UNIVERSIDAD DE SALAMANCA
MÁSTER UNIVERSITARIO EN MODELIZACIÓN MATEMÁTICA

Modelización del flujo de tráfico mediante algoritmos inteligentes

AUTOR: Marcos Severt Silva

TUTORES: Sara Rodríguez González
Roberto Casado Vara

Curso 2020-2021

UNIVERSIDAD DE SALAMANCA
MÁSTER UNIVERSITARIO EN MODELIZACIÓN MATEMÁTICA

Modelización del flujo de tráfico mediante algoritmos inteligentes

AUTOR: Marcos Severt Silva

TUTORES: Sara Rodríguez González
Roberto Casado Vara

Curso 2020-2021

Certificado de los tutores

Los profesores Dra. Sara Rodríguez González y Dr. Roberto Casado Vara, miembros del departamento de informática y automática de la Universidad de Salamanca y tutores de este trabajo de investigación.

Hacen constar:

Que el trabajo titulado "**Modelización del flujo de tráfico mediante algoritmos inteligentes**", que se presenta, ha sido realizado por D. Marcos Severt Silva para la superación de la asignatura Proyecto de Fin de **Máster de a Titulación de Máster Universitario en Modelización Matemática**.

En Salamanca, a 12 de Julio de 2021

RODRIGUEZ GONZALEZ SARA - 70864126E

Firmado digitalmente por RODRIGUEZ GONZALEZ SARA - 70864126E
Fecha: 2021.07.12 11:43:11 +02'00'

Dra. Sara Rodríguez González

CASADO VARA ROBERTO CARLOS - 45686237A

Firmado digitalmente por CASADO VARA ROBERTO CARLOS - 45686237A
Fecha: 2021.07.12 11:32:03 +02'00'

Dr. Roberto Casado Vara

Índice general

Índice de figuras

Capítulo 1

Resumen

Este Trabajo de Fin de Master consiste en la modelización del flujo de tráfico de vías urbanas mediante el uso de algoritmos inteligentes y así determinar cual de todos ellos se aproxima mejor a dar una solución al problema.

El desarrollo de este proyecto se inicia mediante la investigación de modelos basados en inteligencia artifical aplicados a este problema anteriormente. Tras ello se lleva a cabo la planificación del procesado de los datos existentes lo que nos permitirá prepararlos para ser empleados por los diferentes algoritmos inteligentes.

Una vez se tienen preparados los datos se procede a analizarlos para estudiar diferentes condiciones de los mismos con el objetivo de comprobar si se pueden emplear en los diferentes algorimtos inteligentes seleccionados.

A partir de estos datos preprocesados y analizados, se cosntruyen los diferentes modelos que van a ser empleados en el desarrollo del proyecto.

Una vez se tienen construidos los modelos, se emplearán con los datos existentes para comprobar cuanto se acercan las diferentes predicciones a la realidad y así determinar cual de todos ellos es el más óptimo para modelizar este problema.

El código del proyecto se ha desarrollado en Python junto con algún entorno de programación en cloud como es el caso de Google Colab.

La documentación de este proyecto y su desarrollo consta de la presente memoria y el anexo final correspondiente al código empleado para desarrollar y probar los diferentes modelos expuestos.

Capítulo 2

Abstract

This Master Thesis consists of modeling the traffic flow of urban roads by using intelligent algorithms and thus determine which of them best approximates to provide a solution to the problem.

The development of this project begins with the investigation of models based on artificial intelligence applied to this problem previously. After that, the planning of the processing of the existing data is carried out, which will allow us to prepare them to be used by the different intelligent algorithms.

Once the data are prepared, we proceed to analyze them to study different conditions in order to check if they can be used in the different intelligent algorithms selected.

From these preprocessed and analyzed data, the different models that will be used in the development of the project are built.

Once the models are built, they will be used with the existing data to check how close the different predictions are to reality and thus determine which of them is the most optimal for modeling this problem.

The project code has been developed in Python together with a cloud programming environment such as Google Colab.

The documentation of this project and its development consists of this report and the final annex corresponding to the code used to develop and test the different models presented.

Capítulo 3

Introducción

3.1. Motivación

La predicción temporal del tráfico vial en una ciudad grande es muy crítica para las difrentes empresas de mensajería ya que sin una previsión adecuada del tráfico, la planificación de las rutas de entrega sería ineficiente haciendo que los clientes tengan largos tiempos de espera y decidan cambiar de proveedor. Sin embargo, no se trata de una tarea sencilla ya que se basa en hacer predicciones en base a la arbitrariedad del tráfico, pues este depende exclusivamente del comportamiento humano. Debido a que es muy importante en la planificación y logística del transporte, la previsión del tráfico vial siempre ha sido un tema amplio de investigación en la comunidad de ingenieros durante los últimos cuarenta años [12]. Un gran número de estudios empíricos demuestran que el tráfico es una cuestión dificil de modelar [24]. Esto se debe al comportamiento irregular y arbitrario del mismo. La situación del tráfico está estrechamente relacionada con la vida cotidiana, que ha atraído una atención cada vez mayor. Por lo tanto, existe una gran necesidad de proporcionar una supervisión dinámica del tráfico para realizar y actualizar continuamente las variables de tráfico y las predicciones de tiempo de enlace entre diferentes puntos. El mejorar la precisión de las predicciones del flujo de tráfico beneficiaría a la gestión inteligente del tráfico en términos de optimización de rutas o reducción de la congestión de diferentes vías [7]. Este proceso de predicción de tráfico es complejo al verse afectado por diferentes factores como puede ser la disponibilidad de los datos, el área a la que se desea aplicar, la metereología, etc. Actualmente el método para la predicción de tráfico se basa en la selección de características del flujo de tráfico y los parámetros del modelo en base a algunos supuestos básicos. Esto implica que la precisión de la predicción está intrinsecamente relacionada con la precisión de estos supuestos básicos, sin embargo, este defecto puede corregirse mediante el uso de una cantidad de datos lo suficientemente grande. Recientemente gracias al desarrollo de las tecnologías Big Data, la existencia de plataformas para el procesamiento masivo de datos y la filosofia Open Source, la predicción del flujo de tráfico se ha vuelto mucho más precisa [20].

3.2. Planteamiento del trabajo

La predicción del flujo de tráfico como campo de la ciencia de datos, ha experimentado un gran crecimiento gracias a la incorporación de mecanismos informáticos para el tratamiento de grandes cantidades de información. El incluir técnicas de aprendizaje automático ha supuesto un amplio beneficio para este campo gracias a la reducción de costes y tiempo permitiendo así llevar el campo a un nuevo nivel.

La capacidad de procesamiento simultanea de grandes cantidades de datos es uno de los beneficios que el aprendizaje automático trae al campo de las predicciones temporales. Muchas veces la toma de decisiones en diferentes ámbitos profesionales está ligada a la existencia de suficiente información, es por ello que poder adelantar esta información de forma precisa supondría un ahorro importante en costes de desarrollo y tiempos en toma de decisiones.

A pesar de todos estos avances en el campo de la predicción temporal, áun hay un amplio camino que recorrer respecto a la precisión lograda por los diferentes algoritmos intelijentes aplicados en el campo. De forma paralela el denominado Deep Learning ha tenido avances más que significativos solventando algunos problemas de predicción temporal relaccionados con el flujo de tráfico.

Por todo ello este TFM tiene como fin plantear algunos modelos capaces de modelizar con exactitud el flujo de tráfico existente en dos carreteras de la ciudad española de Zaragoza. Se generarán los diferentes modelos y se obtendrán diferentes resultados y métricas con los que comparar y así poder determinar que modelo cuenta con más exactitud y precisión. Para llegar a ese modelo, se procederá a entrenarlo con los diferentes datos medidios por cuatro radares situados en esas dos calles. Las preguntas en torno a las cuales se intentará conducir el TFM son:

- ¿Qué modelo basado en inteligenci artifical es mejor para predecir el flujo de tráfico?
- ¿Qué parámetros hay que tener en cuenta a la hora de modelizar el flujo de tráfico?
- ¿Cómo podemos determinar cual de los modelos planteados es mejor para resolver el problema?
- ¿Cómo podemos entrenar los modelos de forma eficiente?

3.3. Estructura de la memoria

Este apartado tiene como objetivo representar la estructura de la memoria del TFM y sus diferentes apartados.

En primer lugar, el Capítulo 1 tiene como objetivo dar una visión genérica del contexto en el que se desarrolla el Trabajo de Fin de Master, explicando la motivación de aplicar algoritmos inteligentes a un problema de predicción de flujo de tráfico y los posibles beneficios de modelizar este problema con técnicas de ML. Este capítulo también recoge un resumen esquemático punto a punto de las partes de la memoria explicando que objetivo tiene cada una de ellas.

En segundo lugar, el Capítulo 2 se compone de dos partes claramente diferenciadas:

- Estado del Arte: En este apartado se tienen en cuenta estudios previos en el uso de algoritmos inteligentes en problemas de predicciones en el flujo de tráfico. Este apartado se realiza mediante una exhaustiva revisión bibliográfica de los últimos años.

- Contexto: En este apartado se presentan una serie de conceptos teóricos que son necesarios para entender el resto de la memoria y en los cuales se fundamenta el TFM.

En tercer lugar, el Capítulo 3 recoge los objetivos generales y específicos que se han seguido para desarrollar el presente TFM. Este capítulo también cuenta con un apartado de metodología donde se establece como se desarrolló el proyecto. Es a partir de este capítulo que comienza la parte experimental del proyecto donde se explicarán y probarán los diferntes modelos planteados.

En cuarto lugar, el Capítulo 4 recoge la preparación y análisis de los datos, así como la creación de un conjunto de datasets con los que los diferentes modelos puedan ser entrenados. Una vez tenemos listos estos datos, en este capítulo se realiza un análisis de las series temporales para ver si es factible realizar una predicción a corto y largo plazo.

En quinto lugar, el Capítulo 5 recoge la explicación de cada uno de los modelos desarrollados, así como el proceso de entrenamiento para cada una de las calles y los resultados obtenidos en cada caso. Es en este capítulo donde se obtendrán los diferentes parámetros que permitirán medir la eficiencia de cada modelo.

En sexto lugar, el Capítulo 6 recoge el análisis y la discusión sobre los diferentes resultados obtenidos para cada uno de los modelos. Con el fin de determinar cual de los modelos es el más eficiente, se añaden comentarios y recomendaciones según lo observado tanto en la implmentación como en los resultados.

En séptimo y último lugar, en el Capítulo 7 se recogen las conclusiones extraidas durante todo el proceso de desarrollo del TFM junto a una serie de líneas futuras que quedan pendientes para continuar este trabajo.

Capítulo 4

Contexto y Estado del arte

4.1. Contribuciones a la predicción del tráfico

Hoy en día, numerosos métodos y modelos han sido propuestos para la predicción del tráfico. Estos métodos se han diseñado con el objetivo de aplicarlo a diferentes tipos de tráfico como es el caso de las autopistas, carreteras, ciudades, etc. Además, con el objetivo de crear modelos más precisos se han incorporado una serie de factores como pueden ser las condiciones metereológicas, las festividades, los accidentes, etc. En general estos modelos presentan más precisión cuanto más simple es el caso a estudiar pues menos factores afectan a la circulación de vehículos en la vía.

4.1.1. Predicción del tráfico

En este apartado se va a hacer una revisión de métodos clásicos para la predicción de series temporales aplicados al tráfico de diferentes vías. Uno de los métodos empleado actualmente para la predicción del tráfico es el basado en las Redes de Bayes (BNs) mediante la estrategia de selección de nodos. Una de las estrategias de selección de nodos planteadas es el enfoque Graphical Lasso, otra es el enfoque de vecindad geográfica y otra el enfoque de correlación directa [13]. Otro de los métodos empleados para la predicción del tráfico a corto plazo es el uso de la dependencia espacial entre dos enlaces, donde la información espacial aumentará considerablemente la precisión de las diferentes predicciónes. Este método es fiable pero hay que tener en cuenta la diferencia entre la dependencia espacial realista de los enlaces de tráfico en una red real y la red simulada[5]. Con el objetivo de poder establecer sistemas avanzados de información al viajero (ATIS) y sistemas avanzados de gestión del tráfico (ATMS) que hacen uso de métodos de predicción de tráfico, se han empleado sistemas de simulación de tráfico en tiempo real como es el caso de DynaMIT[3], [4] y DYNASMART-X[9]. Estos sitemas hacen uso de los Filtros de Kalman (KF) como función básica para la estimación del tráfico. Posteriormente y ampliando las investigaciones previas, se han empleado algoritmos de Filtros de Kalman Extendidos (EFKs) incorporando así una red de autopistas macroscópica estocástica [21], [22].

