable, conservando un estado interno que se asemeja a la capacidad de tener memoria. Sin embargo, una de las debilidades de las primeras arquitecturas de tipo RNN, es que comienzan a fallar al momento de trabajar con dependencias de largo alcance y presentan problemas como desvanecimiento y explosión de gradientes. Esto motivó el desarrollo de dos arquitecturas muy populares en esta familia: *long short-term memory* (LSTM) (Hochreiter y Schmidhuber, 1997) y *gated recurrent units* (GRUs) (Cho, van Merrienboer, Bahdanau, y Bengio, 2014). En (Chung, Gulcehre, Cho, y Bengio, 2014) se expone una interesante evaluación

del rendimiento de estas dos arquitecturas en tareas de modelado secuencial.

Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN) (LeCun y cols., 1989) son una clase de red neuronal artificial con la particularidad de que aplican operaciones de convolución y *pooling*. Las capas convolucionales, elemento principal de este tipo de redes, consisten en un conjunto de filtros convolucionales que se aplican a lo largo y ancho de toda la entrada realizando operaciones de convolución entre los valores de entrada y el filtro, con el objetivo de detectar *features* específicas para el problema a resolver. Por otro lado, las capas de *pooling* aplican filtros de agregación, reduciendo la dimensión de los parámetros de entrada a lo largo de la red. Es muy común que este tipo de redes se utilicen en tareas de clasificación y detección en imágenes, donde existe alta correlación espacial como son los casos de las arquitecturas *AlexNet* (Krizhevsky, Sutskever, y Hinton, 2012) y *ResNet* (He, Zhang, Ren, y Sun, 2016). Sin embargo, también se han usado exitosamente para problemas con correlación temporal en el área de series temporales (F. Yu y Koltun, 2016), (Zhao, Lu, Chen, Liu, y Wu, 2017).

Redes Convolucionales de Grafos

En muchos casos, la mejor manera de formalizar problemas es mediante estructuras de grafos, particularmente cuando se trata de información compleja en dominios irregulares o no euclidianos. Por lo tanto, es natural querer aprovechar el potencial de las redes neuronales para aplicarlo a este tipo de estructuras, y es con este fin que se desarrollan las redes convolucionales de grafos (GCN). La idea detrás de la GCN es intuitivamente similar a la de una red neuronal convolucional convencional. Para cada nodo, se aplican operadores de convolución en base a la información de sus vecinos.

Sea $\mathcal{G} = (V, E)$ un grafo genérico. Se representa este grafo mediante una matriz de adyacencia $\mathbf{A} \in \mathbb{R}^{N \times N}$, donde N es la cantidad de nodos del grafo. Tenemos además que cada nodo i tiene un vector de *features* de entrada de dimensión D, $x_i \in \mathbb{R}^D$. Uniendo los vectores construimos la matriz de *features* $\mathbf{X} \in \mathbb{R}^{N \times D}$. Una GCN genérica toma como entrada las matrices $\mathbf{A}$ y $\mathbf{X}$, y produce una matriz de *features* de salida $\mathbf{Z} \in \mathbb{R}^{N \times F}$, donde F es la cantidad de *features* de salida para cada nodo.

Las ecuaciones dispuestas a continuación esquematizan el funcionamiento de una GCN a alto nivel:

$$\mathbf{H}^{(0)} = \mathbf{X}, \tag{2.3}$$

$$\mathbf{H}^{(k+1)} = f(\mathbf{H}^{(k)}, \mathbf{A}) \; \forall k \in [0, \ldots, L], \tag{2.4}$$

$$\mathbf{Z} = \mathbf{H}^{(L)}. \tag{2.5}$$

La Ecuación 2.3 muestra que la entrada de la primera capa es la matriz de *features*, $\mathbf{X}$. Luego, la Ecuación 2.4 indica que cada capa está definida como la aplicación de una función $f()$, en general no lineal, sobre la salida de la capa

anterior y la matriz de adyacencia **A**. Siendo L la cantidad de capas de la arquitectura definida, la última capa retorna Z, como muestra la Ecuación 2.5.

Existen dos grandes categorías de GCN (o formas de interpretarlas), las redes espaciales y las redes espectrales. Las redes espectrales, derivadas del área del procesamiento de señales, están basadas en la teoría espectral de grafos. Se definen filtros espectrales y se utiliza la descomposición espectral de la matriz Laplaciana vinculada al grafo para construir la operación de convolución. Es bajo esta categoría donde se introducen las redes convolucionales para grafos (Bruna, Zaremba, Szlam, y LeCun, 2014), uniendo por primera vez la teoría espectral junto al aprendizaje profundo para grafos no dirigidos. Construyendo sobre esta arquitectura, (Defferrard, Bresson, y Vandergheynst, 2016) propone una serie de mejoras para atacar algunas de las carencias de la arquitectura anterior aumentando la eficiencia y mejorando el rendimiento. Partiendo de este enfoque de convolución espectral, (Kipf y Welling, 2017) implementan algunas simplificaciones al modelo mejorando sustancialmente la velocidad de entrenamiento y el rendimiento en tareas predictivas con especial énfasis en aprendizaje semi-supervisado.

Por otro lado, existen las redes GCN espaciales. Dentro de esta categoría existen distintos tipos de red, como por ejemplo las basadas en redes de convolución clásicas o las basadas en métodos de propagación, por mencionar algunas. Lo que tienen en común estas redes, y lo que define a las redes de tipo espacial, es que realizan la convolución aplicando algún tipo de agregación sobre el vecindario de cada nodo. El tipo de red que más interesa para este proyecto, son las redes basadas en modelos de propagación o difusión. El trabajo presentado en (Atwood y Towsley, 2016), sienta las bases para estas redes desarrollando la operación llamada *diffusion-convolution*, basada en los procesos de difusión sobre grafos.

Redes Temporales Convolucionales

Presentamos a continuación las redes temporales convolucionales (*Temporal Convolutional Networks* - TCN). Estas redes, presentadas en (Bai, Kolter, y Koltun, 2018), surgen por la necesidad de encontrar una mejor solución para los datos secuenciales. El estándar para este tipo de tareas era la aplicación de arquitecturas basadas en redes neuronales recurrentes, sin embargo, esta clase de redes sufren de ciertas limitaciones. Reconsiderando el vínculo entre el modelado secuencial y las redes recurrentes es que se construyen las redes TCN, una arquitectura basada en CNN con el objetivo de resolver problemas de modelado de secuencias y obteniendo resultados por encima de muchos modelos basados en redes recurrentes y en una variedad de tareas distintas.

Hay dos características importantes que distinguen a la TCN. En primer lugar, al momento de predecir no se utiliza información futura en relación al valor que se quiere predecir. Y en segundo lugar, la red produce una salida del mismo tamaño de la entrada. Estas dos características serán explicadas más en detalle una vez se expliquen los componentes de la arquitectura.

Como ya mencionamos, la tarea objetivo de una TCN es el modelado de

secuencias. Este se puede formalizar de la siguiente manera. Sea una secuencia de entrada $x_0, \cdots, x_T$, se desea encontrar una función $f()$, tal que genere una predicción $\hat{y_0}, \cdots, \hat{y_T} = f(x_0, \cdots, x_T)$ con la particularidad de que para predecir un valor y_t, para cualquier tiempo t, solo se deben utilizar valores observados hasta ese tiempo: $x_0, \cdots, x_t$. Dejando de lado el aspecto espacial de la tarea de predicción de tráfico y la complejidad que esto conlleva, podemos centrarnos en un nodo y reducir la tarea a un modelado de secuencias, en la que se quiere predecir velocidades a partir de lecturas observadas en un solo nodo.

Para conservar el tamaño de la entrada a lo largo de la red, TCN utiliza una red *1D fully-convolutional network* (FCN) (Long, Shelhamer, y Darrell, 2015). Luego, para mantener el principio de no generar predicciones con valores futuros, se aplican convoluciones causales. Las convoluciones causales son un tipo de convolución donde un elemento correspondiente a un tiempo t solo puede convolucionarse con elementos de tiempo t o anteriores (ver Figura 2.3).

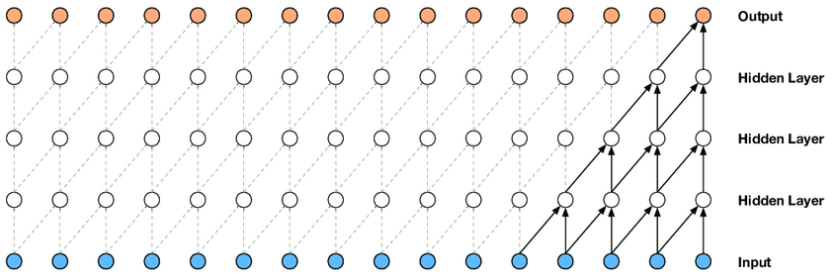


Figura 2.3: Convolución causal, tomado de WaveNet (van den Oord y cols., 2016)

El problema de las convoluciones causales es que requieren de arquitecturas altamente profundas con una gran cantidad de capas para lograr mantener un historial de gran tamaño. Para resolver esta limitante, tanto en (van den Oord y cols., 2016) como en (Wu y cols., 2019), se implementan convoluciones causales dilatadas (F. Yu y Koltun, 2016). En la convolución dilatada se utiliza un factor de dilatación para saltear valores a la hora de realizar la operación de convolución, permitiendo trabajar con un campo receptivo de crecimiento exponencial. En la Figura 2.4 se puede ver gráficamente un ejemplo de convolución dilatada donde claramente se aprecia el aumento del campo receptivo respecto de la convolución causal tradicional ilustrada en la Figura 2.3.

2.4.2. Graph WaveNet

Una vez establecidos los fundamentos necesarios para comprender la arquitectura de Graph WaveNet, entraremos en detalle sobre como está construido y que funciones cumple cada componente utilizado. En la Figura 2.5 vemos un esquema general de la arquitectura y sus principales bloques. La descripción y el análisis estará basado no solo en (Wu y cols., 2019), sino también en el

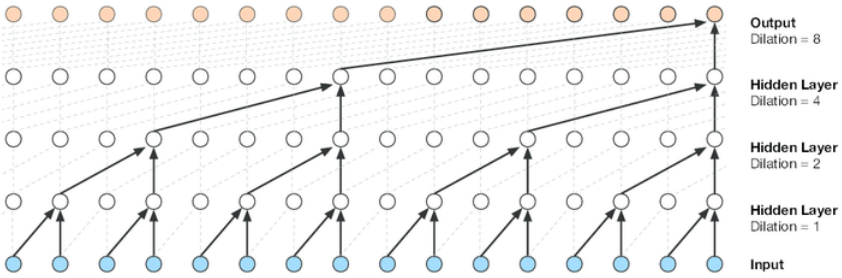


Figura 2.4: Convolución causal dilatada, tomado de WaveNet (van den Oord y cols., 2016)

desarrollo expuesto en (Beyer, Ahmad, Yang, y Rodríguez-Bocca, 2023).

Recordando la definición del problema de predicción de tráfico planteado en la Ecuación 2.1, definiremos exactamente el problema que se desea resolver con Graph WaveNet. Tenemos un conjunto de N ubicaciones, de las que se desea predecir velocidades futuras, a partir de un conjunto de mediciones pasadas. Más precisamente utilizando la velocidad como única *feature*, a partir de las últimas doce mediciones de velocidad ($T = 12$) se genera una predicción para los siguiente doce valores ($S = 12$).

La arquitectura Graph WaveNet consiste esencialmente en múltiples capas espacio-temporales, donde cada una combina una capa de convolución temporal (Gated TCN) con una capa de convolución de grafos. La entrada pasa inicialmente por una capa lineal para luego enviarse a estas capas. La idea de la implementación de múltiples capas espacio-temporales es para poder trabajar en paralelo con distintas dependencias espaciales sobre distintos horizontes temporales. Las capas inferiores trabajan con información temporal de corto plazo, mientras que las capas superiores se ocupan de las dependencias temporales de largo plazo. La salida de estas capas es propagada utilizando *skip connections* directamente a la capa de salida. La capa de salida combina la función de activación *ReLU()* junto con transformaciones lineales para construir la salida del modelo.

Capa de convolución temporal

En esta arquitectura se utiliza una capa de convolución causal dilatada (F. Yu y Koltun, 2016) como convolución temporal. Como se mencionó anteriormente, esto permite aumentar el campo receptivo de manera exponencial sobre la cantidad de capas de la arquitectura, permitiendo al modelo un mejor rendimiento al momento de trabajar sobre las dependencias temporales de largo plazo y logrando mayor eficiencia computacional. Matemáticamente, podemos definir la operación de convolución causal dilatada sobre una entrada

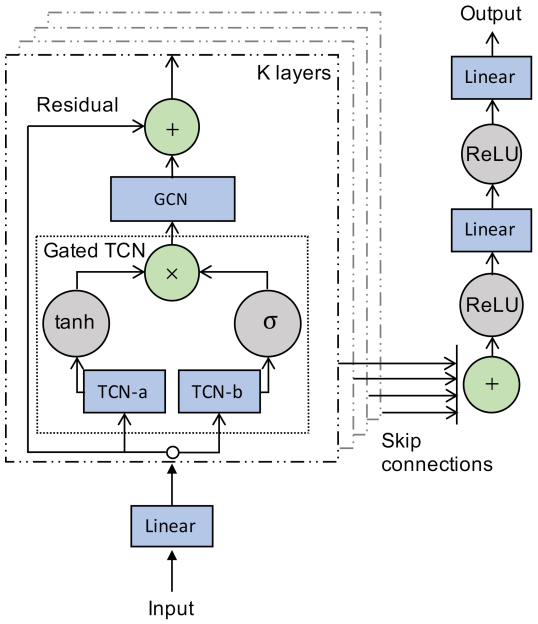


Figura 2.5: Diagrama de arquitectura Graph WaveNet. Fuente (Wu y cols., 2019).

unidimensional $x \in \mathbb{R}^T$ y un filtro $f \in \mathbb{R}^K$ en un paso t como:

$$\mathbf{x} \star \mathbf{f}(t) = \sum_{s=0}^{K-1} \mathbf{f}(s)\mathbf{x}(t - ds) \tag{2.6}$$

donde d es el factor de dilatación. Podemos definir entonces la capa de convolución temporal como:

$$\mathbf{y} = \sigma(\mathbf{\Theta} \star \mathbf{X} + b), \tag{2.7}$$

donde $\mathbf{X}$ es la entrada, σ es la función de activación, $\mathbf{\Theta}$ y b son parámetros del modelo.

Esta capa se implementa con un mecanismo de *gating*. Este mecanismo colabora en el control del flujo de información a través de las capas (Dauphin, Fan, Auli, y Grangier, 2017). La estructura final de la capa de convolución temporal con *gating* aplicado consiste de dos capas paralelas de convolución temporal (TCN-a y TCN-b). Una de las diferencias entre ambas capas es la función de activación que utilizan. TCN-a utiliza la función tangente hiperbólica, mientras que TCN-b aplica la función sigmoide. Formalizando estas ideas obtenemos la siguiente expresión para la capa:

$$\mathbf{h} = \tanh(\mathbf{\Theta}_1 \star \mathbf{X} + b_1) \odot \sigma(\mathbf{\Theta}_2 \star \mathbf{X} + b_2), \tag{2.8}$$

donde $\mathbf{X}^{N \times D \times T}$ es la entrada, $\mathbf{\Theta}_1$, $\mathbf{\Theta}_2$, b_1 y b_2 son parámetros del modelo, y $\odot$ es el producto elemento a elemento.

Capa de convolución de grafos

El otro componente clave de la arquitectura Graph WaveNet es la capa de convolución de grafos (GCN). El conjunto de sensores se encuentra modelado como un grafo dirigido $\mathcal{G} = (V, E)$, donde V es el conjunto de nodos, y E el conjunto de aristas. $\mathbf{A} \in \mathbb{R}^{N \times N}$ es la matriz de adyacencia vinculada al grafo y definida como se muestra en la Ecuación 2.2.

La capa GCN trabaja sobre este grafo para modelar las dependencias espaciales transformando y aplicando agregaciones sobre las *features* de los vecinos de cada nodo. Por lo tanto, pertenece a la clase de redes GCN de enfoque espacial.

La construcción de esta capa está basada en la convolución de difusión presentada en (Li y cols., 2018). Sean $\mathbf{X} \in \mathbb{R}^{N \times D}$ las *features* de entrada, $\mathbf{Z} \in \mathbb{R}^{N \times M}$ las señales de salida del modelo, $\mathbf{W} \in \mathbb{R}^{D \times M}$ la matriz de parámetros de la GCN, $\mathbf{P}_f = \mathbf{A}/rowsum(\mathbf{A})$ la matriz de transición hacia adelante y $\mathbf{P}_b = \mathbf{A}^T/rowsum(\mathbf{A}^T)$ la matriz de transición hacia atrás, tenemos entonces que la capa GCN queda definida como:

$$\mathbf{Z} = \sum_{k=0}^{K} \left(\mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2} \right). \tag{2.9}$$

Uno de los aspectos más novedosos propuestos en Graph WaveNet, es la implementación de una matriz de adyacencia auto-adaptativa $\tilde{\mathbf{A}}_{adp}$. Esto permite trabajar aún cuando no se conoce la estructura del grafo $\mathcal{G}$, ya que la matriz permite descubrir las dependencias espaciales ocultas. De hecho, aun cuando se dispone de la información estructural del grafo $\mathcal{G}$, los resultados muestran que incorporar la matriz de adyacencia auto-adaptativa permite al modelo descubrir nuevas dependencias espaciales. La matriz auto-adaptativa esta definida como:

$$\tilde{\mathbf{A}}_{adp} = \text{SoftMax}\left(\text{ReLU}\left(\mathbf{E}_1 \mathbf{E}_2^\intercal\right)\right),$$

donde $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{R}^{N \times c}$ son dos espacios embebidos de nodos (de dimensión c) que se aprenden de los datos y son inicializados de manera aleatoria. Cuando la estructura del grafo no está disponible, la convolución se define utilizando solamente la matriz auto-adaptativa como se ve en la Ecuación 2.10:

$$\mathbf{Z} = \sum_{k=0}^{K} \tilde{\mathbf{A}}_{adp}^k \mathbf{X} \mathbf{W}_{k3}. \tag{2.10}$$

En otro caso, cuando contamos con la estructura espacial y se desea mejorarla, la convolución queda definida como la suma de las Ecuaciones 2.9 y 2.10, resultando en una capa de convolución de grafos resumida en la Ecuación 2.11:

$$\mathbf{Z} = \sum_{k=0}^{K} \left(\mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2} + \tilde{\mathbf{A}}_{adp}^k \mathbf{X} \mathbf{W}_{k3} \right). \tag{2.11}$$

Los resultados presentados en Graph WaveNet muestran que el mejor rendimiento del modelo se da cuando se dispone de la estructura del grafo y al mismo

tiempo se utiliza la matriz auto-adaptativa. Además, también se muestra que utilizar únicamente la matriz auto-adaptativa genera resultados favorables, por lo que el modelo tiene un buen funcionamiento en escenarios en los que no se dispone de la estructura espacial.

2.5. Resumen de trabajos relacionados

Presentaremos a continuación una serie de trabajos, tanto en el área de predicción de tráfico como en el de imputación de datos de tráfico, que dan una perspectiva histórica de algunos de los desarrollos que sentaron las bases para el modelo utilizado en este proyecto y trabajos que construyeron a partir de la arquitectura seleccionada.

- *WaveNet: A Generative Model for Raw Audio* (van den Oord y cols., 2016), es una publicación fundamental, ya que si bien es un modelo enfocado en generación de audio, inspira la arquitectura utilizada en Graph WaveNet. Uno de los elementos novedosos e importantes planteados en el modelo WaveNet son las convoluciones causales dilatadas utilizadas para capturar dependencias temporales. Este elemento es utilizado en Graph WaveNet para mejorar el campo receptivo en el tratamiento de dependencias temporales de largo alcance. A grandes rasgos, la arquitectura planteada en Graph WaveNet es una adaptación de la presentada en WaveNet. La diferencia principal es la incorporación de una capa de convolución de grafos para tratar dependencias espaciales, algo que en el área de procesamiento de audio no es necesario.

- *Diffusion Convolutional Recurrent Neural Network* (DCRNN) (Li y cols., 2018). Este artículo es de gran importancia en el área de predicción de tráfico ya que es el primero en aplicar convoluciones sobre grafos, modelando la red de calles como un grafo ponderado dirigido.

  Este trabajo enfoca el modelado de dependencias espaciales vinculando el flujo de transito a un proceso de difusión, considerando la difusión hacia adelante y la difusión hacia atrás, ya que considerar ambos sentidos del tráfico colabora al momento de capturar estas dependencias. En la práctica, esto se implementa mediante una capa convolucional de difusión (*diffusion convolution layer*), truncando el proceso de difusión a uno de K pasos. Este elemento es la base de la GCN planteada en Graph WaveNet. Sin embargo, las dependencias temporales son modeladas distinto, proponen una solución basada en RNN, utilizando *gated recurrent units* (GRUs), integrando ambos componentes en una unidad llamada *Diffusion Convolutional Gated Recurrent Unit* (DCGRU).

  Otro aporte significativo, es que la construcción del *dataset* de referencia METR-LA es producto de este trabajo. Utiliza el método de núcleo Gaussiano propuesto en (Shuman, Narang, Frossard, Ortega, y Vandergheynst, 2013) para determinar los pesos de las aristas en la matriz de adyacencia

a partir de la distancia por calle entre los nodos. Este mismo *dataset* de referencia es utilizado tanto en Graph WaveNet como en múltiples publicaciones posteriores, considerándose un estándar en el área de predicción de tráfico.

- Otro artículo importante para este trabajo es *Incrementally Improving Graph WaveNet Performance on Traffic Prediction* (Shleifer, McCreery, y Chitters, 2019). Donde se proponen mejoras directamente sobre la arquitectura Graph WaveNet, que sirven como insumo para este proyecto. En particular, una de las mejoras propuestas de mayor interés es que la imputación de la media general en los datos de valor 0 mejora el rendimiento del modelo. La explicación está vinculada a que el modelo debe aprender que bajas velocidades están asociadas a un alto volumen de tráfico, pero cuando la velocidad es nula significa que no hay tráfico. Imputando con la media general obtienen no solo mejoras en las métricas, sino que una velocidad de convergencia mayor.

  Otras de las mejoras propuestas van en la línea de modificar algunos hiperparámetros como aumentar la cantidad de filtros utilizados en los bloques TCN y GCN de 32 a 40, disminuir el valor de recorte de gradientes de $L2 = 5$ a $L2 = 3$ y aplicar *learning rate decay* para reducir el valor de *learning rate* luego de cada época. Además, incorporan más *skip connections* para la salida del bloque TCN y la salida de la capa GCN para conservar gradientes más grandes. Por último, observan que entrenar modelos enfocados en predecir horizontes más cercanos potencialmente mejoran la predicción a corto plazo. Inicializando el entrenamiento de la tarea completa utilizando los pesos del modelo enfocado en predicciones de corto plazo presenta una leve mejora en el modelo final. En cuanto a resultados de métricas concretas, obtienen un valor medio de MAE (*Mean Average Error*) sobre el *dataset* METR-LA de 2.98, mejorando el 3.04 reportado en el modelo Graph WaveNet original. El valor medio de MAE es calculado siguiendo la siguiente expresión:

$$MAE_t = \frac{\sum_{i \in Sensores} |\hat{y}_i - y_i|}{|Sensores|}, \quad \text{donde } y_i \neq 0,$$

$$\overline{MAE} = \frac{1}{12} \sum_{1 \leq t \leq 12} MAE_t.$$

  Sin embargo, este artículo no solo es útil por las mejoras presentadas, sino también por los experimentos fallidos expuestos que de alguna manera permitieron enfocar nuestro proyecto para obtener resultados nuevos, evitando repetir experiencias que ya fueron reportadas como fallidas. Algunas de estas son interesantes de mencionar. En primer lugar, la incorporación de más pasos de información histórica a la predicción de cada punto no parece mejorar el resultado. Se realizaron experimentos incorporando los

últimos 75 minutos en lugar de los últimos 60 y no se observaron mejoras. De la misma manera, si bien se puede observar que el tráfico presenta un comportamiento distinto durante los fines de semana, incorporar la información del día de la semana a la predicción, mediante un escalar o una representación *one-hot*, tampoco parece afectar las métricas de manera positiva. Otras de las modificaciones que no aportaron mejoras fueron modificar el tamaño de los *embeddings* $\mathbf{E}_1$ y $\mathbf{E}_2$, utilizar *transformers* en lugar de convoluciones 1D y modificar la cantidad de capas de los bloques, entre otras.

- *A review of irregular time series data handling with gated recurrent neural networks* (Weerakody y cols., 2021). Este artículo presenta la problemática vinculada a la creciente presencia de datos irregulares como resultado de redes de sensores en distintas áreas. Con el amplio uso a nivel industrial y comercial de modelos de clasificación y predicción en base a series temporales, surge la necesidad de tratar el problema de la carencia e irregularidad de los datos de manera de obtener el mejor rendimiento posible de estos. El problema está en que la gran mayoría de estos modelos son desarrollados para trabajar sobre datos regulares, por lo tanto, es necesario desarrollar mecanismos que permitan el pasaje de datos irregulares a regulares. En este artículo se presenta una discusión y repaso sobre distintos mecanismos para intentar resolver esta problemática. Si bien el análisis está enfocado en torno a modelos basados en redes recurrentes, es útil para obtener un panorama general del problema y de potenciales soluciones para aplicar en distintas arquitecturas.