4.1.2. Inteligencia artificial para predicción del tráfico

En los últimos años, en la tarea de la predicción del tráfico, los métodos de aprendizaje automático han planteado graves desafíos a los modelos estadísticos tradicionales [2]. A pesar de la existencia de una amplia gama de algoritmos dedicados al aprendizaje automático, no existen métodos conocidos para determinar cual es el método adecuado para abordar los problemas de predicción del tráfico que se recogen en la literatura. Las estrategias de IA han demostrado su eficacia en la gestión de este problema en diferentes ámbitos, pero apenas se ha investigado en la estimación del tráfico [1]. Usando estos modelos, como k vecinos más cercanos (KNN), máquinas de vectores de soporte (SVM) y redes neuronales (NN), se puede lograr una mayor precisión de predicción y un modelado de datos más complejo [25]. Uno de los métodos que hace uso de la inteligencia artificial es el basado en Cross-Domain Learning. En este método se propone el aprendizaje entre dominos para poder predecir de forma precisa el tráfico en base a la correlación entre dominios y las correlaciones temporales. Mediante la correlación de estas características con datos reales del tráfico se pueden extraer correlaciones transversales relevantes [6]. Otra opción son los enfoques en Deep Learning los cuales se han aplicado ampliamente y con éxito a diversas tareas de predicción de tráfico. Se ha logrado un progreso significativo en trabajos relacionados en el ambito de Deep Learning, como Deep Trust Network (DBN) [10], [8] y Stacked Autoencoder (SAE) [14].

4.1.3. Enfoque actual en la modelización de predicciones del tráfico: Long short-term memory

En la gestión inteligente del tráfico, la modelización precisa y oportina del tráfico a corto y largo plazo juega un papel fundamental en el control del transporte inteligente [11]. Actualmente uno de los métodos favoritos de modelización de problemas de predicción de flujo mediante técnicas de inteligencia artificial, es el uso de las Long Shor-Term Memory (LSTM) [23]. Las LSTM son un tipo de red neuronal recurrente, es decir, existe realimentación entre las neuronas que la componen y la información se propaga por tanto hacia adelante y hacia atrás [18]. A su vez, las redes neuronales recurrentes, se componen de capas de neuronas cuyas conexiones forman un grafo dirigido, permitiendo asi la predicción de un comportamiento de forma dinámica en el tiempo [17], [19]. Actualmente algunos estudios propenen el uso de arquitecturas de redes neuronales convolucionales LSTM para poder modelizar el flujo de tráfico de varios carriles [16]. Estos estudios en comparación a los métodos ya existentes permiten aplicar numerosas características del entorno para caracterizar las condiciones del tráfico y predecir el tráfico en el futuro [15].

4.2. Conceptos teóricos

Antes de comenzar con la parte técnica de la memoria es importante tener en mente algunos conceptos relaccionados con el Machine Learning y así poder comprender conceptos empleados a lo largo del desarrollo del TFM,

4.2.1. Machine Learning

El denominado aprendizaje automático o Machine Learning (ML) es una rama de la inteligencia artificial centrada en emplear información y algoritmos especializados en imitar el procedimiento de aprendizaje humano para así aumentar su precisión. El aprendizaje automático es una parte muy importante y esencial en el creciente campo de la ciencia de datos o Data Science. Estos algoritmos inteligentes hacen uso de métodos estadísticos para poder entrenar en las tareas de clasificar, predecir o encontrar nuevos conceptos haciendo uso de grandes fuentes de datos. El concepto de Machine Learning es definido por primera vez en 1950 por Arthur Samuel como el campo de estudio que da a los ordenadores la capacidad de aprender sin ser programados explicitamente. Esta definición es válidada actualmente por los principales investigadores en el campo de la inteligencia artificial.

Cuando las diferentes compañías crean diferentes programas inteligentes lo más propable es que empleen diferentes técnicas de aprendizaje automático de forma que estos puedan aprender sin necesidad de ser programados de forma explícita por un equipo de programadores experimentados. El principal objetivo de los algoritmos inteligentes, es crear modelos que permitan exhibir çomportamientos inteligentes de humanos", es decir, que sean capaces de recordar, entender o realizar acciones en el mundo real. Sin embargo, es importante tener en cuenta que escribir un programa para que la máquina sea capaz de aprender de forma automática puede llevar mucho tiempo o ser imposible en algunos casos. Un ejemplo de esto sería intentar hacer que un ordenador sea capaz de determinar que dos personas cualquiera del mundo son diferentes o iguales, mientras que un humano sería capaz de hacerlo de forma inmediata al planteamiento del problema, una máquina nunca podría aprender a distinguir al conjunto total de la población.

El Machine Learning comienza con una cantidad importante de datos, pudiendo estos ser valores numéricos, fotografias, testo videos, etc. Estos datos se preprocesan quedando listos para ser empleados en el proceso de entrenamiento de los diferentes algoritmos inteligentes. Por esto es importante tener en cuenta que cuantos más datos, más preciso será el programa inteligente.Una vez se tienen preparados los datos los expertos en algoritmos inteligentes escogen el modelo de aprendizaje más adecuado y dejan al modelo entrenando para aumentar su precisión.Algunos de estos datos preparados son separados del conjunto original para ser empleados como método de evaluación y así poder medir la precisión aproximada de los modelos creados y entrenados. Una vez finalziado el entrenamiento es posible que el propio experto decida ajustar manualmente parámetros del modelo para hacerlo más preciso y obtener mejores resultados. Podemos clasificar el aprendizaje automático en dos grandes grupos.

Aprendizaje automático supervisado

El aprendizaje automático supervisado consiste en entrenar un modelo partiendo de una serie da datos etiquetados, es decir, datos de los cuales sabemos cual debería ser la salida de nuestro modelo. Esto permite que el modelo aprenda de ellos y se vuelva mas preciso. Este tipo de aprendizaje es el más empleado en la actualidad dada la facilidad para entrenar modelos con grandes cantidades de datos. Un ejemplo de aplicación de este problema es por ejemplo un conjunto de medidas de area y sus precios, a partir de los cuales el modelo podría aprender a asignar precios a diferentes superficies. A continuación se hará especial incapié en este grupo de algoritmos pues será el empleado en los diferentes

modelos aplicados posteriormente. En primer lugar sería adecuado realizar una breve notación matemática que simplifique la explicación de otros conceptos:

- X: Conjunto de entradas.
- Y: conjunto de etiquetas o respuestas esperadas para cada entrada.
- $y : X \to Y$: función de dependencia entre las entradas y etiquetas.
- $y_i = y\left(x^{(i)}\right), i = 1, \ldots, l$: valores conocidos de la función de dependencia.

El problema de aprendizaje supervisado sera entonces encontrar una función h,

$$h : X \to Y$$

de forma que esta aproxime el conjunto de entradas a las etiquetas de salida, es decir, el objetivo es entrenar a un algoritmo de forma que aproxime esa función.

En segundo lugar es adecuado definir que es cada una de las entradas. Formalmente las podemos definir como:

$$x^{(i)} \in \mathbb{R}^n$$

, es decir, una entrada es un vector de n dimensiones donde n es el conjunto de características que queremos que el algoritmo tenga en cuenta sobre cada una de las entradas. Por este motivo es crítico el proceso de selección de características para cada entrada pues dependen de la naturaleza del algoritmo y del problema a resolver.

En tercer y último lugar es adecuado explicar el proceso de aprendizaje en los modelos de aprendizaje supervisado. Si consideramos un conjunto de parejas x_i, y_i que representan una entrada y su respectiva etiqueta o salida esperada, para el aprendizaje supervisado tendremos un conjunto de entrenamiento de la forma:

$$\left\{\left(x^{(i)}, y^{(i)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\} \subset X$$

El proceso de aprendizaje cuenta entonces con dos pasos importantes que pueden verse en la figura 4.1:

1. Entrenamiento: Un conjunto de las parejas de entrada/etiqueta es usado para entrenar al algoritmo y así dar con la función h adecuada.
2. Prueba: Otro conjunto de las parejas es usado para probar el funcionamiento del algoritmo entrenado.

Aprendizaje no supervisado

A diferencia de en el aprendizaje supervisado, en el no supervisado es el propio algoritmo el que busca patrones no etiquetados. Esto supone que el algoritmo inteligente pueda dar con patrones que las personas que lo hayan desarrollado no esten buscando de forma explicita. El aprendizaje no supervisado permiten realizar tareas más complejas a nivel computacional que los modelos de aprensizaje supervisado, sin embargo, hay que tener en cuenta el factor de no ser predecibles. Es importante recalcar que el aprensizaje no

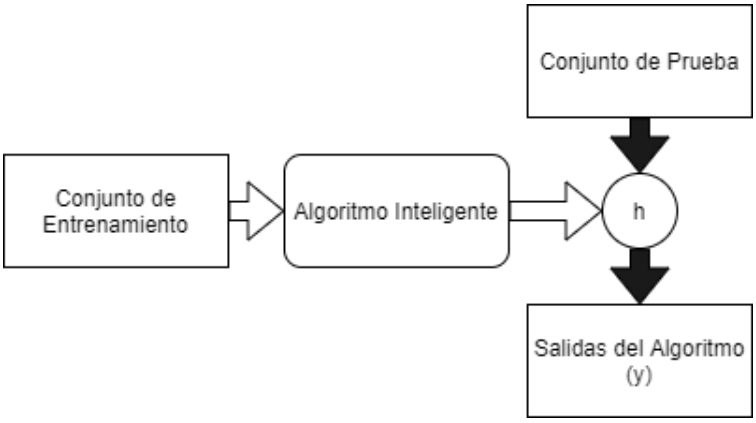

Figura 4.1: Proceso en un algoritmo de aprendizaje supervisado.

supervisado es importante pues ayuda a descubrir patrones desconocidos por los humanos en los datos siempre teniendo en cuenta la imposibilidad de determinar la precisión de los algoritmos pues no se conoce el resultado previo o esperado. Estos algoritmos manejan datos sin haber sido entrenados de forma previa, es decir, en este caso consisten en una función que realiza una tarea determinada con los datos de los que dispone. Estos datos son datos no etiquetados, es decir, el proposito de estos algoritmos es la exploración de los datos con el objetivo de encontrar resultados. Podemos definir los siguientes tipos de algoritmos no supervisados:

- Agrupamiento: Se trata de determinar un patrón existente es lo datos no etiquetados que se dan como entrada. Estos algoritmos son denominados de Cluster y se encargarán de procesar los datos y encontrar grupos en los mismos en caso de que existan. El proceso de agrupamiento permite predefinir cuantos grupos se desean encontrar. Para agrupar pueden emplear diferentes técnicas:
  - Particionado: Los datos se agrupan de forma que cada entrada pueda pertenecer a un solo grupo de salida.
  - Aglomerado: Los datos forman cada uno un grupo concreto, siendo esto utilizado en agrupaciones jerárquicas por ejemplo.
  - Solapado: Los datos pueden pertenecer a uno o a varios grupos de forma simultánea.
  - Probabilístico: Consiste en emplear diferentes técnicas probabilísticas para determinar los grupos.
- Asociación: Esta técnica consiste en encontrar relaciones entre variables de grandes grupos de datos.

4.2.2. Redes Neuronales Artificiales

Las denominadas redes neuronales artificiales son un conjunto de neuronas artificiales que se basan en el funcionamiento de las neuronas existentes en el cerebro humano. Para comprender como funcionan estas neuronas artificiales es imporntante comprender el

funcionamiento de nuestras neuronas y establecer una similitud. Las partes que forman una neurona humana son:

- Dendritas: Captan impulsos nerviosos de otras neuronas.
- Soma: Unidad de procesamiento de los estímulos nerviosos.
- Axón: Emite impulsos nerviosos a las neuronas cercanas.

Por otra parte las neuronas artificiales tenemos las siguientes partes:

- Entradas: Pueden ser binarias o analógicas en función del algoritmo empleado.
- Pesos de las entradas: Representan la interaccón entre la neurona anterior y la posterior.
- Bias: Proporciona el valor del potencial de la neurona e función de sus pesos y entradas. Generalmente es una suma de los pesos.
- Función de activación: Proporciona el estado de activación actual de la neurona en función de su estado anterior y su potencial postsináptico.

El esquema de su funcionamiento puede verse en la figura 4.2.

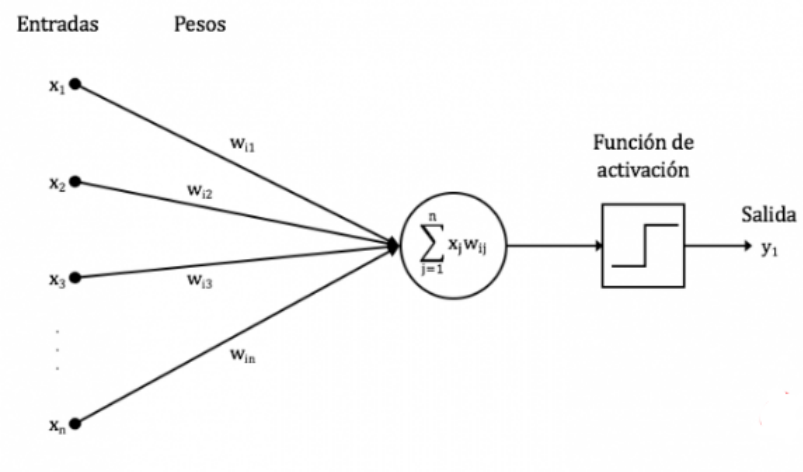

Figura 4.2: Neurona artificial.

4.2.3. Redes Neuronales Recurrentes RNN: Una visión general

Para entender como funciona una red neuronal recurrente internamente es importante comprender la diferencia entre estas y las redes neuronales convencionales y convolucionales. En las redes neuronales convencionales y convolucionales la información circula en una sola dirección, desde la entrada hasta salida lo que las hace ideales para el reconocimiento de patrones y clasificación. Este enfoque tradicional implica que las redes para producir una salida, solo tienen en cuentra la entrada actual, sin embargo, las redes neuronales recurrentes es capaz de tener en cuenta las entradas y salidas previas permitiendo así

la predicción temporal en base a datos pasados. Si hablamos en términos temporales las RNN tienen en cuenta estados temporales actuales y previos siendo ideales para analizar secuencias temporales.Para entender las RNN el primer paso es definir una serie de conceptos:

- Instante de tiempo: Un instante de tiempo es un entero que define una posición dentro de la secuencia temporal o time series. En nuestro caso cada instante de tiempo es una medición de flujo de tráfico.
- x_t, y_t: Entrada a la RNN en el instante de tiempo t y salida de la RNN en el instante de tiempo t repectivamente. Estos conceptos pueden verse en la figura 4.3. Como

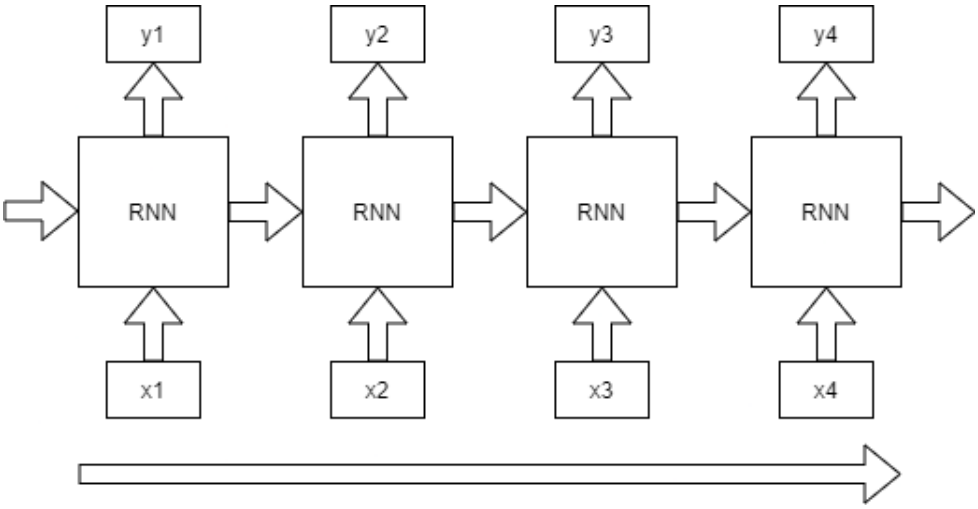

Figura 4.3: Entradas y salidas de una RNN.

puede verse en la figura, en cada instante t, la RNN no tiene una sola entrada y salida pues hay que tener en cuenta las felchas horizontales que representan la activación en el instante anterior y en el isntante actual. Esto se puede ver en la figura 4.4.

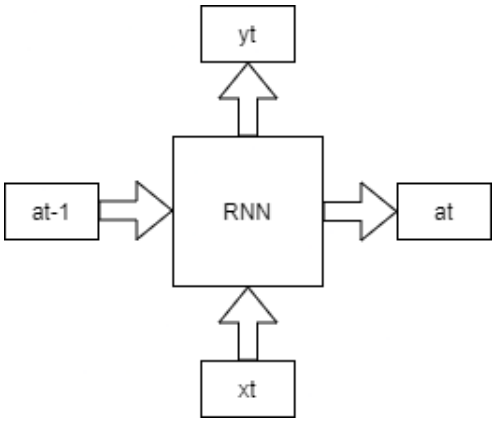

Figura 4.4: Entradas y salidas de una RNN con activaciones.

- Activación: Esta activación también es conocida como Hidden State o Estado Oculto. Son estas activaciones las que permiten preservar y compartir la información entre dos estados de tiempo.

El siguiente paso es explicar el funcionamiento interno de las RNN, para así comprender la utilidad en la modelización de problemas de predicción con series temporales. La arquitectura de los modelos RNN es similar a la de otros basados en redes neuronales artificiales. En términos generales, una red neuronal recurrente tiene una capa de entrada, una capa oculta y una capa de salida, actuando siempre en un orden estándar.

- Capa de entrada: La capa de entrada es responsable de obtener datos, preprocesar y luego pasar los datos filtrados a la capa oculta.
- Capa oculta: La capa oculta está compuesta por redes neuronales, algoritmos y funciones de activación para recuperar información útil de los datos.
- Capa de salida: Transforma la información de la capa oculta en el resultado esperado.

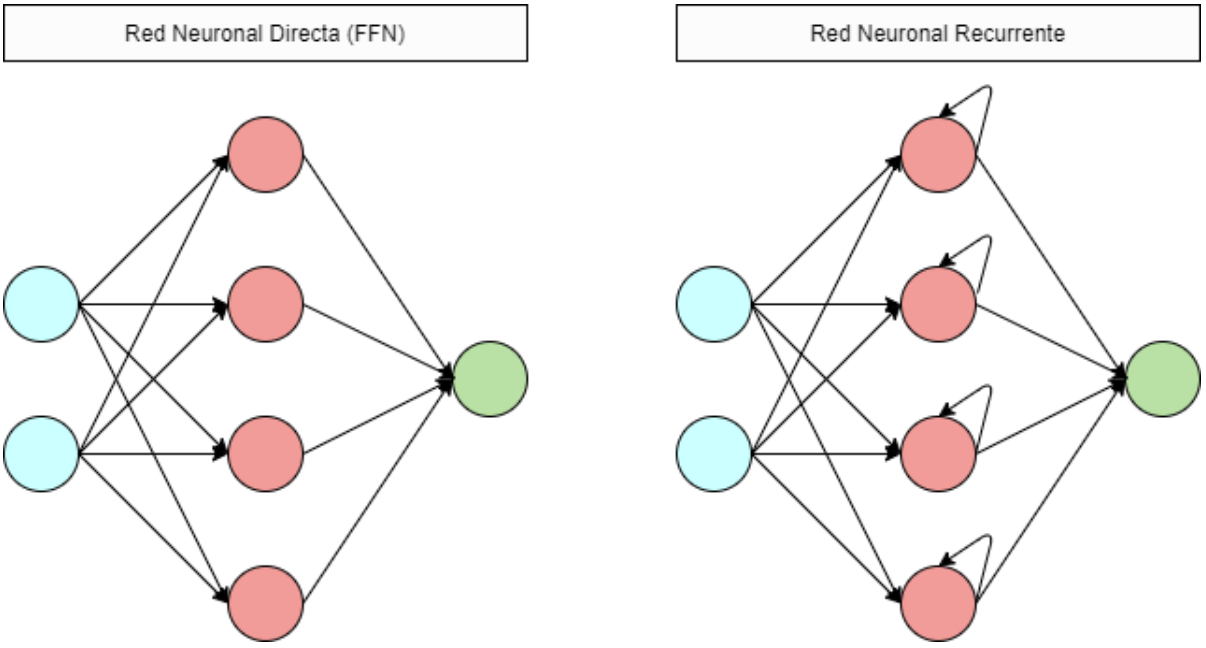

Figura 4.5: Comparación entre FFN y RNN.

En la figura 4.5 se representa la diferencia entre una red neuronal directa (FFN) o tradicional y una red neuronal recurrente. Tal y como se ha explicado antes en las FFN la información solo se mueve en una dirección, comenzando en la capa de entrada luego a la oculta y finalmente a la de salida. Dado que la red neuronal FFN solo considera la entrada actual, no tiene información sobre lo que sucedió en el pasado, excepto por el procedimiento de entrenamiento. En las RNN el flujo de la informaciόm cambia considerablemente. Toda la información que pasa por la red pasa por un bucle haciendo asi que cada resultado dependa del anterior para tomar una decición. En las RNN a cada capa se le asigna el mismo peso y sesgo transformando todas las cariables independientes en dependientes, por tanto, el bucle de las RNN garantiza que la información pasada permanece en su ”memoria”.

4.2.4. Long Short-Term Memory

Las LSTM surgen con el objetivo de solventar los problemas de memoria existentes en las RNN tradicionales. Este problema de las RNN consiste en la pérdida de relación entre los elementos distantes en el tiempo haciendo así que se pierda todo el potencial a la hora de realizar predicciones a largo plazo teniendo en cuenta una gran cantidad de datos pasados. Durante el proceso de retropropagación se ven afectadas por el denominado "desaparición del gradiente". Al ser los gradientes valores utilizados para la actualización de pesos en las redes neuronales, este problema consiste en la contracción del gradiente a lo largo de su propagación atrás en el tiempo. Si un gradiente se vuelve muy pequeño no aportará nada al proceso de aprendizaje puesto que:

$$NuevosPesos = PesosAnteriores - LearningRate * Gradiente$$

Es decir, en las RNN aquellas capas que reciban un gradiente muy pequeño dejaran de aprender. Al usarse el algoritmo de retropropagación las capas que se suelen ver más afectadas son las capas anteriores eliminando así la memoria a largo plazo. Para solucionar este problema surgen las LSTM eliminando así la dependencia exclusiva de una memoria a corto plazo. Para ello cuentan con mecanismos internos denominados puertas que permiten que cada celda regule el flujo de información. Esto puede verse en la figura 4.6. El concepto