  Se revisan soluciones enfocadas en dos clases: por un lado las técnicas de imputación de datos en etapas de pre-procesamiento, y por otro lado la modificación directa sobre los algoritmos para tratar los datos faltantes durante la etapa de entrenamiento. Sobre la primer clase, se presentan distintos métodos de imputación, entre los que se encuentran reemplazo de valores, interpolación, autorregresión, *resampling* y métodos de aprendizaje automático. Las variantes más sencillas de imputación son la de reemplazo por valores estadísticos como la media, la moda y la mediana. Si bien se expone que estos métodos presentan limitaciones a la hora de conservar información temporal, en conjuntos de baja carencia de datos tienden a mostrar buenos resultados. Los modelos de aprendizaje serían una mejor solución para imputar conservando dependencias temporales, pero a un mayor costo computacional. En otra línea, se presentan modelos basados en redes recurrentes diseñados para trabajar sobre datos irregulares. En líneas generales, estos modelos cuentan con una etapa de generación de datos con el objetivo de reconstruir las series temporales previo al entrenamiento.

Capítulo 3

Metodología y datos utilizados

En este capítulo describimos los aspectos más importantes vinculados a los datos utilizados en este proyecto. Comenzado por la metodología para la construcción del *dataset* de Montevideo, y luego pasando por la etapa de análisis descriptivo para ambos *datasets* (METR-LA y MVD). Además, una descripción de alto nivel de los aspectos principales del código de implementación del modelo Graph WaveNet y el tratamiento de la métrica de evaluación seleccionada, finalizando con la fase de búsqueda de hiper-parámetros.

3.1. Pre-procesamiento de datos

En esta sección presentamos detalladamente las etapas implementadas en el pre-procesamiento de datos, partiendo desde los datos crudos pertenecientes a Montevideo, hasta el formato final compatible con la entrada del modelo, resultando en la versión final del *dataset* MVD.

Partiendo de los datos presentados en la Sección 2.3.1, el conjunto de datos inicialmente contiene 241 ubicaciones distribuidas a lo largo de Montevideo, con una concentración en torno a calles de alto volumen de tráfico como avenidas y bulevares. Los datos contienen la información de velocidad dividida por carril, en las calles que cuentan con más de un carril. Sin embargo, nuestro interés está en las observaciones a nivel global de cada nodo. Entendiendo esto, la granularidad de las observaciones a nivel de carril no sería necesaria, por lo que la primer etapa consiste en unir la información de los carriles tomando el promedio de estas medidas.

Analizando en detalle las mediciones y en particular las ubicaciones geográficas, observamos algunos nodos que por problemas de precisión numérica y redondeo de sus coordenadas, se superponen en el conjunto final de ubicaciones. Por lo tanto tomamos la decisión de eliminar estas ubicaciones problemáticas del conjunto.

Siendo este un conjunto de datos reales, debido a múltiples factores existe una cantidad importante de mediciones faltantes (representadas en los datos mediante el valor `NaN`). Este fenómeno se acentúa en algunas ubicaciones particulares, por lo tanto la siguiente etapa consiste en analizar la distribución de mediciones a partir de un histograma de cantidad de mediciones por ubicación para obtener un panorama de la cantidad de mediciones faltantes y luego tomar la decisión de descartar ubicaciones en las que falten demasiados datos o en las que existan rachas considerablemente largas de datos faltantes. A partir de lo que se observa en el histograma de la Figura 3.1, tomamos la decisión de eliminar ubicaciones con menos de 60.000 mediciones. Esta etapa redujo la cantidad de ubicaciones a 171.

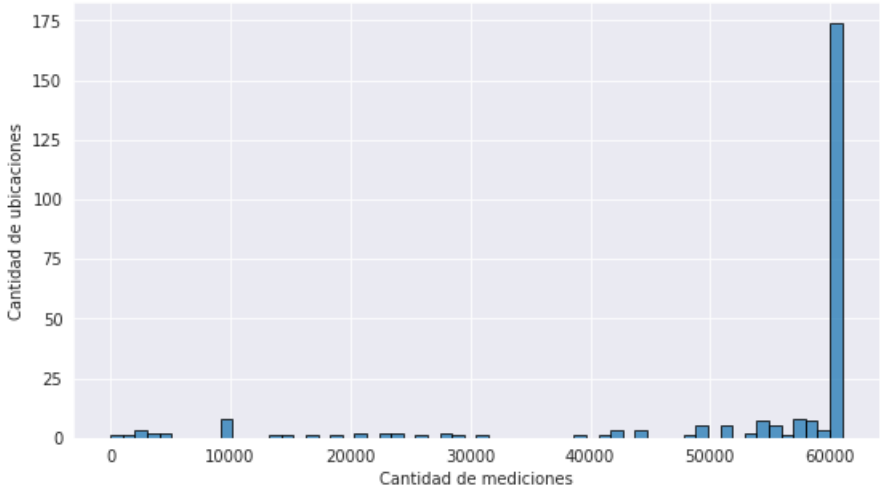


Figura 3.1: Histograma de cantidad de mediciones por ubicación.

Un detalle importante, es que hasta este momento los datos no tienen el formato que el modelo requiere como entrada. Por lo tanto, la siguiente etapa consiste en realizar una operación de pivoteo sobre los datos para obtener el formato (timestamp $\times$ ubicación) que el modelo necesita. Sin embargo, el problema que surge como producto de esta operación, es que en esta nueva disposición de los datos se generan nuevas entradas con valor `NaN` en ubicaciones donde no existen mediciones para los *timestamps* correspondientes. Para solucionar esta situación, tomamos la decisión de completar las rachas de datos inexistentes mediante interpolación lineal entre el último valor disponible y el siguiente valor disponible, siguiendo la línea de lo propuesto en (B. Yu, Yin, y Zhu, 2018). A pesar de que este método es una vía para resolver el problema, observamos que múltiples ubicaciones presentan secuencias de datos faltantes consecutivos extremadamente largas, por lo que decidimos limitar la interpolación a rachas de hasta 10 datos consecutivos, y descartar las ubicaciones que aun presentaran valores faltantes luego de la interpolación. Esta última etapa redujo a 124 la

cantidad de ubicaciones. Como resultado final, obtenemos un conjunto de 124 ubicaciones con mediciones desde enero de 2021 a julio de 2021, en el formato requerido por Graph WaveNet. Este formato se encuentra presentado a modo ilustrativo en la Figura 3.2.

Una vez filtradas las ubicaciones deseadas y ya con el conjunto final, es necesario obtener las distancias por calle entre cada par de ubicaciones para la construcción de la matriz de adyacencia. Generamos tuplas utilizando las coordenadas de cada ubicación, y sencillamente iterando sobre cada par de ubicaciones, se envían consultas a la API de *Google Maps Distance Matrix*[1] colocando una ubicación como origen y otra como destino. Es importante mencionar que para obtener un resultado más preciso, no consideramos simétrica la distancia entre dos ubicaciones, sino que la calculamos de manera independiente en ambos sentidos. Procesando las respuestas de la API almacenamos para cada consulta las coordenadas de origen, las de destino y la distancia en metros.

El último paso requerido antes de generar la matriz de adyacencia es la construcción de un índice que mantenga una correspondencia entre las ubicaciones y sus identificadores numéricos definidos de manera arbitraria. Para la construcción de la matriz de adyacencia utilizamos la función `get_adjacency_matrix()` tomada del código de DCRNN (Li y cols., 2018). Esta toma como entrada el *dataframe* de distancias con formato (origen, destino, distancia) y la lista de identificadores de sensores, y construye la matriz como se detalla en la Ecuación 2.2.

id	0	1	2	3	4	5	6	7
fecha_hora								
2021-01-01 00:00:00	13.5	0.0	36.0	10.666667	50.0	21.5	37.0	35.5
2021-01-01 00:05:00	22.0	0.0	0.0	1.666667	53.0	14.5	64.5	40.0
2021-01-01 00:10:00	0.0	0.0	49.0	1.333333	51.0	13.5	57.0	31.0

Figura 3.2: Ejemplo ilustrativo del formato de los datos: en este caso observamos las mediciones para tres *timestamps* en ocho ubicaciones pertenecientes al *dataset* MVD.

3.2. Análisis descriptivo de los datos

Dada la naturaleza de este proyecto, una parte importante de este trabajo está vinculada al análisis descriptivo de los datos. Esta etapa permite comprender los datos desde otra perspectiva, intentando echar luz sobre los patrones

[1] Distance Matrix API overview - Google for Developers `https://developers.google.com/maps/documentation/distance-matrix/overview`

y comportamientos subyacentes, favoreciendo la comprensión más allá de los números fríos y las métricas arrojadas por el modelo. Es este tipo de análisis que permite descubrir sutilezas que en algunos casos permiten explicar algunos resultados obtenidos en el entrenamiento y también permite comparar, al menos en este caso, los *datasets* utilizados y qué tienen, o no, en común.

Dado que el *dataset* METR-LA se corresponde a una colección de sensores ubicados en autopistas en la ciudad de Los Angeles, y que el *dataset* MVD toma velocidades de calles de zonas urbanas de Montevideo, es esperable notar diferencias en los patrones y las dinámicas de tráfico observadas ya que se tratan no solo de ubicaciones diferentes, sino de tipos de vías distintas.

En los datos de METR-LA, la velocidad global promedio es de 53 mph, que corresponden a 85 km/h, con una desviación estándar de 20 mph, correspondiéndose a 32 km/h. En la Figura 3.3 observamos un *boxplot* construido a partir de estos datos. Es claro que al tratarse de datos de una autopista, las velocidades que se observan son altas en promedio. Es interesante destacar que si se observan las velocidades medias de cada hora, representadas por los puntos blancos, se ve claramente que existen dos caídas en las velocidades. La primera se ve en la mañana, alrededor de las 8 horas, y la segunda se da en la tarde, entorno a las 17 horas. Ambas se corresponden a las horas pico de transito al comienzo y fin de la jornada laboral, momentos en los que también se puede observar, analizando los bigotes, que presentan una mayor variabilidad en las velocidades medidas. El resto de los horarios, si bien presentan *outliers*, muestran una concentración de datos que denota un comportamiento más estable.

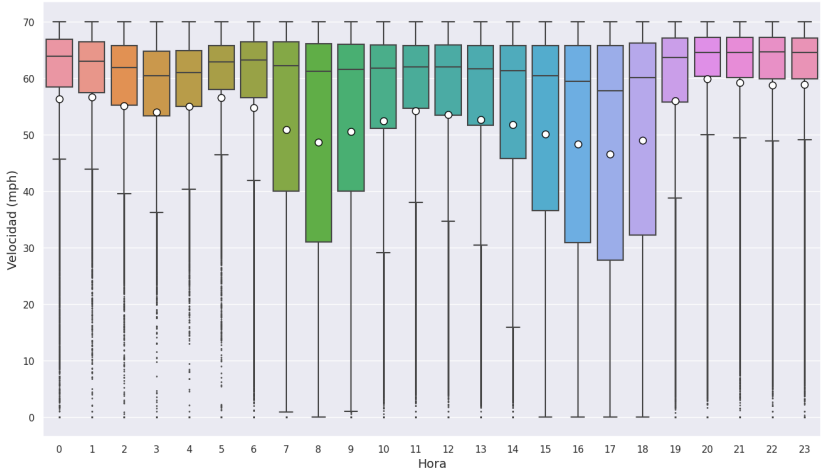


Figura 3.3: Boxplot de velocidades por hora para METR-LA.

Siguiendo la línea de esta discusión, analizamos los datos para Montevideo presentados en la Figura 3.4. En MVD, la velocidad global promedio es 30 km/h, con una desviación estándar de 16 km/h. Tratándose de un escenario conformado por calles en un contexto urbano, las velocidades observadas son notoriamente

más bajas. En este caso no se observa un descenso tan claro en las horas pico como se vio en el caso anterior, pero se ve que durante el horario de oficina la tendencia se mantiene relativamente estable siendo estos los horarios que presentan menor variabilidad. La madrugada se presenta mucho más inestable con velocidades muy bajas y muy altas, en particular en el horario entres las 2 am y las 5 am.

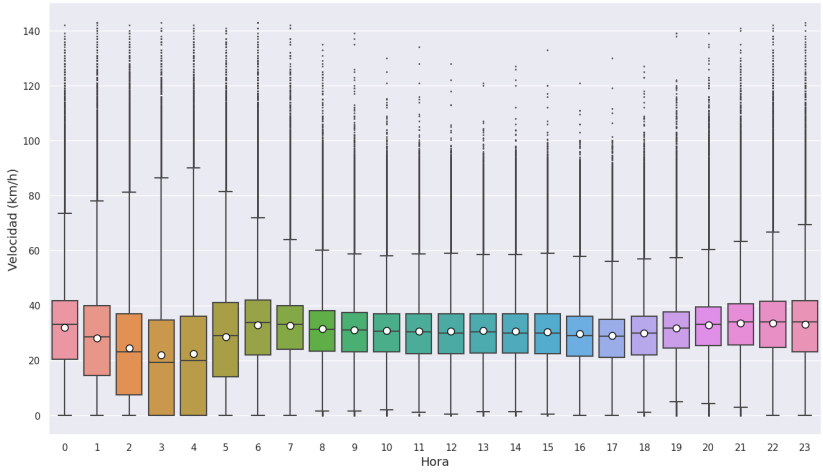
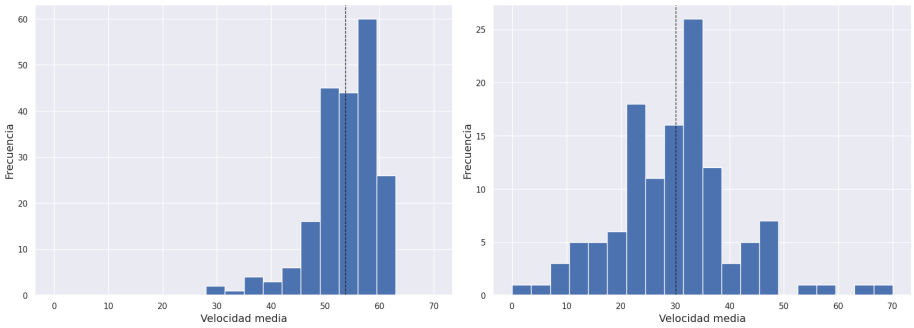


Figura 3.4: Boxplot para velocidades por hora para MVD.

Otro aspecto interesante de analizar sobre los datos tiene que ver con las velocidades observadas a nivel de ubicación. En la Figura 3.5 podemos observar la distribución de velocidades medias por ubicación para ambos escenarios. Analizando ambos histogramas vemos nuevamente que el escenario METR-LA muestra velocidades mayores. Por otro lado, la distribución en Montevideo muestra que existen algunas ubicaciones de muy bajas velocidades y algunas otras de altas velocidades, mientras que la mayor parte de las ubicaciones se encuentran entorno a la velocidad media general de todo el *dataset*. Este comportamiento indica que las ubicaciones presentan distintos patrones de tráfico, menos uniformes que en METR-LA, probablemente debido a que son zonas de la ciudad con distinto volumen de tráfico y con distintos limites de velocidad establecidos a nivel de reglamentación.

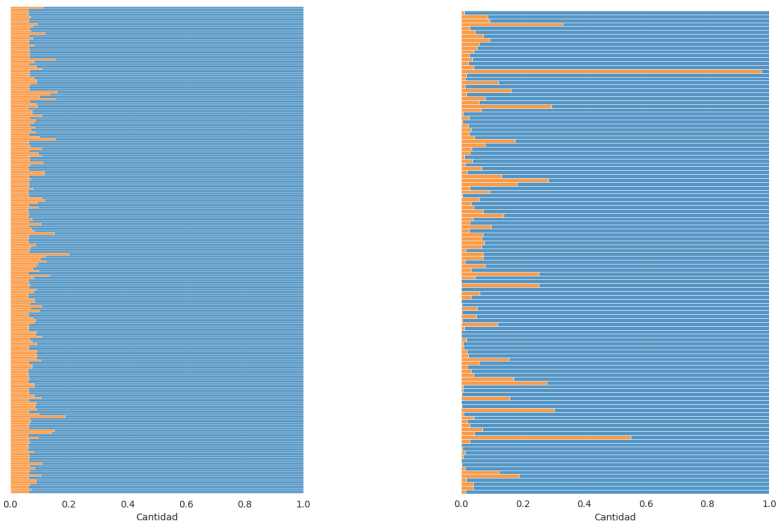
En la Figura 3.6 podemos analizar la distribución de datos faltantes para ambos *datasets*. Si bien ambos *datasets* tienen un porcentaje de carencia de datos similar, METR-LA 8 % y MVD 7 %, es interesante notar que la manera en la que están distribuidos estos datos faltantes es muy distinta. En el caso METR-LA podemos ver que la carencia de datos se da de manera más regular, donde las ubicaciones tienen un porcentaje de datos faltantes similar entre ellas. No así en el caso de MVD, donde se pueden ver ubicaciones completas o con muy pocos datos faltantes, y en el otro extremo otras ubicaciones con una gran porcentaje de carencia.



(a) Histograma de velocidades medias por ubicación en *dataset* METR-LA.

(b) Histograma de velocidades medias por ubicación en *dataset* MVD.

Figura 3.5: Distribución de velocidades medias por ubicación.



(a) Datos faltantes en *dataset* METR-LA.

(b) Datos faltantes en *dataset* MVD.

Figura 3.6: Datos faltantes para ambos *datasets*. En color naranja se puede observar el porcentaje de datos faltantes para cada ubicación.

Llevando el análisis de datos faltantes al aspecto relativo al horario, en las Figuras 3.7 y 3.8 podemos observar mapas de calor que ilustran la cantidad de datos faltantes por hora y por ubicación. Es interesante remarcar la diferencia en los patrones de datos faltantes entre ambos datasets. En el caso de METR-LA vemos que la perdida es razonablemente uniforme. Todos los horarios y ubicaciones muestran cantidades de datos faltantes similares. Sin embargo, el caso de MVD difiere en cuanto a la distribución de estos datos. Se puede observar que durante la madrugada falta la mayor parte de los datos. Esto puede estar vinculado a una baja en el flujo de tráfico durante estas horas, que por el contexto urbano no sucede en el *dataset* METR-LA. Es esperable que esto impacte de alguna manera el rendimiento del modelo en la predicción de la madrugada. Si bien los porcentajes totales de carencia de datos son similares, la distribución de estos es muy distinta entre estos escenarios.

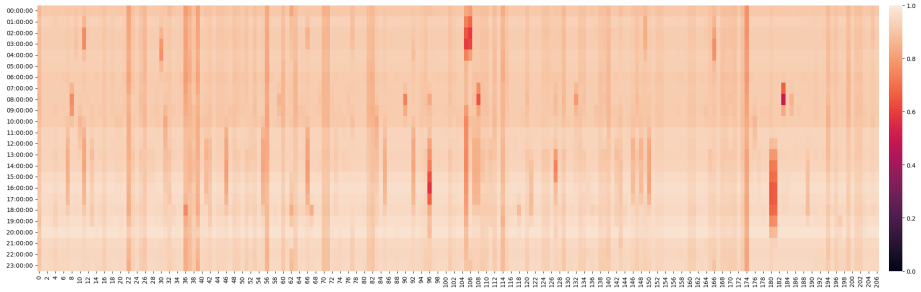


Figura 3.7: Mapa de calor de datos faltantes por hora y ubicación para METR-LA.

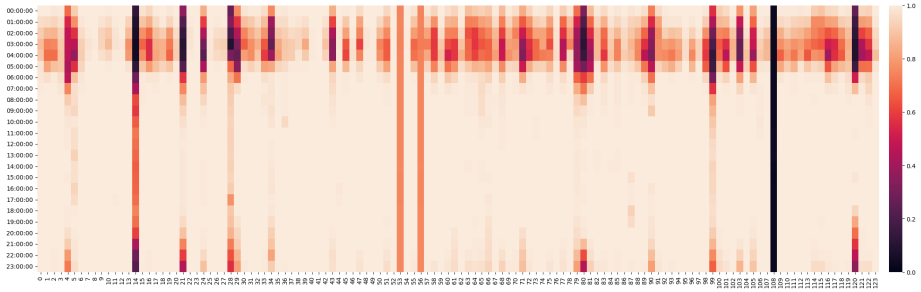


Figura 3.8: Mapa de calor de datos faltantes por hora y ubicación para MVD.

Tanto en este aspecto, como a nivel de ubicaciones, vemos que la carencia de datos se da de manera distinta en ambos *datasets*. Esto sin dudas puede estar vinculado al tipo de tráfico que se observa en cada contexto, a las costumbres y culturas distintas, así como también a temas vinculados a infraestructura, presupuesto y mantenimiento de la red de calles y sensores.

3.3. Implementación Graph WaveNet

La implementación de la arquitectura Graph WaveNet descrita a nivel teórico en la Sección 2.4.2, se encuentra realizada utilizando PyTorch[2], disponible públicamente en GitHub[3]. El código consiste de un módulo de implementación del modelo, un módulo de *testing*, uno de entrenamiento y uno de utilidades que contiene el cálculo de métricas, carga de datos y operaciones matriciales.

La separación de datos implementada por defecto es de 70 % de datos para entrenamiento, 10 % para validación y 20 % para testing, tomados respetando el orden cronológico en que se presenta la información. Este aspecto se mantiene desde la implementación del modelo DCRNN (Li y cols., 2018).

En el módulo de utilidades se implementa el cálculo de métricas del modelo. Las métricas disponibles para Graph WaveNet son MAE (*Mean Average Error*), RMSE (*Root Mean Squared Error*) y MAPE (*Mean Absolute Percentage Error*).

Si bien originalmente el modelo imprime los resultados en la salida estándar, modificamos el código para que genere dos archivos en formato CSV: uno con las métricas para cada época y otro con las métricas finales para cada horizonte predictivo.

Debido a la carga que presenta, se requiere de infraestructura de cómputo dedicada para el entrenamiento del modelo.

3.4. Métricas

Elegimos tomar la métrica *Root Mean Squared Error* (RMSE) para evaluar el rendimiento del modelo. El valor de RMSE para un horizonte t está definido como:

$$RMSE_t = \sqrt{\frac{\sum_{i \in Sensores} (\hat{y}_i - y_i)^2}{|Sensores|}}, \tag{3.1}$$

mientras que el valor de RMSE medio para los 12 horizontes de predicción esta definido por la siguiente expresión:

$$\overline{RMSE} = \frac{1}{12} \sum_{1 \leq t \leq 12} RMSE_t. \tag{3.2}$$

La implementación original de Graph WaveNet realiza el cálculo de sus métricas aplicando una máscara para evitar tomar en cuenta las posiciones en las que el valor real es 0, bajo la idea de que el valor 0 representa que ningún vehículo ha pasado por el sensor en la ventana de tiempo considerada, y no que el tráfico se encuentra detenido. Esta interpretación, bondadosa para con el rendimiento del modelo, está vinculada con que el foco de la tarea es predecir velocidades de tráfico y no la presencia y/o ausencia del mismo. Por este motivo

[2]PyTorch `https://pytorch.org/`. Accedido el 2023-06-15.

[3]`https://github.com/nnzhan/Graph-WaveNet`. Accedido el 2023-06-15.

no se espera que el modelo prediga de manera acertada los momentos en los que no hay tráfico, y se permite remover estas observaciones para que no sean tomadas en cuenta a la hora de evaluar el rendimiento.

Por lo tanto, la expresión del cálculo implementado originalmente para un horizonte t se define de la siguiente forma. Dado el conjunto de sensores con observaciones no nulas en t, $Sensores' = \{i \in Sensores / y_i \neq 0\}$ entonces:

$$RMSE_t = \sqrt{\frac{\sum_{i \in Sensores'} (\hat{y}_i - y_i)^2}{|Sensores'|}}. \tag{3.3}$$

Ya que esta métrica está implementada para poder eludir el problema de la carencia de datos, decidimos implementar un cálculo de RMSE específico, que sirva como una métrica justa al momento de evaluar distintos métodos de imputación en distintos escenarios. Dado que la predicción de cada horizonte de velocidad está realizada en base a las doce mediciones anteriores (potencialmente con valores imputados), decidimos calcular el RMSE tomando en cuenta únicamente los datos con observaciones completas (sin imputar). Esto significa tomar en consideración los puntos en los que existen sus doce mediciones anteriores, o para ponerlo de otra manera, tomar en cuenta los puntos donde sus doce mediciones anteriores no contienen ningún 0. Esto implica que para estos puntos la información utilizada para la predicción no carece de ningún dato. Consideramos que esta es la forma justa de evaluar los resultados, porque de esta forma siempre (imputando o no) evaluamos los mismos datos, los que se conocen, más allá de la imputación realizada. De otra forma, al imputar datos estaríamos ampliando artificialmente el conjunto de evaluación, y en muchos casos lo haríamos de una forma que incluya datos promedio, más fáciles de predecir, y por tanto, distorsionaríamos injustamente nuestra evaluación.