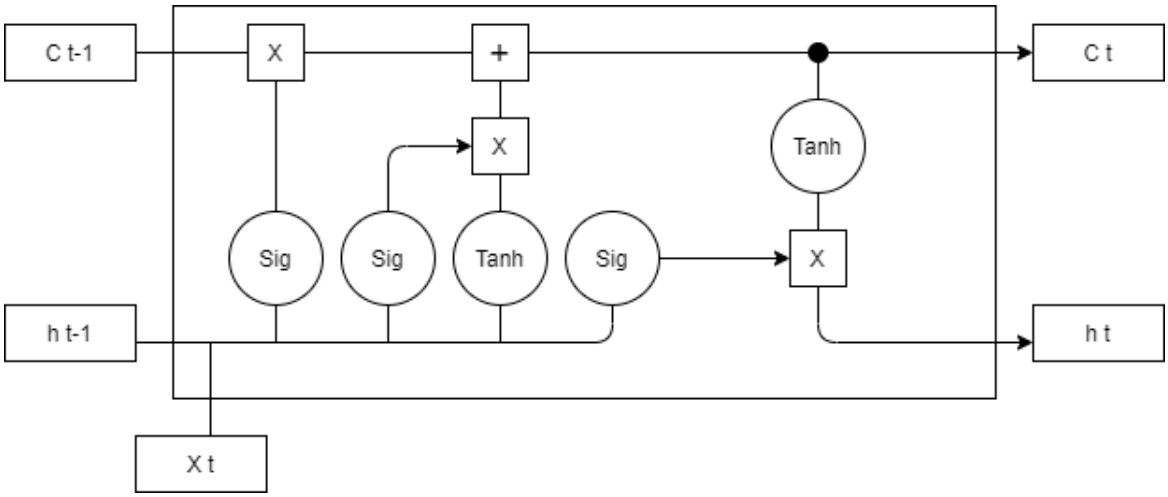

Figura 4.6: Celda de una LSTM

principal a tener en cuenta en las LSTM es el estado de la celda y las diferentes puertas. Este estado funciona como un medio de transporte para la información a lo largo de toda su secuencia, es decir, como una memoria de la red neuronal. A medida que la secuencia de información viaja a través del estado de la celda la información se agrega o elimina a este estado mediante las puertas de cada celda. Siendo estas las que determinan que información es relevante que pase de un estado previo al siguiente. Antes de proceder explicar el funcionamiento de estas celdas y sus puertas es conveniente explicar las dos funciones de activación que aparecen representadas en la figura 4.6:

- Activación de Tanh: Esta función de activación se encarga de regular los valores que pasana a lo largo de nuestra red, trasnformando cualquier entrada a valores comprimidos entre el -1 y el 1. Esto puede verse en la figura 4.7. En las redes neuronales

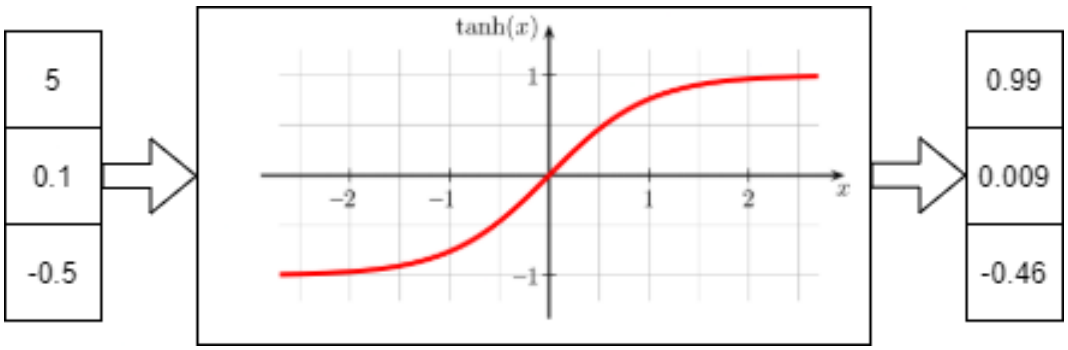

Figura 4.7: Activación Tanh

los valores sufren numerosos cambios gracias a diversas funciones matemáticas. Esta función de activación Tanh nos asegura que los valores estarán entre el -1 y el 1.

- Activación Sigmoide: Esta función es similar a la anterior (Tanh), sin embargo, comprime los valores entre el 0 y el 1. Esto puede verse en la figura 4.8. Esta

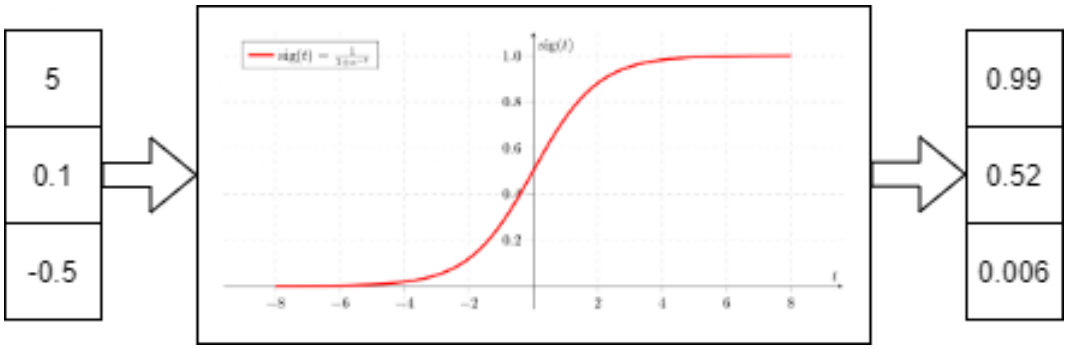

Figura 4.8: Activación Sigmoide

función de activación es util para mantener u olvidar información, pues un valor multiplicado por 1 es el mismo valor y por 0 es 0.

El siguiente paso para entender el funcionamiento de una LSTM es explicar las diferentes puertas que forman cada una de las celdas.

- Puerta de olvido: Esta es la puerta que en cada celda decide que información debe mantenerse y cual debe olvidarse. Al pasar la entrada por una sigmoidea, cuanto más cerca del 0 significa que es adecuado olviadrlo y cuanto más cerca del 1 más adecuado mantenerlo. Esta parte de la celda puede verse en la figura 4.9 marcada en rojo.

- Puerta de entrada: Esta puerta es la encargada de actualizar el estado interno de la celda. El procedimiento de como funciona se detalla a continuación:

  1. El estado oculto anterior h_{t-1} y la entrada actual X_t pasan a una función signmoidea para asi decidir que valores son importantes (Cercanos al 1) y cuales no (Cercanos al 0).

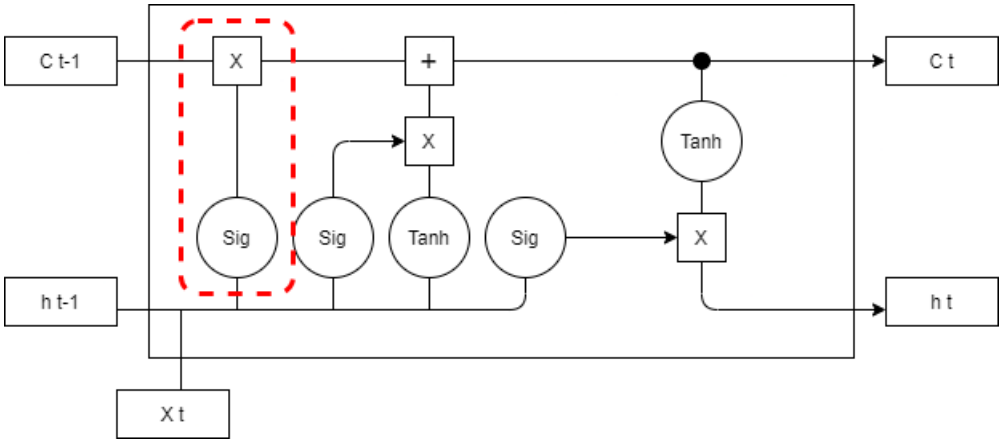

Figura 4.9: Puertas de una celda LSTM: Puerta de olvido

2. El estado oculto anterior h_{t-1} y la entrada actual X_t pasab a una funcion tanh para reducir valores entre el -1 y el 1 y así regular de forma adecuada la red.
3. Se multiplica la salida de la tanh y la de la sigmoidea para ponderar la importancia de la información.

Esta parte de la celda puede verse en la figura 4.10 marcada en rojo.

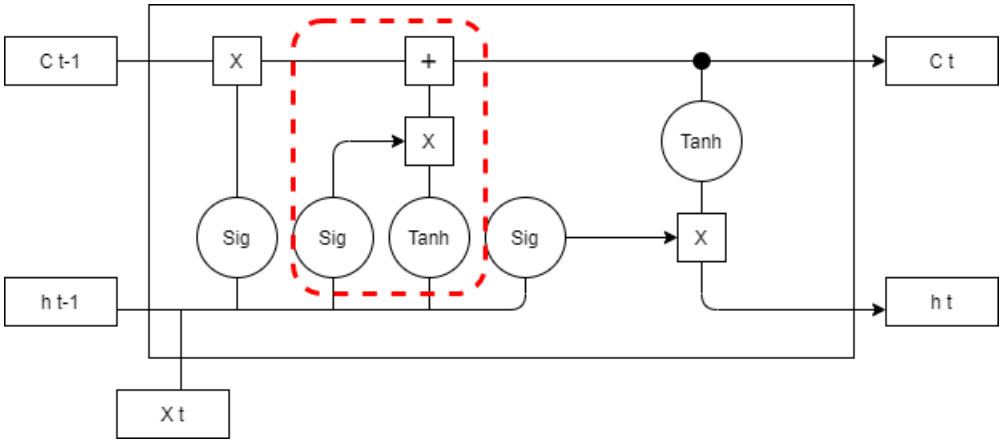

Figura 4.10: Puertas de una celda LSTM: Puerta de entrada

- Estado interno de la celda: El estado anterior C_{t-1} se multiplica por el vector producido por la puerta de olvido pudiendo asi eliminar valores que se consideran poco importantes. Se suma lo producido anteriormente por la salida de la puerta de entrada dando como resultado un nuevo estado interno de la celda C_t. Esta parte de la celda puede verse en la figura 4.11 marcada en rojo.
- Puerta de salida: Se encarga de determinar el siguiente estado oculto h_t de la celda. El funcionamiento de esta puerta es el descrito a continuación:

  1. El estado oculto anterior h_{t-1} y la entrada actual X_t se pasan a una función sigmoidea.

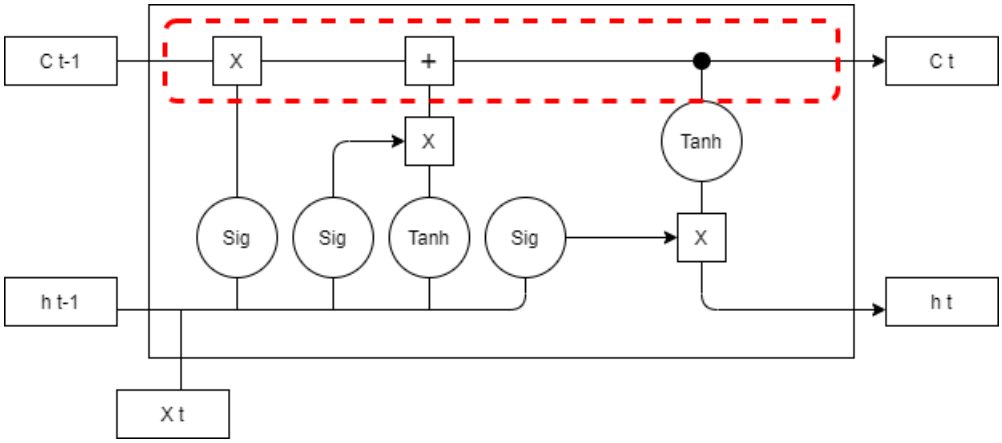

Figura 4.11: Puertas de una celda LSTM: Estado interno de la celda

2. El estado interno de la celda recien calculado se pasa a una función tanh.
3. Los dos resultados anteriores se multiplican dando como resultado el estado interno de la celda.

Esta parte de la celda puede verse en la figura 4.12 marcada en rojo.

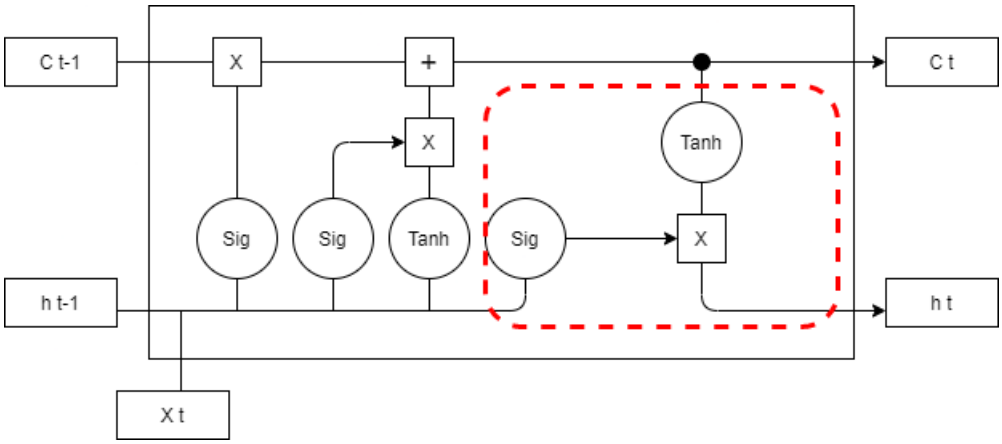

Figura 4.12: Puertas de una celda LSTM: Puerta de salida

Todas las RNN tienen estructura de cadaena de céldas repetidos en forma de red neuronal. En las RNN este módulo que se repite es muy simple pues solo cuenta con una capa tanh. En cambio en las LSTM esta estructura que se repite como una caadena tiene la forma que puede verse en la figura 4.13. A continuación se va a proceder a explicar como se generan las diferentes salidas de cada una de las puertas explicadas anteriormente. En primer lugar es decidir que información es relevante y cual no en cada estado de las celdas. Para ello tal y como se ha explicado anteriormente se usa la puerta de olvido, la cual calcula el valor f_t tal y como se puede ver en la figura 4.14 en rojo.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

El siguiente paso es el calcular el resultado de la puerta de entrada.Para ello se calculan los

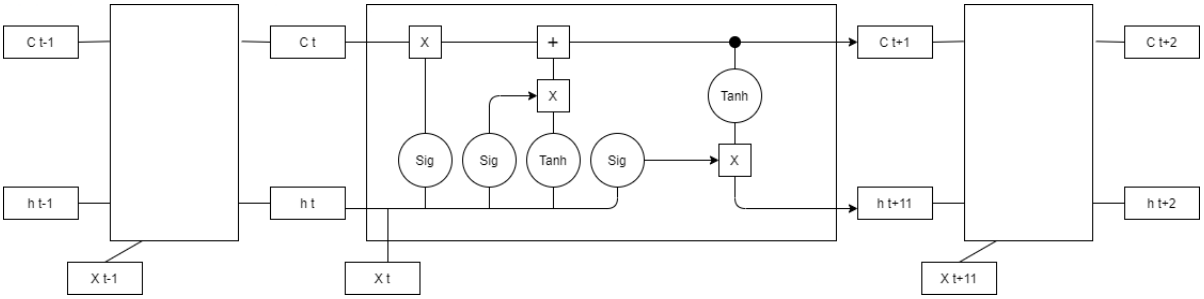

Figura 4.13: Cadena de una LSTM

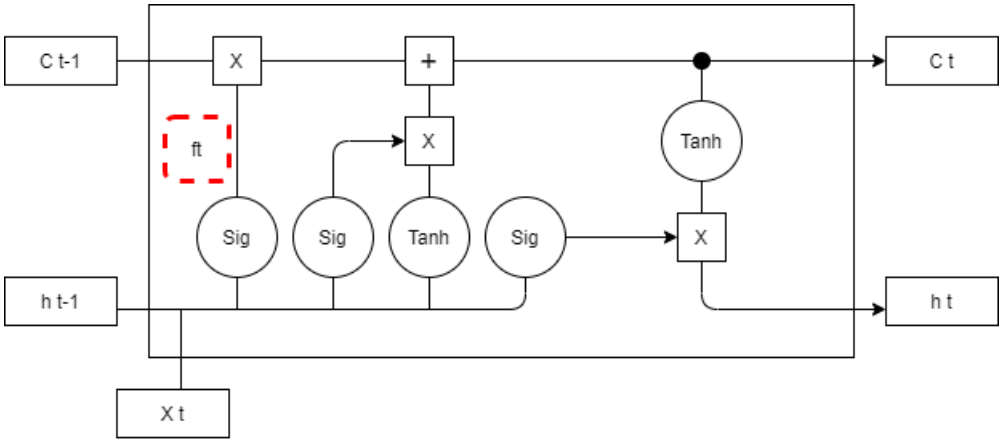

Figura 4.14: Primer paso en el calculo interno de una celda de una LSTM

valores de las salidas de las diferentes funciones de esta puerta explicadas anteriormente que pueden verse en la figura 4.15 en rojo.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$C_t^* = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$

Donde i_t es el resultado de aplicarle a la entrada actual y el estado oculto anterior la sigmoide y C_t^* es el resultado de aplicarles la tanh. El siguiente paso es la actualización del estado interno de la celda C_t tal y como se ha explicado en el apartado dedicado a la puerta .ªtado interno de la celda". Esto puede verse en la figura 4.16 marcada en rojo.

$$C_t = f_t * C_{t-1} + i_t * C_t^*$$

En último lugar se calcula la salida de lac celda h_t, siendo necesario primero calcular el valor de la salida o_t de la sigmoide de la puerta de salida explicada anteriormente. Esto puede verse en la figura 4.17 marcada en rojo.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

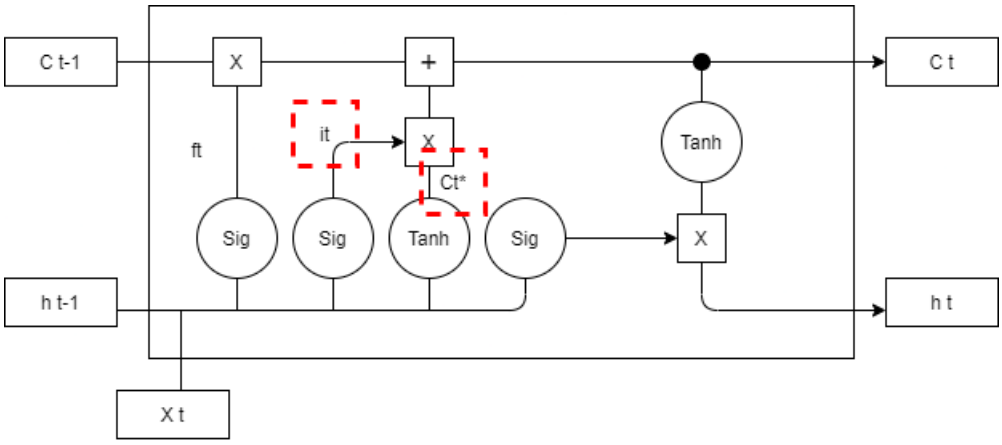

Figura 4.15: Segundo paso en el calculo interno de una celda de una LSTM

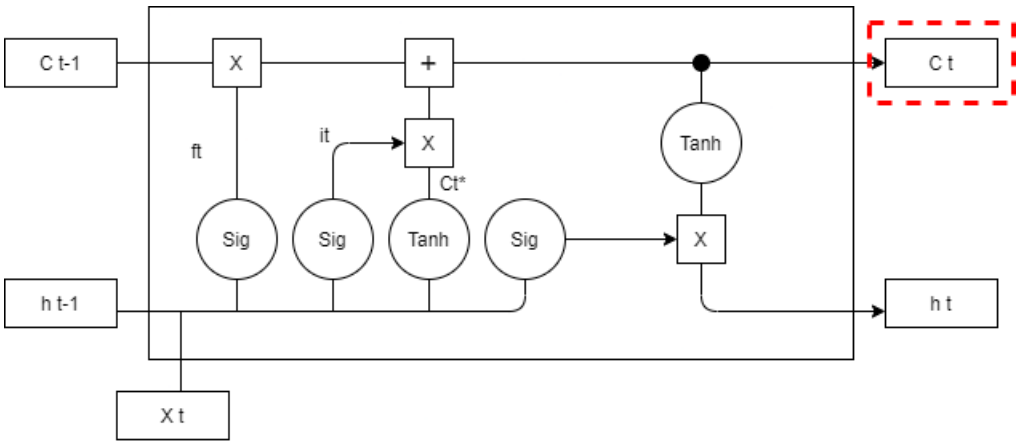

Figura 4.16: Tercer paso en el calculo interno de una celda de una LSTM

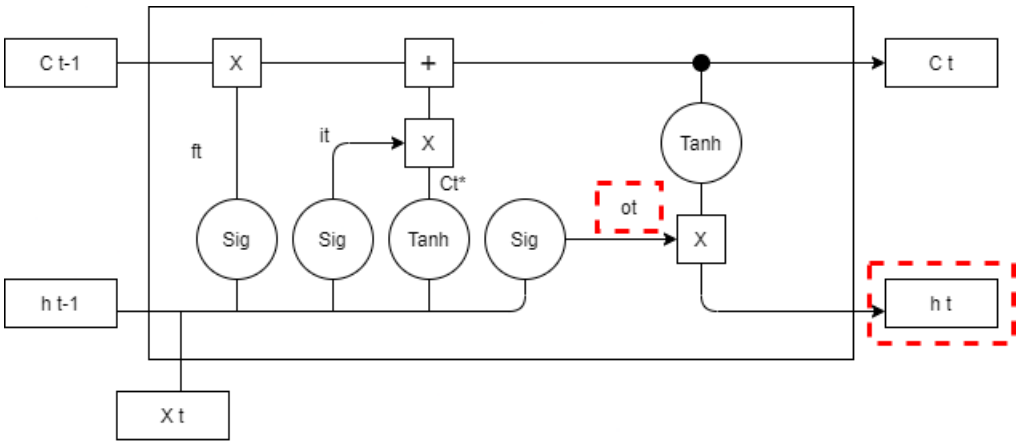

Figura 4.17: Cuarto paso en el calculo interno de una celda de una LSTM

Capítulo 5

Objetivos y metodología

5.1. Objetivo general

El objetivo general de este proyecto, es diseñar un modelo capaz de predecir el flujo de tráfico en dos de las principales carreteras de la ciudad de Zaragoza, el cual tendrá como partida diferentes datos recogidos por cuatro radares situados en las vías de tráfico. Para ello se emplearán técnicas de Machine Learning, concretamente redes neuronales de tipo LSTM para así generar modelos de los algoritmos deseados que posteriormente permitan obtener una solución óptima al problema de la predicción de tráfico.

5.2. Objetivos especificos

Los objetivos específicos a logar durante el desarrollo de este trabajo son los siguientes:

1. Revisar bibliográfica de técnicas y modelos existentes usados en la modelización del tráfico de diferentes vías urbanas.

2. Identificar variables del entorno a tener en cuenta a la hora de modelizar un problema de flujo de tráfico.

3. Realizar un correcto preprocesamiento de la información para posteriormente poder aplicar los modelos escogidos.

4. Realizar un correcto análisis de las series temporales para esocger los modelos que mejor se adapten.

5. Llevar a cabo una comparativa de los diferentes modelos de predicción escogidos en este TFM.

6. Analizar los resultados obtenidos y determinar cual es el mejor modelo para poder realizar predicciones a corto y largo plazo.

7. Determinar que lineas de trabajo futuras serían adecuadas para mejorar el trabajo realizado.

5.3. Metodología

Para el desarrollo de este TFM se ha empleado una metodología ágil denominada Scrum. Scrum permite reducir la complejidad y el tiempo de desarrollo de los proyectos. La metodología Scrum se basa en el desarrollo en una serie de sprints semanales al final de los cuales existe una versión usable del proyecto, es decir, no es necesario esperar al final del desarrollo para observar diferentes resultados. La prioridad de las diferentes tareas se ha establecido en base a la complejidad de las mismas y los requisitos de computación de cada una de ellas, de forma que aquellas más costosas se intercalaban con tareas de caracter teórico o de revisión bibliográfica. El desarrollo de este TFM, se puede dividir en dos fases claramente diferenciadas:

- Fase de Investigación.
- Fase de Modelización y Comparación.

5.3.1. Fase de Investigación

En esta fase se ha realizado una revisión profunda de la bibliografía existente con el objetivo dealcanzar un mayor conocimiento y compresión del problema a modelar, así como de los diferentes algoritmos a emplear para obtener mejores resultados a la hora de emplearlos. También se han identificado que parámetros hay que tener en cuenta a la hora de modelizar un problema de flujo de tráfico en base a los estudios previos existentes. Durante esta fase se ha seleccionado el dataset de mayor calidad en cuanto a cantidad de datos y características disponibles en los mismos.

5.3.2. Fase de Modelización y Comparación

Una vez el dataset ha sido escogido, se han observado los datos que se tienen a disposición y se han establecido que objetivos se quieren lograr es el momento de desarrollar el proyecto haciendo uso de Python y Google Colab. En esta fase se implementan los difernetes modelos planteados con los que se irán realizando pruebas y se irán mejorando según los objetivos a lograr. En último lugar se determinará cual es el mejor de los modelos planteados gracias a una serie de análisis que se realizarán a cada uno de ellos.