3.5. Búsqueda de hiper-parámetros

La búsqueda de hiper-parámetros es una etapa clave en una investigación vinculada al área de aprendizaje automático. Consiste en intentar obtener valores para algunos hiper-parámetros del modelo que permitan optimizar el proceso de aprendizaje para obtener el mejor rendimiento posible en las condiciones dadas. Estos hiper-parámetros, contrario a los parámetros del modelo cuyos valores son aprendidos durante el entrenamiento, son aquellos que toman valores de entrada configurables al inicio del entrenamiento y definen aspectos estructurales que impactan sobre el proceso de entrenamiento y por lo tanto en las predicciones finales del modelo.

Algunos de los hiper-parámetros más importantes del modelo Graph WaveNet son el valor de *learning rate*, *dropout*, recorte de gradientes, cantidad de filtros, cantidad de capas, coeficientes de dilatación y otros tantos más.

Sin embargo, la búsqueda de hiper-parámetros óptimos presenta un alto costo computacional debido a la cantidad de hiper-parámetros. Por este motivo

es que en el contexto de este proyecto decidimos realizar una búsqueda de hiper-parámetros de cuadricula (*grid search*) limitada a dos de ellos: *learning rate* y *dropout*.

Este tipo de búsqueda consiste en plantear series de valores para algunos hiper-parámetros y evaluar todas las posibles combinaciones de valores considerados. La ventaja de este tipo de búsqueda es que eligiendo la cantidad de hiper-parámetros y valores adecuados hace que esta etapa sea más manejable con los recursos de cómputo disponibles. En este caso decidimos tomar los valores 0.01, 0.001 y 0.0001 para el *learning rate* y 0.2, 0.3 y 0.4 para el valor de *dropout*, entrenando sobre el *dataset* MVD original. Consideramos que no fue necesario realizar esta experiencia sobre el *dateset* METR-LA, ya que al ser este uno de los datasets de referencia, los valores de hiper-parámetros por defecto en la publicación están seleccionados para este mismo conjunto de datos.

	Learning rate		
Dropout	0.01	0.001	0.0001
0.2	6.525	6.529	6.528
0.3	6.479	**6.467**	6.489
0.4	6.487	6.483	6.491

Cuadro 3.1: hiper-parámetros

En el Cuadro 3.1 presentamos los resultados obtenidos. Observando los valores obtenidos de la búsqueda por cuadricula, tenemos que los valores óptimos para los conjuntos considerados son 0.001 para *learning rate* y 0.3 para *dropout*. Estos valores son los tomados por defecto en el modelo, por lo tanto, no fue necesario entrenar con otros valores de hiper-parámetros.

Capítulo 4

Experimentación

El eje principal de este trabajo es determinar un método de imputación de datos de tráfico de bajo costo computacional integrable a la etapa de preprocesamiento, que mejore el rendimiento predictivo del modelo Graph WaveNet. Por lo tanto, la experimentación es una etapa clave para llegar al objetivo. En este capítulo detallamos los experimentos realizados y analizamos los resultados obtenidos. En la Sección 4.1 comentamos el primer experimento relacionado a la respuesta del modelo en distintos escenarios de carencia de datos para estudiar su sensibilidad. Continuamos en la Sección 4.2 con el segundo experimento, elemento medular en este desarrollo, en el que proponemos distintos métodos de imputación de datos y se analiza el desempeño en distintos escenarios construidos a partir del *dataset* de referencia METR-LA. Por último, en la Sección 4.3, aplicamos al *dataset* real MVD los métodos analizados en la fase anterior y evaluamos los resultados.

Detallamos en el Cuadro 4.1 la infraestructura utilizada para el desarrollo de estos experimentos en Google Colab[1] y el tiempo de entrenamiento del modelo para cada GPU utilizando el dataset METR-LA de referencia. Utilizamos esta infraestructura en todas las etapas de experimentación: construcción de escenarios, entrenamiento y evaluación del modelo, y cálculo de métricas.

La configuración experimental del modelo utilizada para todos los experimentos planteados fue la misma que se propone en (Wu y cols., 2019). Se usan ocho capas Graph WaveNet con una secuencia de factores de dilatación 1, 2, 1, 2, 1, 2, 1, 2. La capa de convolución de grafos es la modelada por la Ecuación 2.11, con un valor de difusión $K = 2$. Los espacios embebidos de nodos se construyen con dimensión $c = 10$ y se inicializan de manera aleatoria. En base a nuestra búsqueda de hiper-parámetros (presentada en la Sección 3.5), se utiliza *Adam* (Kingma y Ba, 2015) como optimizador con un valor de *learning rate* de 0.001 y se aplica *dropout* con probabilidad 0.3. Como ya mencionamos anteriormente, la métrica en la que nos enfocamos es RMSE.

[1] Colaboratory - Google Colab `https://colab.research.google.com/`. Accedido el 2023-06-15.

Infraestructura	
RAM 16 GB	
Intel(R) Xeon(R) CPU @ 2.20GHz	
NVIDIA A100	(40 minutos aprox.)
NVIDIA P100	(90 minutos aprox.)
NVIDIA T4	(140 minutos aprox.)
NVIDIA K80	(350 minutos aprox.)

Cuadro 4.1: Infra

4.1. Análisis de sensibilidad frente a la carencia de datos

El objetivo de este experimento es estudiar el impacto que tiene la carencia de datos sobre las métricas del modelo Graph WaveNet. Para esto planteamos una serie de escenarios generados artificialmente a partir del *dataset* de referencia, generando variantes donde se modifica la cantidad de datos faltantes. Esta experiencia sirve como una primera aproximación a la evaluación de la bondad del modelo frente a escenarios donde existen ciertos niveles de datos faltantes previo a intentar utilizar cualquier método de imputación para mejorar el rendimiento.

En primera instancia es necesario obtener conjuntos de datos con distintas cantidades de datos faltantes. En lugar de buscar distintos *datasets* de tráfico reales con distintos porcentajes de carencia, decidimos generar cuatro variantes de manera artificial partiendo de la matriz de velocidades del *dataset* de referencia METR-LA. La construcción es realizada eliminando datos de manera aleatoria hasta alcanzar el porcentaje de pérdida deseado. Los porcentajes de carencia de datos de estas variantes son 10 %, 20 %, 30 % y 40 %.

Inicialmente, la manera elegida para eliminar datos fue colocando ceros de manera aleatoria a lo largo de todo el dataset. Revisando la implementación del modelo más en detalle hallamos que el cálculo de las métricas, en particular el cálculo de RMSE, está implementado de manera que ignora las predicciones en las ubicaciones en las que el valor original es cero, como se explica en la Sección 3.4. Es decir, el impacto de los ceros en el *dataset* es equivalente a reducir la muestra de datos, ya que estos no se toman en consideración en el cálculo de las métricas. En el Cuadro 4.2 podemos observar los valores obtenidos para las tandas de matrices con 0. Observando los valores obtenidos se ve que *a priori* aumentar la cantidad de datos faltantes parece afectar el modelo mínimamente.

Nuestra conclusión preliminar es entonces: si bien en el experimento se observa cierto grado de robustez en los resultados, lógicamente al aumentar la cantidad de datos faltantes se observa un deterioro en las métricas. Si bien este deterioro es bajo, tiene sentido plantear una solución de imputación de datos en este tipo de *datasets* para lograr aprovechar al máximo y explotar los datos con el objetivo de obtener una mejor performance del modelo. Además, esta con-

Datos	$S = 3$ **15 min**	$S = 6$ **30 min**	$S = 12$ **60 min**
METR-LA 10 %	5.185	6.184	7.241
METR-LA 20 %	5.310	6.321	7.408
METR-LA 30 %	5.309	6.258	7.302
METR-LA 40 %	5.438	6.345	7.303

Cuadro 4.2: RMSE de las matrices con 0 imputado. Se puede observar que el deterioro de la métrica a medida que aumenta la cantidad de datos faltantes es mínimo.

clusión se encuentra alineada con los resultados expuestos en (Shleifer y cols., 2019), donde se concluye que utilizar la velocidad media general como valor de imputación para los datos faltantes produce una mejora moderada en las métricas del modelo.

Como ya mencionamos anteriormente, las soluciones de imputación consideradas en los siguientes experimentos son métodos de bajo costo computacional integrables en las etapas de procesamiento de datos.

4.2. Análisis de métodos de imputación en *dataset* de referencia

Partiendo de cuatro escenarios distintos de carencia de datos, el objetivo de este experimento es evaluar el rendimiento del modelo bajo una serie de propuestas de imputación de datos. Utilizando el *dataset* METR-LA como base con un 8 % de carencia de datos, construimos cuatro escenarios de pérdida distintos. Siguiendo la misma línea que se comenta en la Sección 4.1 anterior, eliminamos datos de manera aleatoria hasta obtener el porcentaje de carencia de datos deseado. Los escenarios planteados son de 20 %, 30 %, 40 % y 75 % de datos faltantes. Incorporamos un escenario de 75 % de carencia de datos para evaluar el modelo en un caso más extremo.

Apegándonos a la idea de plantear métodos de imputación de bajo costo computacional, elegimos evaluar cinco métodos distintos. Basándonos en los resultados presentados en (Shleifer y cols., 2019), decidimos evaluar la imputación utilizando la media general (M.G) del *dataset* como la alternativa más sencilla. Buscando aprovechar la potencial localidad geográfica del comportamiento del tráfico también evaluamos la imputación de la media de cada ubicación (M.U.) y en una línea similar, buscamos lo mismo a nivel temporal imputando la media por hora (M.H.). Para intentar aumentar la granularidad de los métodos, utilizando estas últimas dos ideas planteamos la imputación de la media por hora de cada ubicación (M.U-H). Por último, aplicamos el método de imputación *forward-fill* (F), para reconstruir la señal del nodo a partir del último dato disponible sabiendo que existe cierta regularidad observando entornos pequeños en

los datos.

Para la evaluación de los modelos es importante mencionar dos aspectos. En primer lugar, entrenamos los modelos utilizando los *datasets* con los datos ya imputados para cada método. Una vez generado el modelo, este fue evaluado sobre el *dataset* original, en este caso METR-LA, sin ningún tipo de modificación para poder comparar la predicción exclusivamente en los valores reales. Y en segundo lugar, el cálculo de RMSE fue realizado de acuerdo a como se explica en la Sección 3.4, tomando en consideración únicamente las observaciones sin datos faltantes en el *dataset* original, para evaluar todas las alternativas de manera razonablemente justa.

En el Cuadro 4.3 se pueden observar las métricas calculadas para cada método en los distintos escenarios en horizontes de predicción de 15, 30 y 60 minutos, además del promedio calculado para los doce horizontes de predicción.

Datos		**S.I.**	**M.G.**	**M.U.**	**M.H.**	**M.U-H**	**F**
METR-LA Original	15 min	5.05	6.30	6.21	5.04	6.08	6.13
	30 min	6.06	8.00	7.89	6.05	7.64	7.74
	60 min	7.16	9.63	9.62	7.21	9.04	9.25
	Promedio	**5.93**	7.75	7.69	5.95	7.37	7.49
METR-LA 20 %	15 min	11.57	6.21	6.41	6.14	6.22	6.08
	30 min	11.85	7.93	8.75	7.89	8.06	7.72
	60 min	12.18	11.68	12.34	11.31	10.99	9.29
	Promedio	11.83	8.26	8.88	8.19	8.18	**7.48**
METR-LA 30 %	15 min	14.10	7.24	7.52	7.93	7.50	6.20
	30 min	14.63	9.11	10.71	9.45	9.73	7.82
	60 min	15.31	11.73	15.04	11.58	12.87	9.33
	Promedio	14.65	9.10	10.77	9.75	9.91	**7.56**
METR-LA 40 %	15 min	16.88	10.28	11.54	11.50	11.55	6.13
	30 min	17.45	12.16	13.57	14.23	13.17	7.77
	60 min	18.36	14.01	15.68	16.94	14.76	9.29
	Promedio	17.52	11.84	13.17	13.52	14.02	**7.52**
METR-LA 75 %	15 min	26.27	18.05	16.50	17.98	16.99	6.25
	30 min	26.45	18.05	16.66	18.00	17.10	7.32
	60 min	27.96	18.05	17.03	17.99	17.41	9.13
	Promedio	26.77	18.05	16.70	17.99	17.13	**7.33**

Cuadro 4.3: Métricas para METR-LA con cálculo de RMSE para observaciones completas. La predicción de 15 minutos corresponde a $S = 3$, la de 30 minutos a $S = 6$ y la de 60 minutos a $S = 12$, siendo este el último horizonte de predicción. El promedio está calculado sobre las métricas de los doce horizontes de predicción.

Por otro lado, ya que tanto en (Wu y cols., 2019) como en (Shleifer y cols.,

2019) naturalmente se presentan resultados utilizando las métricas originales de Graph WaveNet (cálculo de RMSE aplicando una mascara para ignorar predicciones donde el valor original es 0), en la Cuadro 4.4 observamos los resultados de este experimento utilizando estas métricas para analizar el comportamiento del modelo con los cálculos oficiales de las publicaciones.

Datos		**S.I.**	**M.G.**	**M.U.**	**M.H.**	**M.U-H**	**F**
METR-LA Original	15 min	5.16	8.17	8.27	5.15	7.91	8.26
	30 min	6.20	10.17	10.45	6.20	9.74	10.55
	60 min	7.31	11.81	12.36	7.36	11.05	12.89
	Promedio	**6.07**	9.82	10.11	6.08	9.36	10.26
METR-LA 20 %	15 min	11.39	8.04	8.19	7.94	7.97	8.32
	30 min	11.73	10.07	10.76	10.03	10.00	10.64
	60 min	12.17	13.56	14.30	13.28	12.66	13.06
	Promedio	11.72	10.20	10.76	10.05	**9.97**	10.37
METR-LA 30 %	15 min	13.93	8.81	9.09	9.27	9.23	8.32
	30 min	14.48	10.89	12.41	11.04	11.62	10.61
	60 min	15.17	13.37	16.54	13.01	14.23	12.94
	Promedio	14.50	10.76	12.36	11.16	11.58	**10.43**
METR-LA 40 %	15 min	16.72	11.25	12.31	12.18	12.38	8.33
	30 min	17.17	13.09	14.33	14.83	14.14	10.65
	60 min	17.89	14.80	16.35	17.42	15.68	12.96
	Promedio	17.22	12.74	13.92	14.13	14.78	**10.32**
METR-LA 75 %	15 min	25.80	18.07	16.70	18.01	17.20	8.67
	30 min	25.87	18.06	16.91	18.02	17.34	10.38
	60 min	27.19	18.06	17.27	18.03	17.60	12.76
	Promedio	26.18	18.07	16.93	18.02	17.36	**10.27**

Cuadro 4.4: Métricas originales del modelo Graph WaveNet sobre METR-LA. La predicción de 15 minutos corresponde a $S = 3$, la de 30 minutos a $S = 6$ y la de 60 minutos a $S = 12$, siendo este el último horizonte de predicción. El promedio está calculado sobre las métricas de los doce horizontes de predicción.

En la Figura 4.1 observamos gráficamente la comparación de cada uno de los métodos de imputación elegidos en cada uno de los escenarios generados tomando las métricas directamente del modelo. En primer lugar, aquí se puede notar que salvo para el caso del *dataset* original, en general los métodos de imputación mejoran el rendimiento del modelo en distinta medida de acuerdo a la cantidad de datos faltantes. Dentro de las distintas alternativas de imputación planteadas, a medida que la carencia de datos se hace mayor, el método *forward-fill* (ffill) es el que presenta mejor rendimiento.

Complementando lo observado anteriormente, en la Figura 4.2 podemos analizar más en detalle el comportamiento de cada método enfocados en un nodo

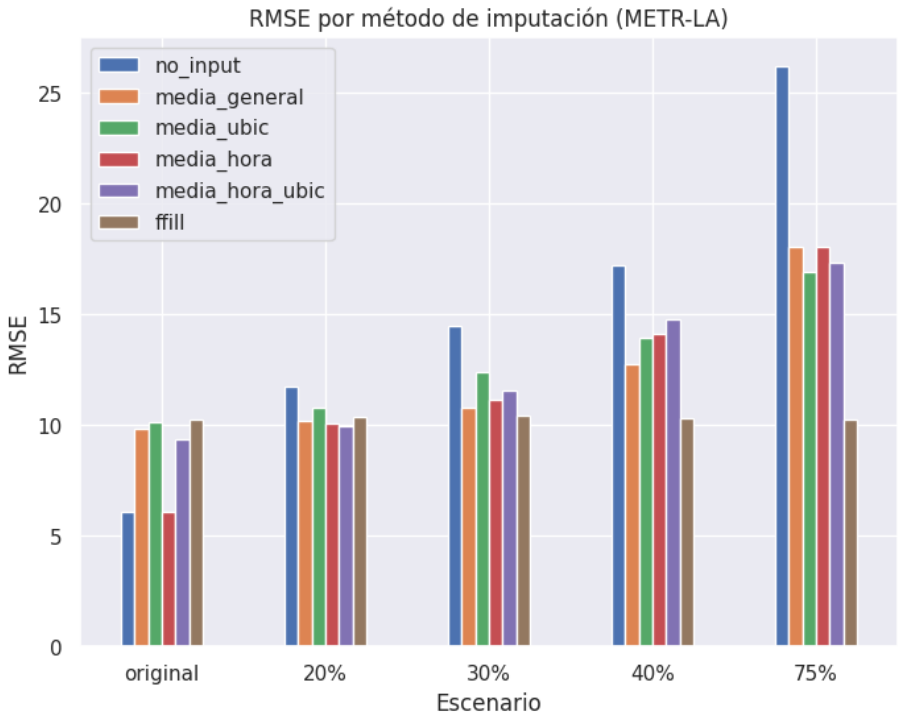


Figura 4.1: RMSE calculado por Graph WaveNet para cada método de imputación sobre todos los escenarios planteados para METR-LA

específico y reducido a una fecha particular. En esta oportunidad se tomó el nodo con *id* 2 del *dataset* METR-LA. En la Figura 4.2a vemos la predicción en el caso del *dataset* original. El rendimiento del modelo sin imputar muestra ser superior al resto, especialmente al momento de predecir la caída de velocidad que se presenta a partir de la hora 6 hasta aproximadamente la hora 10 probablemente debida a un alto flujo de tránsito, donde los otros modelos presentan un mayor error prediciendo velocidades menores a las reales.

Sin embargo, esta ventaja del modelo sin imputar desaparece ya en el escenario de 20 % de carencia que se observa en la Figura 4.2b. Aquí podemos ver que esta alternativa falla de manera considerable en la predicción de la caída de velocidad. No solo falla en esta situación, sino que más adelante se genera una predicción de una leve reducción de velocidad en torno a la hora 15 que no se observa de ninguna manera en las mediciones reales. En líneas generales los métodos de imputación mantienen un rendimiento parejo entre ellos, aunque se puede observar que en las horas anteriores y posteriores a la hora pico de la mañana se generan predicciones con mayor ruido que en el primer escenario.

En el escenario con 30 % de carencia observado en la Figura 4.2c, el comportamiento predictivo sigue un patrón similar al anterior. Como es de esperar,

todos los métodos presentan un deterioro en sus predicciones, pero en este punto ya se comienza a observar cierta estabilidad en el método *ffill* por sobre los demás. Es en el caso de 40 %, ilustrado en la Figura 4.2d, donde se da un quiebre más observable en las predicciones. Aquí el método sin imputar falla completamente en predecir la caída de velocidad, y ya se puede observar como el resto de los métodos generan predicciones alejadas de la realidad. Por ejemplo la imputación por ubicación, si bien logra predecir la bajada de velocidad, falla en las predicciones anteriores y posteriores, generando valores muy por debajo de la velocidad real. Nuevamente en este caso el método *ffill* se muestra razonablemente estable siguiendo el comportamiento de la velocidad real y aumentando la brecha de rendimiento frente al resto de los métodos.

Por último, en la Figura 4.2e, observamos el caso más extremo con un 75 % de carencia de datos. Como ya se observaba en las métricas, los métodos presentan sus comportamientos más extremos. La imputación por media general produce una predicción de velocidad media constante. Los métodos en los que interviene información específica sobre la ubicación muestran que en la caída de velocidad intentan predecir una caída, pero moderada y en líneas generales muy imprecisa. Lo destacable de este escenario es que el método *ffill* no solo continua mostrándose estable en la predicción, sino que si se observa en detalle se puede ver que funciona mejor que en los casos anteriores. Esto se ve claramente en la predicción de la caída, que en los casos anteriores si bien seguía el comportamiento con algunos errores, en este escenario la predicción de este evento puntual se ve notoriamente más ajustada.

Para concluir, de acuerdo a lo observado podemos decir que en general, a partir de cierto volumen de datos faltantes, la aplicación de un método de imputación es beneficiosa para el rendimiento del modelo. Si bien en el caso del *dataset* original no se observa una mejora en ninguno de los métodos de imputación, en todos los otros escenarios se mejora de manera sustancial el resultado. Teniendo en cuenta que los métodos elegidos para esta experiencia son métodos estadísticos sencillos y de bajo costo, sería esperable que la aplicación de técnicas más complejas de imputación generen resultados aún mejores que los obtenidos.

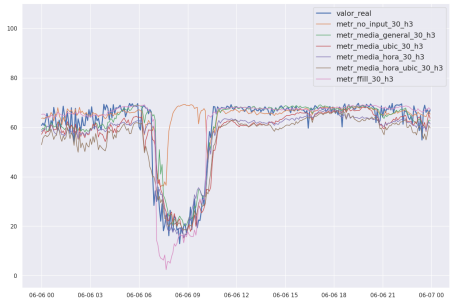
4.3. Análisis de métodos de imputación en *dataset* real

En este experimento final aplicamos las técnicas estudiadas en la Sección 4.2 a nuestro caso real. El objetivo de este experimento es estudiar si los resultados obtenidos en el *dataset* de referencia se sostienen en el *dataset* construido para Montevideo y de esta manera dar robustez a los resultados obtenidos. Siguiendo la misma metodología, construimos nuevamente cuatro escenarios de pérdida distintos, pero esta vez basándonos en el *dataset* MVD.

Los métodos de imputación evaluados son los mismos cinco presentados en el experimento anterior: media general, media por ubicación, media por hora,

(a) Predicción METR-LA original

(b) Predicción METR-LA 20 %



(c) Predicción METR-LA 30 %

(d) Predicción METR-LA 40 %

(e) Predicción METR-LA 75 %

Figura 4.2: Visualización de predicciones para cada método de imputación en los escenarios planteados para el *dataset* METR-LA en el nodo con id 2.

media por hora de cada ubicación y *forward-fill*. Las variantes de carencia generadas también son de 20 %, 30 %, 40 % y 75 %, partiendo del *dataset* original con un 7 % de datos faltantes y generadas de la misma manera que se presenta en la Sección 4.2.

Presentamos a continuación las métricas obtenidas para cada escenario. En el Cuadro 4.5 observamos los resultados para los horizontes de predicción de 15, 30 y 60 minutos, mientras que en el Cuadro 4.6 tenemos las métricas obtenidas directamente de la salida de Graph WaveNet.

Datos		**S.I.**	**M.G.**	**M.U.**	**M.H.**	**M.U-H**	**F**
MVD Original	15 min	6.31	6.62	6.50	6.59	6.41	6.48
	30 min	6.50	6.89	6.74	6.87	6.63	6.71
	60 min	6.71	7.23	7.04	7.21	6.88	6.96
	Promedio	**6.47**	6.85	6.68	6.82	6.60	6.67
MVD 20 %	15 min	7.06	6.75	6.67	6.74	6.62	6.54
	30 min	7.36	7.00	6.90	7.02	6.81	6.79
	60 min	7.70	7.32	7.14	7.42	7.03	7.02
	Promedio	7.32	6.97	6.86	7.01	6.78	**6.75**
MVD 30 %	15 min	8.39	7.35	6.96	7.50	6.80	6.54
	30 min	8.65	7.57	7.12	7.82	6.97	6.79
	60 min	9.07	7.85	7.32	8.26	7.16	7.03
	Promedio	8.53	7.54	7.09	7.80	6.94	**6.79**
MVD 40 %	15 min	10.40	9.58	8.28	9.31	7.98	6.59
	30 min	10.75	9.73	8.29	9.52	8.10	6.87
	60 min	11.25	9.92	8.29	9.80	8.30	7.13
	Promedio	10.75	9.71	8.27	9.52	8.11	**6.92**
MVD 75 %	15 min	18.87	16.60	14.05	16.77	14.09	8.15
	30 min	19.46	16.60	14.12	16.71	14.12	7.26
	60 min	20.20	16.61	14.34	16.58	14.22	7.20
	Promedio	19.47	16.61	14.15	16.70	14.13	**7.43**

Cuadro 4.5: Métricas para MVD con cálculo de RMSE para observaciones completas. La predicción de 15 minutos corresponde a $S = 3$, la de 30 minutos a $S = 6$ y la de 60 minutos a $S = 12$, siendo este el último horizonte de predicción. El promedio está calculado sobre las métricas de los doce horizontes de predicción.