Capítulo 6

Solución Propuesta

6.1. Observación del dataset

El primer paso de esta propuesta al igual que en cualquier otro problema de modelización es necesaria una primera carga y observación de los datos (dataset) disponibles para modelizar el problema. Un dataset es una colección de datos con una estructura definida. Los campos que componen cada entrada de este dataset vienen recogidos en la tabla 6.1.

Cuadro 6.1: Campos disponibles en el dataset

Nombre del Campo	Tipo de Campos	Descripción
N_Dia	int64	Día del dataset
Fecha	int64	Fecha en formato "timestamp"
Radar_Id	int64	Identificador único del radar (1-4)
Direccion	Objeto	Nombre de la calle
Lectura	int64	Densidad de tráfico
Year	int64	Año de la medición
Mes	int64	Mes de la medición (1-12)
Dia	int64	Día de la medición (1-7)
Dia_Semana	int64	Dia de la semana de la medición (1-7)
Hora	int64	Hora de la medición (0-23)
Minuto	int64	Minuto de la medición (0-60)
Estacion	int64	Estación del año (1-4)
Dia_Laboral	int64	Determina si es día laboral
Dia_Escolar	int64	Determina si es día escolar
Temp_Media	float64	Temperatura media en ese momento
Temp_Minima	float64	Temperatura mínima de ese día
Temp_Maxima	float64	Temperatura máxima de ese día
Precipitacion_Dia	float64	Precipitaciones de ese día
Vel_Media_Viento	float64	Velocidad media del viento ese día
Evento	int64	Determina si hay un evento ese día

6.2. Preprocesado de la información

En el caso de los datos con los que se contaba se pueden ver como una colección de rasgos que capturan características de un evento (Captura de la densidad de tráfico por parte de un radar) siendo estos rasgos los campos descritos anteriormente.
Al tratarse de modelizar computacionalmente un problema es imprescindible comprender que las máquinas no entienden textos, imágenes o vídeos. Es por ello que el segundo paso a realizar fue el preprocesado de la información disponible para poder permitir a la máquina trabajar con ella.

6.2.1. Datos Perdidos

Es muy común en los datasets la existencia de Missing Values, es decir, valores que se han perdido en la recogida de información o por alguna validación de los datos previa a la obtención de los mismos por nuestra parte. Independientemente de la causa de estos Missing Values siempre es necesario tenerlos en cuenta y realizar un tratamiento adecuado de los mismos. Se puede optar por dos estrategias:

- Eliminar filas con Missing Values
- Estimar los valores que faltan

En este caso, se optó por la estrategia de eliminarlos. El motivo de esta elección se debe a que es una estrategia simple y eficaz en problemas con gran cantidad de datos y pocos Missing Values.

6.2.2. Campos sin utilidad

Otro fenómeno muy común en la modelización computacional es la existencia de campos sin utilidad para el problema que se esta tratando. En este caso tras realizar el análisis correspondiente de todos los campos, se determinó que los campos sin utilidad y los motivos de us eliminación eran:

- N_Dia: Es irrelevante para la predicción temporal el número de día respecto al dataset.
- Fecha: El formato de fecha en "timestamp"no es adecuado en el caso de modelizar series temporales dada su baja utilidad a la hora de representar la información.
- Evento: Es indiferente dada la cantidad de datos la existencia o no de un evento en el momento de la medición ya que una alteración puntual del tráfico no marcará la diferencia en el modelo a realizar.

6.2.3. Fecha en formato estandar

Tal y como se ha descrito anteriormente el objetivo era modelizar el problema haciendo uso de series temporales. Por ello, era necesaria la creación de un campo de fechas en algún formato estandar haciendo uso de los campos separados disponibles. Para ello se optó por el formato estandar "Y-M-D H:m"donde Y es el año de la medición, M el mes, D el día, H la hora y m los minutos. Este formato puede verse en la figura 6.1.

	Date	Year	Mes	Dia	Hora	Minuto	Radar_Id	Direccion	Lectura	Dia_Semana	Estacion	Dia_Laboral	Dia_Escolar	Temp_Media	Temp_Minima
0	2017-1-1 0:0	2017	1	1	0	0	1	Fernando el Catolico (sentido NE)	1	7	4	0	0	1.8	0.6
1	2017-1-1 0:0	2017	1	1	0	0	2	Fernando El Catolico (sentido SW)	2	7	4	0	0	1.8	0.6
2	2017-1-1 0:0	2017	1	1	0	0	3	Avda Goya (sentido SE)	2	7	4	0	0	1.8	0.6
3	2017-1-1 0:0	2017	1	1	0	0	4	Avda Goya (sentido NW)	1	7	4	0	0	1.8	0.6

Figura 6.1: Datos con formato de fecha estandar.

6.2.4. Correlación entre los radares de cada calle

El siguiente paso en el preprocesado de la información fue tener en cuenta la existencia de dos únicas calles con dos radares en cada una. Si fuera posible juntar las mediciones de los dos radares de cada calle en un único radar, la modelización del problema sería mucho mas simple al tener un único conjunto de mediciones para cada calle. Para ello se creó un dataset formado por las mediciones de cada radar en cada fecha tal y como se ve en la figura 6.2.

Date	Fernando el Catolico (sentido NE)	Fernando el Catolico (sentido SW)	Avda Goya (sentido SE)	Avda Goya (sentido NW)
2017-1-1 0:0	1	2	2	1
2017-1-1 0:15	1	1	2	2
2017-1-1 0:30	3	3	4	5
2017-1-1 0:45	3	4	5	5

Figura 6.2: Datos preparados para comprobar la correlación.

Una vez construido este conjunto de datos, se emplearon diferentes métodos para comprobar la correlación entre los datos de los dos radares de cada calle.

Observación de las series temporales de cada calle

El primer método y el más elemental que se empleó fue la observación de las series temporales para comprobar si existía una relación evidente entre ambas. La comparación entre los radares de la calle Fernando el Católico está representada en la figura 6.3 y la comparación entre los radares de la calle Avenida de Goya en la figura 6.4.

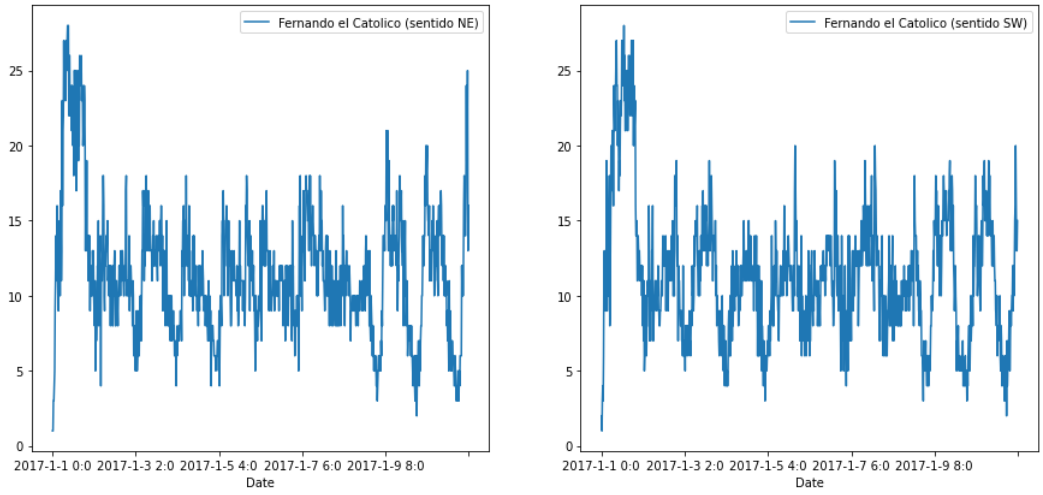

Figura 6.3: Comparación de las series temporales de Fernando el Católico.

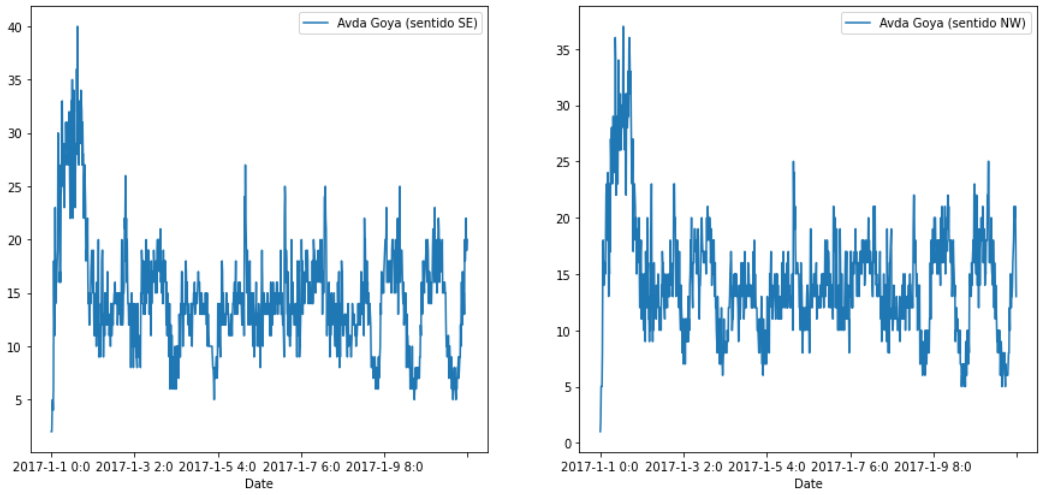

Figura 6.4: Comparación de las series temporales de Avenida de Goya.

A simple vista era evidente la relación existente entre las mediciones realizadas por los radares de cada calle, sin embargo, una simple comprobación visual no es suficiente para tomar la decisión de simplificar el modelo.

Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson es una prueba que se utiliza para medir la relación estadística entre dos variables continuas. Los valores que puede tomar son:

- De -1 a 0: Un valor menor que 0 indica una asociación negativa; es decir, a medida que aumenta el valor de una variable, el valor de otra variable disminuye.
- 0: Un valor de 0 significa que no existe correlación entre las dos variables.
- De 0 a 1: Los valores superiores a 0 indican una asociación positiva.

Los resultados que se obtuvieron pueden verse en la tabla 6.2.

Cuadro 6.2: Resultados obtenidos tras computar el coeficiente de Pearson

Variable A	Variable B	Coeficiente de Pearson	P-valor
F.C. (sentido NE)	F.C.(sentido SW)	0.7947278140916817	0
A.G. (sentido SE)	A.G.(sentido NW)	0.890853025351838	0

Nuevamente se podía observar una relación entre las mediciones de los dos radares de cada calle reafirmando la hipótesis anterior tomada de forma visual.

TLCC - Time Lagged Cross Correlation

Al tratarse de comparar dos series temporales (Una por cada radar), otro de los métodos que se decidió emplear fue la TLCC el cual permite identificar la direccionalidad entre dos señales en base a una relacción lider-seguidor. El objetivo de este método por tanto es extraer la idea de que serie temporal va por delante en base a sus correlaciones cruzadas. Para realizar esta comparación lo que se hace es ir desplazando una serie respecto a la otra y calculando la correlación entre ambas. Si el pico de correlación está en el centro (offset = 0), indica que las dos series de tiempo están más sincronizadas en ese momento. La comparación entre los radares de la calle Fernando el Católico está representada en la figura 6.5 y la comparación entre los radares de la calle Avenida de Goya en la figura 6.6.

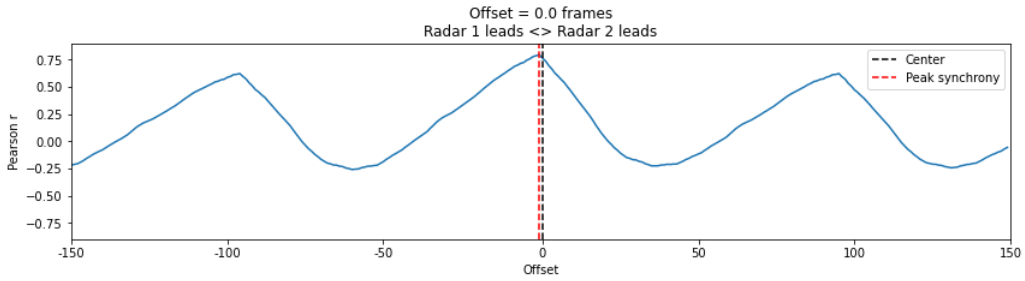

Figura 6.5: Time Lagged Cross Correlation de Fernando el Católico.