Siguiendo la idea del experimento anterior, en la Figura 4.3 podemos analizar gráficamente el rendimiento de cada método de imputación sobre las variantes generadas a partir del *dataset* MVD. Observando la gráfica se puede notar que si bien los valores de las métricas son distintos, dado a que esto se realiza sobre otro *dataset*, el patrón de rendimiento de los métodos es muy similar a lo obtenido en el experimento anterior. Nuevamente, en el escenario del *dataset* original todos

Datos		S.I.	M.G.	M.U.	M.H.	M.U-H	F
MVD Original	15 min	7.30	7.87	7.71	7.84	7.92	7.75
	30 min	7.46	8.10	7.89	8.08	8.06	7.89
	60 min	7.63	8.40	8.12	8.39	8.23	8.10
	Promedio	**7.42**	8.08	7.87	8.06	8.04	7.90
MVD 20 %	15 min	8.25	8.24	8.05	8.48	8.40	7.81
	30 min	8.45	8.46	8.22	8.74	8.54	7.95
	60 min	8.72	8.71	8.39	9.08	8.65	8.10
	Promedio	8.39	8.43	8.19	8.72	8.51	**7.99**
MVD 30 %	15 min	10.21	8.96	8.40	9.52	8.62	7.85
	30 min	10.26	9.17	8.52	9.82	8.69	7.95
	60 min	10.44	9.41	8.64	10.13	8.73	8.10
	Promedio	10.28	9.14	8.49	9.77	8.66	**8.05**
MVD 40 %	15 min	11.94	11.09	9.58	11.22	9.81	7.96
	30 min	12.13	11.22	9.60	11.40	9.90	8.04
	60 min	12.44	11.36	9.57	11.53	10.01	8.23
	Promedio	12.14	11.20	9.57	11.36	9.90	**8.25**
MVD 75 %	15 min	20.21	16.85	14.82	17.07	14.81	10.66
	30 min	20.72	16.85	14.88	17.01	14.83	9.18
	60 min	21.41	16.85	15.00	16.87	14.82	8.63
	Promedio	20.76	16.85	14.89	17.00	14.82	**9.38**

Cuadro 4.6: Métricas originales del modelo Graph WaveNet sobre MVD. La predicción de 15 minutos corresponde a $S = 3$, la de 30 minutos a $S = 6$ y la de 60 minutos a $S = 12$, siendo este el último horizonte de predicción. El promedio está calculado sobre las métricas de los doce horizontes de predicción.

los métodos de imputación tienen un rendimiento por debajo de la opción de no imputar datos. Luego, a medida que aumenta la carencia de datos vemos que a gran escala los métodos de imputación comienzan a tomar ventaja sobre la alternativa de no imputar datos, presentándose una vez más el método *ffill* con una estabilidad superior al resto y un rendimiento destacable en particular en el caso más extremo de un 75 % de faltante de datos.

En la Figura 4.4 se presentan las predicciones tomadas para el nodo ubicado en Bulevar Artigas entre 21 de Setiembre y Bulevar España el viernes 9 de julio de 2021. Comenzando con la predicción sobre el *dataset* original, ilustrada en la Figura 4.4a se puede ver que en general todos los métodos, incluida la alternativa de no imputación, tienen un rendimiento similar. Todos siguen la línea del comportamiento del tráfico pero también vemos que son incapaces de capturar las sutilezas vinculadas al ruido de la señal. Esto se ve también a nivel de métricas ya que la diferencia del RMSE medio de todos los métodos en este escenario es bastante mínima como se puede ver en el Cuadro 4.6.

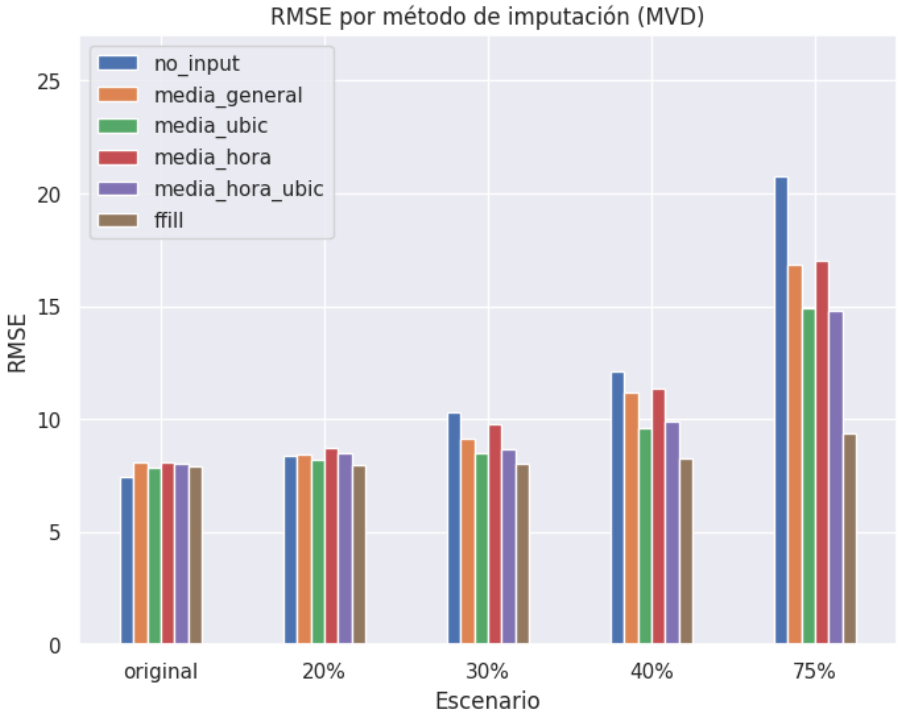


Figura 4.3: RMSE calculado por Graph WaveNet para cada método de imputación sobre todos los escenarios planteados para MVD

Algo muy similar sucede en caso de 20 % de carencia. Como se puede ver en la Figura 4.4b, los métodos logran capturar nuevamente la tendencia de las velocidades, donde en este caso se puede observar una dificultad para el método sin imputar en capturar la inestabilidad de la madrugada. Si bien de acuerdo a las métricas, el método *ffill* es el que muestra mejor rendimiento, la diferencia sobre el resto continua siendo mínima.

Ya en el escenario de 30 % de datos faltantes, algunos métodos comienzan a tener mayor dificultad en la predicción. Se puede ver claramente que el ruido observado en la madrugada entre la hora 3 y 6, genera desviaciones en las predicción del método de media general, media por ubicación y media por hora y ubicación. En este punto se comienza a ver cierta robustez del método *ffill*, como ya se observó con el *dataset* METR-LA. Luego, sobre el siguiente escenario la Figura 4.4d, la imprecisión de los métodos se acentúa, salvo en el caso de *ffill* que continúa siguiendo la tendencia del tráfico razonablemente bien y no parece ser afectado por las fluctuaciones de la madrugada.

Por último, en la Figura 4.4e, los métodos presentan una cantidad muy alta de errores (al igual que en el caso del *dataset* METR-LA). Se observa nuevamente que el método de media general presenta una predicción constante. Sin embargo,

el método *ffill*, como ya se analizó antes, presenta una predicción razonablemente buena y un nivel de robustez similar al observado en el experimento anterior.

Nuestra conclusión del experimento es: teniendo en cuenta que estructuralmente este experimento es igual al planteado en la Sección 4.2, los resultados obtenidos reflejan consistencia en la aplicación de los métodos de imputación. En ambos casos se observa que a medida que el porcentaje de carencia aumenta, es cada vez más beneficioso utilizar un método para imputar datos previo al entrenamiento del modelo, donde en particular el método *forward-fill* aparenta ser la mejor alternativa entre las planteadas.

4.4. Discusión de resultados

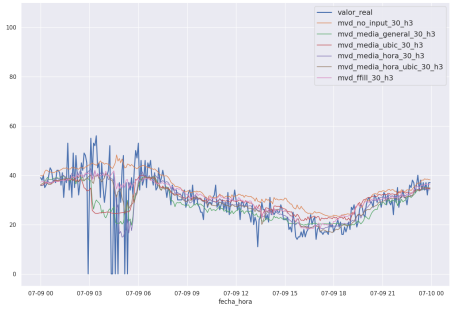
Tanto en la Sección 4.2 como en la Sección 4.3 concluimos, en base a los resultados obtenidos, que no solo es beneficioso implementar un método de imputación para datos faltantes a partir de cierto nivel de carencia, sino que también la técnica que presenta mayor robustez es el método *forward-fill* planteado. Sin embargo, es pertinente realizar una discusión basada en estos resultados obtenidos, con el objetivo de reflexionar al menos de manera superficial, sobre si la forma en que se diseñaron los experimentos puede haber afectado estos resultados.

Una de las debilidades de los experimentos realizados es que se generó una única instancia para cada escenario tratado. El motivo es que, limitados por la capacidad de cómputo, se decidió enfatizar la evaluación intentando asegurar cierta robustez del entrenamiento frente a esa única instancia, y no tanto la variabilidad de la respuesta frente a múltiples instancias. Otro aspecto importante a mencionar es la manera en que se eliminaron los datos. Como se explica en secciones anteriores, decidimos generar los escenarios de mayor carencia borrando datos de manera aleatoria y no mediante ráfagas. Este es un aspecto que potencialmente puede favorecer al método *forward-fill* por sobre los demás, ya que este podrá reconstruir la señal con mayor precisión que el resto porque puede completar estos datos eliminados utilizando información del entorno aprovechando las tendencias locales.

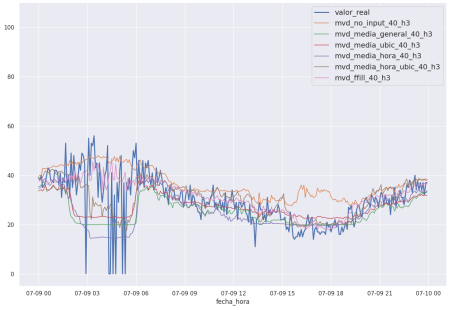
Observando que en los casos de carencia original (7 % para METR-LA y 8 % para MVD) el mejor método es la no imputación, pero que luego a partir de un 20 % de carencia en adelante los métodos presentan un mejor rendimiento, entendemos que debería existir algún valor intermedio que funcione como umbral para este cambio en el impacto de los métodos. Dados los valores obtenidos, nuestra conjetura es que existen para estos *datasets* y para esta forma de eliminar datos, un umbral en el entorno de un 10 % o 15 % donde los métodos de imputación pasan a ser más efectivos que utilizar el *dataset* original. Verificar esto y encontrar este umbral es una tarea muy difícil de llevar adelante en estas condiciones ya que requeriría de cientos de muestras de matrices y entrenamientos para obtener un resultado estadísticamente confiable, algo que naturalmente resulta altamente costoso en cómputo, y por lo tanto, prohibitivo para el contexto de este trabajo.

(a) Predicción MVD original

(b) Predicción MVD 20 %



(c) Predicción MVD 30 %



(d) Predicción MVD 40 %

(e) Predicción MVD 75 %

Figura 4.4: Visualización de predicciones para cada método de imputación en los escenarios planteados para el *dataset* MVD, en el nodo ubicado en Bulevar Artigas entre 21 de Setiembre y Bulevar España.

Capítulo 5

Conclusiones y trabajo futuro

En este capítulo desarrollamos las principales conclusiones obtenidas del desarrollo de este proyecto y algunos de los aspectos que por limitaciones de tiempo y principalmente de capacidad de cómputo, no se lograron llevar adelante, pero son importantes de mencionar ya que son líneas que permiten profundizar y mejorar los resultados obtenidos. El foco de este trabajo es estudiar la sensibilidad del modelo Graph WaveNet frente a la carencia de datos. Para esto nos planteamos una serie de experimentos basados en dos datasets, METR-LA y MVD, generando distintos escenarios de carencia de datos a partir de los mismos. Para el problema de los datos faltantes se plantea además una serie de métodos de imputación de datos aplicados previo al entrenamiento del modelo, evaluando posteriormente el rendimiento de cada uno.

5.1. Conclusiones

El primer aporte de este trabajo es el estudio y recopilación no solo de antecedentes en el área de predicción de tráfico, sino también en lo vinculado a las redes neuronales utilizadas en este tipo de tareas. Se ha logrado construir una base teórica de los fundamentos necesarios para la comprensión de la arquitectura Graph WaveNet, así como también una descripción detallada de la propia arquitectura y sus componentes.

La segunda contribución tiene que ver con la evaluación del rendimiento del modelo en escenarios con gran cantidad de datos faltantes, ya que este aspecto no ha sido objeto de estudio en investigaciones recientes. Brevemente en (Shleifer y cols., 2019) se propone que utilizar la media general como valor para imputar, ya que se observa una mejora moderada en el resultado. De cualquier manera el análisis realizado sobre este punto en esta publicación es muy superficial. Por lo tanto, exploramos distintas alternativas, teniendo como objetivo mantener el costo computacional similar pero intentando lograr algo más valioso.

En los *datasets* utilizados, tomando las versiones originales con un porcentaje de pérdida entorno al 8 %, no imputar datos resulta ser la opción más conveniente. Sin embargo, en los *datasets* con mayor volumen de pérdida, generados durante la fase de experimentación, se aprecia que los métodos de imputación mejoran frente a la alternativa de no imputar. Entre todos los métodos el que mejor resultados muestra es el *forward-fill*. Este comportamiento se acentúa a medida que aumenta la cantidad de datos faltantes, manteniéndose la opción de imputación *forward-fill* como la mejor de todas. Al haber eliminado datos de manera aleatoria y sin ráfagas, es probable que este método sea beneficiado cuando la señal presenta un comportamiento regular, explicando por que éste es el método que funciona mejor que las otras alternativas.

Otro de los aportes de este trabajo, además de la evaluación y análisis del modelo en contextos de datos faltantes, es la evaluación sobre un *dataset* nuevo, construido desde cero y fuera del conjunto de *datasets* de referencia habituales en este tipo de investigaciones. Naturalmente en las publicaciones del área de predicción de tráfico se suelen utilizar *datasets* estandarizados y de alguna manera cuidadosamente construidos. Por este motivo, es interesante observar el rendimiento del modelo en un contexto totalmente distinto al que se suele presentar públicamente.

5.2. Trabajo futuro

Debido a las limitaciones de este proyecto, principalmente en lo que refiere a la capacidad de cómputo, varios aspectos pueden ser considerados como trabajo futuro para profundizar aún más tanto en el comportamiento del modelo como en los métodos de imputación en sí.

Uno de los aspectos en los que se debería profundizar tiene que ver con la cantidad de entrenamientos del modelo. Haber elegido priorizar la robustez del entrenamiento frente a la variabilidad de escenarios, forzó a dejar de lado entrenar sobre distintas instancias de un mismo escenario. Expandir la investigación evaluando sobre múltiples instancias permitiría alcanzar un resultado estadísticamente más robusto y completo.

En una línea similar, la búsqueda de hiper-parámetros se manejó utilizando una cantidad de hiper-parámetros y valores reducida, permitiendo llevar adelante esta fase dentro de un marco de tiempo razonable para el proyecto. Mayor capacidad de cómputo permitiría incorporar no solo más valores, ampliando el espacio de búsqueda, sino que también más hiper-parámetros. Esto potencialmente permitiría obtener valores que optimicen aún más el entrenamiento del modelo.

Como ya mencionamos, el método *forward-fill* es beneficiado por la eliminación de datos de forma aleatoria. Entendiendo que esta manera de eliminar datos impacta en el rendimiento de los métodos favoreciendo algunos, sería interesante plantear matrices con distintas técnicas de eliminación. Una de las opciones consideradas fue implementar la eliminación de datos mediante el modelo Gilbert-Elliott (Gilbert, 1960). Este modelo utiliza una cadena de Markov

de dos estados para simular errores en ráfaga en un canal de comunicación, que sería útil para generar de manera artificial ráfagas de datos faltantes. Generar escenarios de carencia de datos mediante este modelo permitiría comparar el comportamiento de los métodos de imputación seleccionados, en particular el de *forward-fill*, y observar realmente que tanto impacta la manera de eliminar datos en el resultado final.

Otra línea sobre la cual profundizar sería evaluar los métodos de imputación en distintas arquitecturas de predicción de tráfico como *STAWnet* (Tian y Chan, 2021), *Traffic Transformer* (Cai, Janowicz, Mai, Yan, y Zhu, 2020) y *D2STGNN* (Shao y cols., 2022), por mencionar algunas de las más interesantes.

En el área de análisis descriptivo es interesante, en la medida que se cuente con la información, incorporar datos de conteo de vehículos para obtener una dimensión más sobre la cual realizar el análisis y observar la correlación entre las velocidades de tráfico con el volumen de tráfico. Esta información se encuentra disponible públicamente para Montevideo, pero no se logró encontrar estos datos para el caso correspondiente al *dataset* de velocidades METR-LA.

Referencias

Atwood, J., y Towsley, D. (2016). Diffusion-convolutional neural networks. En D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, y R. Garnett (Eds.), *Advances in neural information processing systems 29: Annual conference on neural information processing systems 2016, december 5-10, 2016, barcelona, spain* (pp. 1993–2001). Descargado de `https://proceedings.neurips.cc/paper/2016/hash/390e982518a50e280d8e2b535462ec1f-Abstract.html`

Bai, S., Kolter, J. Z., y Koltun, V. (2018). *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.* arXiv. Descargado de `https://arxiv.org/abs/1803.01271` doi: 10.48550/ARXIV.1803.01271

Benkraouda, O., Thodi, B. T., Yeo, H., Menendez, M., y Jabari, S. E. (2020). Traffic data imputation using deep convolutional neural networks. *IEEE Access*, *8*, 104740–104752. Descargado de `https://doi.org/10.1109%2Faccess.2020.2999662` doi: 10.1109/access.2020.2999662

Beyer, M., Ahmad, R., Yang, B., y Rodríguez-Bocca, P. (2023). Deep spatial-temporal graph modeling for efficient ndvi forecasting. *Smart Agricultural Technology*, *4*, 100172. Descargado de `https://www.sciencedirect.com/science/article/pii/S2772375523000023` doi: https://doi.org/10.1016/j.atech.2023.100172

Bruna, J., Zaremba, W., Szlam, A., y LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. En Y. Bengio y Y. LeCun (Eds.), *2nd international conference on learning representations, ICLR 2014, banff, ab, canada, april 14-16, 2014, conference track proceedings.* Descargado de `http://arxiv.org/abs/1312.6203`

Cai, L., Janowicz, K., Mai, G., Yan, B., y Zhu, R. (2020). Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, *24*(3), 736-755. Descargado de `https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12644` doi: https://doi.org/10.1111/tgis.12644

Chen, X., Lei, M., Saunier, N., y Sun, L. (2022, aug). Low-rank autoregressive tensor completion for spatiotemporal traffic data imputation. *IEEE Transactions on Intelligent Transportation Systems*, *23*(8), 12301–12310. Descargado de `https://doi.org/10.1109%2Ftits.2021.3113608` doi: 10.1109/tits.2021.3113608

Cho, K., van Merrienboer, B., Bahdanau, D., y Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. En D. Wu, M. Carpuat, X. Carreras, y E. M. Vecchi (Eds.), *Proceedings of ssst@emnlp 2014, eighth workshop on syntax, semantics and structure in statistical translation, doha, qatar, 25 october 2014* (pp. 103–111). Association for Computational Linguistics. Descargado de https://aclanthology.org/W14-4012/ doi: 10.3115/v1/W14-4012

Chung, J., Gulcehre, C., Cho, K., y Bengio, Y. (2014). *Empirical evaluation of gated recurrent neural networks on sequence modeling.* arXiv. Descargado de https://arxiv.org/abs/1412.3555 doi: 10.48550/ARXIV.1412.3555

Creß, C., Bing, Z., y Knoll, A. C. (2022). *Intelligent transportation systems using external infrastructure: A literature survey.*

Dauphin, Y. N., Fan, A., Auli, M., y Grangier, D. (2017). Language modeling with gated convolutional networks. En D. Precup y Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning, ICML 2017, sydney, nsw, australia, 6-11 august 2017* (Vol. 70, pp. 933–941). PMLR. Descargado de http://proceedings.mlr.press/v70/dauphin17a.html

Defferrard, M., Bresson, X., y Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. En D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, y R. Garnett (Eds.), *Advances in neural information processing systems 29: Annual conference on neural information processing systems 2016, december 5-10, 2016, barcelona, spain* (pp. 3837–3845). Descargado de https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html

Gilbert, E. N. (1960). Capacity of a burst-noise channel. *The Bell System Technical Journal*, *39*(5), 1253-1265. doi: 10.1002/j.1538-7305.1960.tb03959.x

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning.* MIT Press. (http://www.deeplearningbook.org)

He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. En *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, las vegas, nv, usa, june 27-30, 2016* (pp. 770–778). IEEE Computer Society. Descargado de https://doi.org/10.1109/CVPR.2016.90 doi: 10.1109/CVPR.2016.90

Hochreiter, S., y Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., y Shahabi, C. (2014, jul). Big data and its technical challenges. *Commun. ACM*, *57*(7), 86–94. Descargado de https://doi.org/10.1145/2611567 doi: 10.1145/2611567

Kingma, D. P., y Ba, J. (2015). Adam: A method for stochastic optimization. En Y. Bengio y Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings.* Descargado de http://arxiv.org/abs/1412.6980

Kipf, T. N., y Welling, M. (2017). Semi-supervised classification with graph convolutional networks. En *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings.* OpenReview.net. Descargado de `https://openreview.net/forum?id=SJU4ayYgl`

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. En F. Pereira, C. Burges, L. Bottou, y K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. Descargado de `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`

LeCun, Y., Bengio, Y., y Hinton, G. (2015, 01 de May). Deep learning. *Nature*, *521*(7553), 436-444. Descargado de `https://doi.org/10.1038/nature14539` doi: 10.1038/nature14539

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., y Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541-551. doi: 10.1162/neco.1989.1.4.541

Li, Y., Yu, R., Shahabi, C., y Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. En *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings.* OpenReview.net. Descargado de `https://openreview.net/forum?id=SJiHXGWAZ`

Long, J., Shelhamer, E., y Darrell, T. (2015). Fully convolutional networks for semantic segmentation. En *IEEE conference on computer vision and pattern recognition, CVPR 2015, boston, ma, usa, june 7-12, 2015* (pp. 3431–3440). IEEE Computer Society. Descargado de `https://doi.org/10.1109/CVPR.2015.7298965` doi: 10.1109/CVPR.2015.7298965

Mitchell, T. M. (1997). *Machine learning, international edition.* McGraw-Hill. Descargado de `https://www.worldcat.org/oclc/61321007`

Shao, Z., Zhang, Z., Wei, W., Wang, F., Xu, Y., Cao, X., y Jensen, C. S. (2022). Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. *Proc. VLDB Endow.*, *15*(11), 2733–2746. Descargado de `https://www.vldb.org/pvldb/vol15/p2733-shao.pdf`

Shleifer, S., McCreery, C., y Chitters, V. (2019). *Incrementally improving graph wavenet performance on traffic prediction.* arXiv. Descargado de `https://arxiv.org/abs/1912.07390` doi: 10.48550/ARXIV.1912.07390

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., y Vandergheynst, P. (2013, may). 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. Descargado de `https://doi.org/10.1109%2Fmsp.2012.2235192` doi: 10.1109/msp.2012.2235192

Tian, C., y Chan, W. K. V. (2021). Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies. *IET Intelligent Transport Systems*, *15*(4), 549-561. Descar-

gado de https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/itr2.12044 doi: https://doi.org/10.1049/itr2.12044

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. En *The 9th ISCA speech synthesis workshop, sunnyvale, ca, usa, 13-15 september 2016* (p. 125). ISCA. Descargado de http://www.isca-speech.org/archive/SSW_2016/abstracts/ssw9_DS-4_van_den_Oord.html

Velocidad promedio vehicular en las principales avenidas de montevideo - catálogo de datos abiertos. (2021, Mar). https://catalogodatos.gub.uy/dataset/intendencia-montevideo-velocidad-promedio-vehicular-en-las-principales-avenidas-de-montevideo. Departamento de Movilidad - Centro de Gestión de Movilidad.

Weerakody, P. B., Wong, K. W., Wang, G., y Ela, W. (2021). A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, *441*, 161-178. Descargado de https://www.sciencedirect.com/science/article/pii/S0925231221003003 doi: https://doi.org/10.1016/j.neucom.2021.02.046

Wu, Z., Pan, S., Long, G., Jiang, J., y Zhang, C. (2019). Graph wavenet for deep spatial-temporal graph modeling. En S. Kraus (Ed.), *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI 2019, macao, china, august 10-16, 2019* (pp. 1907–1913). ijcai.org. Descargado de https://doi.org/10.24963/ijcai.2019/264 doi: 10.24963/ijcai.2019/264

Yu, B., Yin, H., y Zhu, Z. (2018, jul). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. En *Proceedings of the twenty-seventh international joint conference on artificial intelligence.* International Joint Conferences on Artificial Intelligence Organization. Descargado de https://doi.org/10.24963%2Fijcai.2018%2F505 doi: 10.24963/ijcai.2018/505

Yu, F., y Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. En Y. Bengio y Y. LeCun (Eds.), *4th international conference on learning representations, ICLR 2016, san juan, puerto rico, may 2-4, 2016, conference track proceedings.* Descargado de http://arxiv.org/abs/1511.07122

Zhao, B., Lu, H., Chen, S., Liu, J., y Wu, D. (2017). Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, *28*(1), 162-169. doi: 10.21629/JSEE.2017.01.18