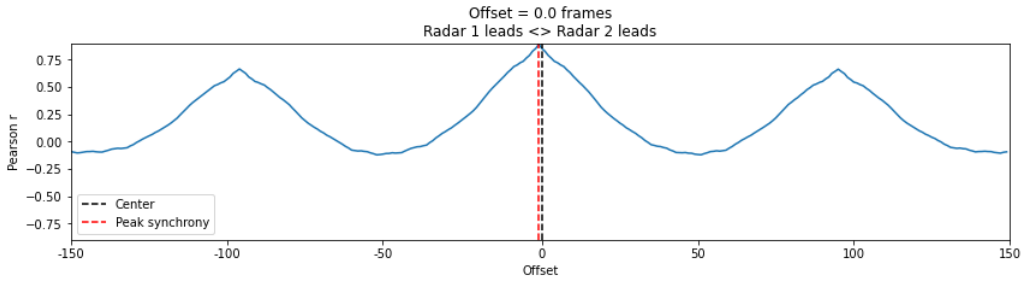

Figura 6.6: Time Lagged Cross Correlation de Avenida de Goya.

Como se puede observar en las gráficas, el momento de máxima correlación entre ambas es un valor muy cercano a 0. Con ello se pudo concluir nuevamente que el modelo era simplificable reduciendo los datos de cada calle a una única serie temporal.

6.2.5. Juntar los datos en dos únicas calles

Una vez se concluyó que los datos de ambos semáforos de cada calle estaban correlacionados el siguiente paso que se tomó fue el unir los datos de forma que solo existieran dos calles y dos radares. El resultado de como resultaron los datos trás esta modificación puede verse en la figura 6.7.

	Date	Year	Mes	Dia	Hora	Minuto	Radar_Id	Direccion	Lectura	Dia_Semana	Estacion	Dia_Laboral	Dia_Escolar	Temp_Media	Temp_Minima
0	2017-1-1 0:0	2017	1	1	0	0	1	Fernando el Catolico	1	7	4	0	0	1.8	0.6
1	2017-1-1 0:0	2017	1	1	0	0	1	Fernando el Catolico	2	7	4	0	0	1.8	0.6
2	2017-1-1 0:0	2017	1	1	0	0	2	Avda Goya	2	7	4	0	0	1.8	0.6
3	2017-1-1 0:0	2017	1	1	0	0	2	Avda Goya	1	7	4	0	0	1.8	0.6

Figura 6.7: Resultado tras quedarse solo con dos calles

De esta forma, el problema que consistía en modelizar el tráfico de cuatro calles diferentes quedó reducido a dos unicas series temporales.

6.3. Creación de un dataset por cada calle

Una vez la información se preprocesó y se determinó que lo correcto era la fusión de las mediciones de cada callem, el siguiente paso que se tomó fue la generación de un dataset independiente para cada una de las calles.

6.3.1. Separación y agrupación de los datos

Para la creación de estos dataset el primer pasó que se realizo fue separar en base a aquellas filas del dataset original que tenían un id de radar 1 y aquellas que tenían un id 2. El resultado para la calle Fernando el Católico puede verse en la figura 6.8 siendo el resultado para la calle Avenida de Goya similar.

	Index	Date	Year	Mes	Dia	Hora	Minuto	Radar_Id	Direccion	Lectura	Dia_Semana	Estacion	Dia_Laboral	Dia_Escolar	Temp_Media	Temp_Minima
0	0	2017-1-1 0:0	2017	1	1	0	0	1	Fernando el Catolico	1	7	4	0	0	1.8	0.6
1	1	2017-1-1 0:0	2017	1	1	0	0	1	Fernando el Catolico	2	7	4	0	0	1.8	0.6
2	4	2017-1-1 0:15	2017	1	1	0	15	1	Fernando el Catolico	1	7	4	0	0	1.8	0.6
3	5	2017-1-1 0:15	2017	1	1	0	15	1	Fernando el Catolico	1	7	4	0	0	1.8	0.6

Figura 6.8: Datos separados de la calle Fernando el Católico

Al haber medidas de los dos radares en cada calle teníamos dos entradas en cada fecha con todos los campos iguales a excepción del campo "Lectura". Por ello, se decidió

agrupar en base a la fecha y sumando las lecturas de ambos radares en el nuevo campo "Lectura". El resultado para la calle Fernando el Católico puede verse en la figura 6.9 siendo el resultado para la calle Avenida de Goya similar.

Date	Lectura sum	Year max	Mes max	Dia max	Hora max	Minuto max	Radar_Id max	Direccion max	Dia_Semana max	Estacion max	Dia_Laboral max	Dia_Escolar max	Temp_Media max	Temp_Minima max	Temp_M max
2017-1-1 0:0	3	2017	1	1	0	0	1	Fernando el Catolico	7	4	0	0	1.8	0.6	
2017-1-1 0:15	2	2017	1	1	0	15	1	Fernando el Catolico	7	4	0	0	1.8	0.6	
2017-1-1 0:30	6	2017	1	1	0	30	1	Fernando el Catolico	7	4	0	0	1.8	0.6	
2017-1-1 0:45	7	2017	1	1	0	45	1	Fernando el Catolico	7	4	0	0	1.8	0.6	

Figura 6.9: Datos agrupados por fecha de la calle Fernando el Católico

De esta forma los datos para el ajuste de nuestro modelo pasaron a ser una serie temporal por cada una de las calles a modelizar.

6.3.2. Correlación entre las columnas de los datasets

Uno de los pasos esenciales antes de la creación de los modelos, fue comprobar la correlación entre las diferentes columnas de los dos datasets de las calles. Para realizar la comprobación, se escogió el coeficiente de correlación de Pearson. La correlación entre el flujo de tráfico y el resto de variables de Fernando el Católico está representada en la figura 6.11 y la correlación entre el flujo de tráfico y el resto de variables de Avenida de Goya en la figura 6.10.

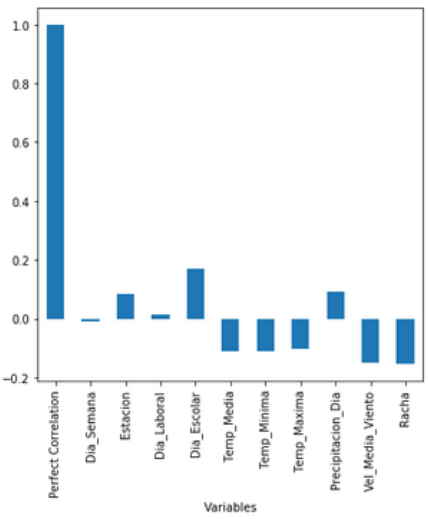

Figura 6.10: Coeficiente de correlación de Fernando el Católico

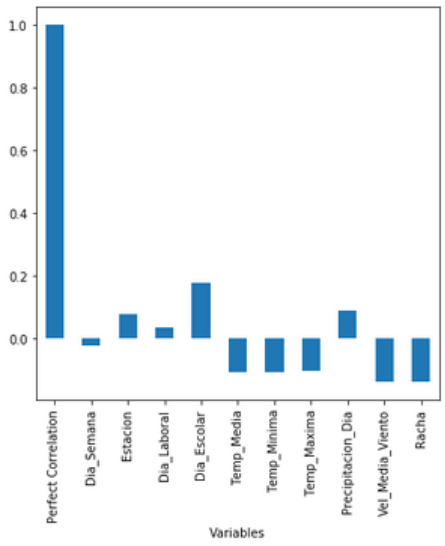

Figura 6.11: Coeficiente de correlación de Avenida de Goya

Gracias a estas gráficas pudimos concluir que no existía una correlación notable entre ninguna de las variables registradas y las mediciones de los radares. De esta forma el problema original de modelizar el flujo de tráfico en base a las mediciones y una serie de variables quedó reducido a modelizar en base al número de mediciones.

6.3.3. Datasets finales

Tras haber realizado los pasos previos, se exportaron los datasets finales para cada calle, los cuales, tenían solo las columnas referidas a la fecha de medición y el flujo de tráfico en ese instante. El dataset de Fernando el Católico puede verse en la figura 6.12 y el de Avenida de Goya en la figura 6.13.

date	total_cars	year	month	day	hour	minute
2017-01-01 00:00:00	3	2017	1	1	0	0
2017-01-01 00:15:00	2	2017	1	1	0	15
2017-01-01 00:30:00	6	2017	1	1	0	30

Figura 6.12: Dataset de Fernando Católico

date	total_cars	year	month	day	hour	minute
2017-01-01 00:00:00	3	2017	1	1	0	0
2017-01-01 00:15:00	4	2017	1	1	0	15
2017-01-01 00:30:00	9	2017	1	1	0	30

Figura 6.13: Dataset de Avenida de Goya

6.4. Análisis de las series temporales

Uno de los pasos cruciales antes de la creación y ajuste de los modelos fue el analizar las series temporales.

6.4.1. Análisis de estacionariedad por variables temporales

El primero de los análisis que se llevó a cabo fue el observar las series temporales así como la media del flujo de coches por años, meses, y días. La media por años de Fernando el Católico puede verse en la figura 6.14 y la de Avenida de Goya en la figura 6.15. Como

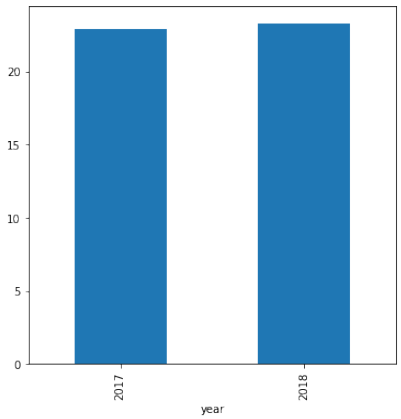

Figura 6.14: Media por años de Fernando Católico

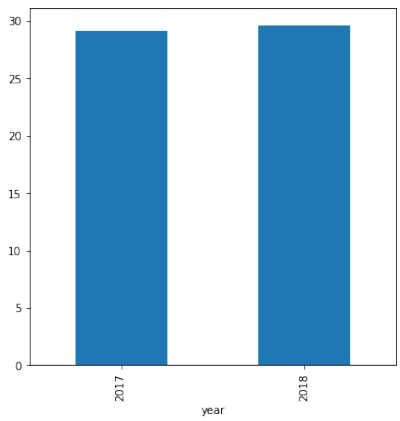

Figura 6.15: Media por años de Avenida de Goya

puede verse en ambas figuras la media de ambos años de los cuales se tienen medicones es muy similar, siendo este el primer indicativo de la existencia de estacionariedad en ambos datasets. A continuación se procedió a analizar la media por meses de ambas series temporales. La media por meses de Fernando el Católico puede verse en la figura 6.16 y la de Avenida de Goya en la figura 6.17. Como puede verse en ambas figuras la media por

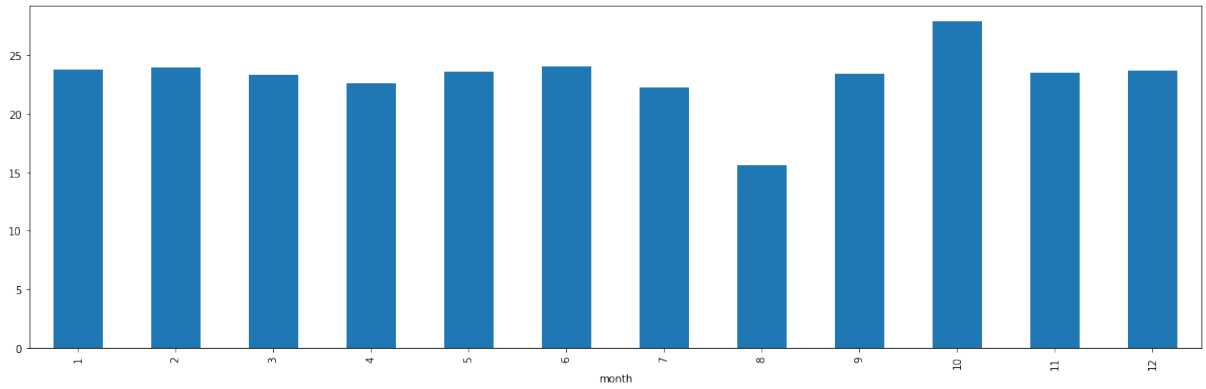

Figura 6.16: Media por meses de Fernando Católico

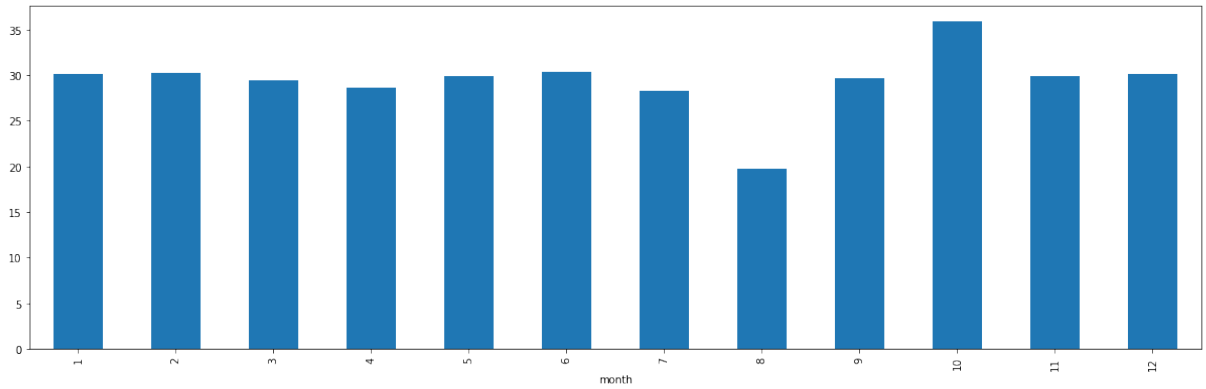

Figura 6.17: Media por meses de Avenida de Goya

meses es bastante similar en ambos casos, aunque en este caso, puede verse una fluctuación del flujo de tráfico que decrece a lo largo del verano y vuelve a aumentar en los meses de septiembre y octubre. A pesar de esta pequeña fluctuación se puede ver un segundo indicativo de la existencia de estacionariedad en ambos datasets. El último de los pasos a realizar en este análisis visual previo fue obserar la media respecto a los días del mes. La media por días de Fernando el Católico puede verse en la figura 6.18 y la de Avenida de Goya en la figura 6.19. Nuevamente podemos ver en amabas figuras que apenas hay variación de la media de unos días respecto a otros, siendo este el tercer indicativo de la existencia de estacionariedad en ambos datasets.

6.4.2. Análisis de estacionariedad por media y varianza

El siguiente paso a tomar fue el estudio de la estacionalidad mediante la observación de la fluctuación de la media y la varianza en diferentes subconjuntos de datos, así como

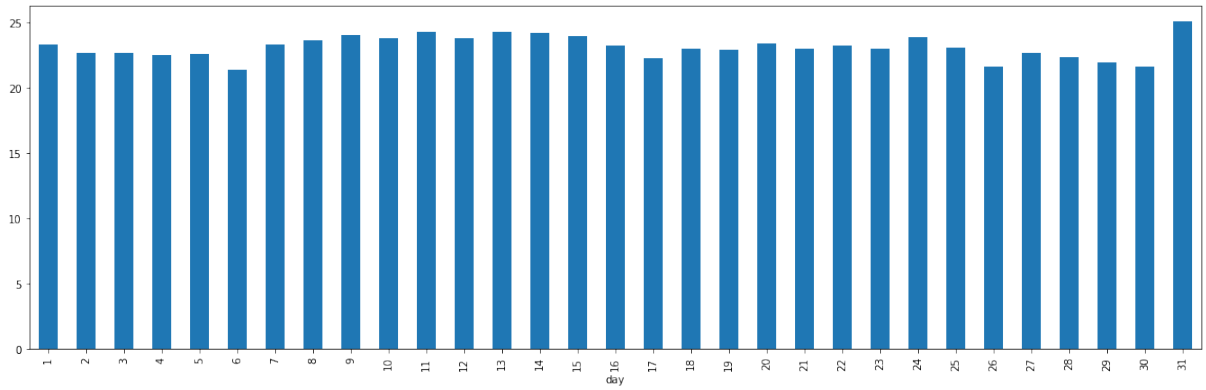

Figura 6.18: Media por días de Fernando Católico

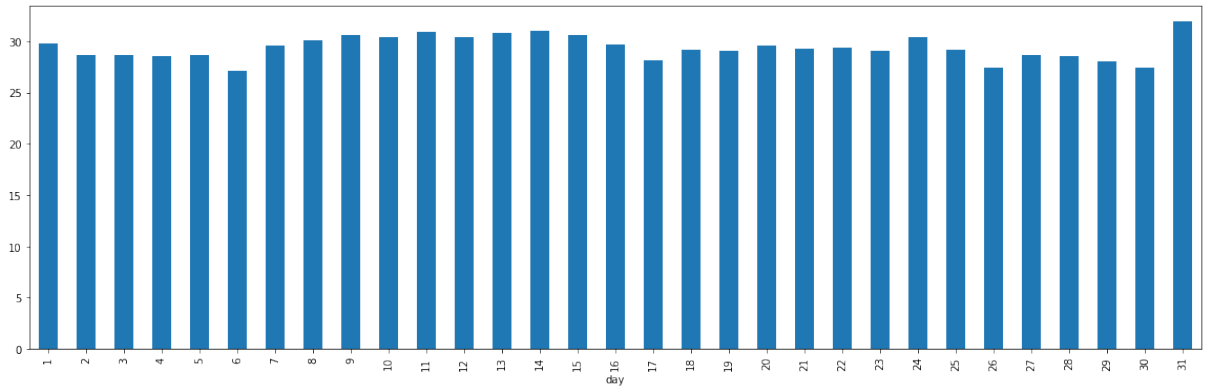

Figura 6.19: Media por días de Avenida de Goya

mediante la observación de la gráfica de distribución de la densidad de flujo. En primer lugar se obervó la gráfica de distribución de la densidad de flujo para la calle Fernando el Católico, representada en la figura 6.20.

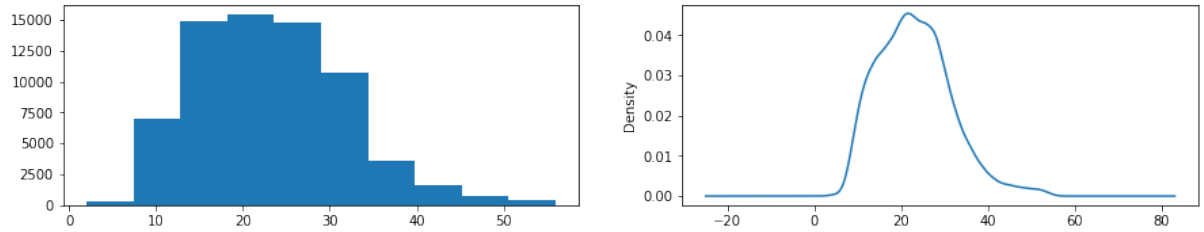

Figura 6.20: Distribución de la densidad de flujo de Fernando el Católico

Como se ve en la figura, la distribución de densidad de las medidas de flujo se concentra claramente sobre el 20 siendo esto un indicativo de la estacionariedad del dataset. En segundo lugar se obervó la gráfica de distribución de la densidad de flujo para la calle Avenida de Goya, representada en la figura 6.21. Como se ve en la figura, la distribución de densidad de las medidas de flujo se concentra claramente sobre el 25 siendo esto un indicativo de la estacionariedad del dataset. Una vez analizada la distribución de densidad de las medidas de flujo, se procedió a dividir cada dataset en tres partes para observar

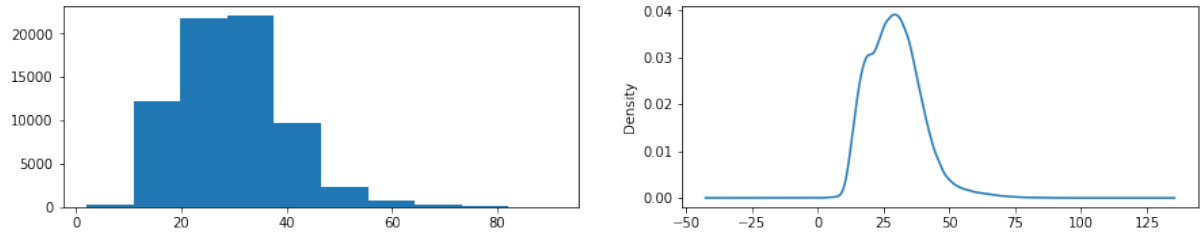

Figura 6.21: Distribución de la densidad de flujo de Avenida de Goya

las variaciones en la media y la varianza. En primer lugar se realizaron los cálculos para la calle Fernando el Católico, visibles en la tabla 6.3. Como conclusión a estos resultados

Cuadro 6.3: Media y varianza de Fernando el Católico

Número de fragmento	Media	Varianza
1	22.255940	64.176324
2	23.848872	77.538048
3	23.226551	76.880958

se saca que existe estacionariedad pues la media y varianza no varían significativamente entre los respectivos fragmentos de la serie temporal. En segundo lugar se realizaron los cálculos para la calle Avenida de Goya, visibles en la tabla 6.4. Al igual que en el caso

Cuadro 6.4: Media y varianza de Avenida de Goya

Número de fragmento	Media	Varianza
1	28.193268	86.551428
2	30.317536	110.504521
3	29.579872	112.507170

anterior, como conclusión a estos resultados se saca que existe estacionariedad pues la media y varianza no varían significativamente entre los respectivos fragmentos de la serie temporal.

6.4.3. Análisis de estacionariedad por el test de Dickey-Fuller

La Prueba de Dickey-Fuller busca determinar la existencia o no de estacionariedad en una serie de tiempo. La hipótesis nula de esta prueba es que no existe estacionariedad. La prueba de Dickey-Fuller se usa generalmente en econometría para verificar si hay una tendencia que cambia con la serie temporal. La particularidad de la prueba Dickey-Fuller es que es la herramienta más fácil de usar en comparación con otras pruebas más complejas, estas pruebas también prueban si existen tendencias en los datos.

En primer lugar, se realizó el test para la calle Fernando el Católico, donde se obtuvieron los resultados observables en la tabla 6.5. La interpretación de los mismos sería que nuestro valor p es definitivamente inferior a 0,5 y es incluso inferior a 0,01, por lo que podemos decir con bastante confianza que podemos rechazar la nulidad (raíz unitaria,

datos no estacionarios) y podemos asumir que nuestros datos son estacionarios. Además, nuestra ADF es mucho menor que nuestro valor de confianza del 1 % de -3,43, por lo que tenemos otra confirmación de que podemos rechazar la nulidad.

Cuadro 6.5: Test Dickey-Fuller para Fernando el Católico

Parámetro	Resultado
ADF	-26.051546243016958
P-Valor	0.0
Valor Crítico 1 %	-3.4304439143951897
Valor Crítico 5 %	-2.861581508491847
Valor Crítico 10 %	-2.5667920935524835

En segundo lugar, se realizó el test para la calle Avenida de Goya, donde se obtuvieron los resultados observables en la tabla 6.6. La interpretación de los mismos sería que nuestro valor p es definitivamente inferior a 0,5 y es incluso inferior a 0,01, por lo que podemos decir con bastante confianza que podemos rechazar la nulidad (raíz unitaria, datos no estacionarios) y podemos asumir que nuestros datos son estacionarios. Además, nuestra ADF es mucho menor que nuestro valor de confianza del 1 % de -3,43, por lo que tenemos otra confirmación de que podemos rechazar la nulidad.

Cuadro 6.6: Test Dickey-Fuller para Avenida de Goya

Parámetro	Resultado
ADF	-23.487849326864115
P-Valor	0.0
Valor Crítico 1 %	-3.4304439143951897
Valor Crítico 5 %	-2.861581508491847
Valor Crítico 10 %	-2.5667920935524835

Una vez analizados estos tests y teniendo en cuenta los realizados anteriormente, se conlcuyó que las series temporales eran series estacionarias, es decir, series cuya media y varianza no varía a lo largo del tiempo y que tampoco siguen una tendencia. Este resultado es muy significativo pues predecir una serie temporal es una tarea mucho más sencilla si se comportaba de forma similar en el pasado pues podemos asumir que en el futuro lo hará de forma muy similar. La mayoría de algoritmos inteligentes que se emplean para la predicción de series temporales presuponen que las series son estacionarias. La comparativa entre estacionariedad y no estacionariedad se puede ver en 6.22.

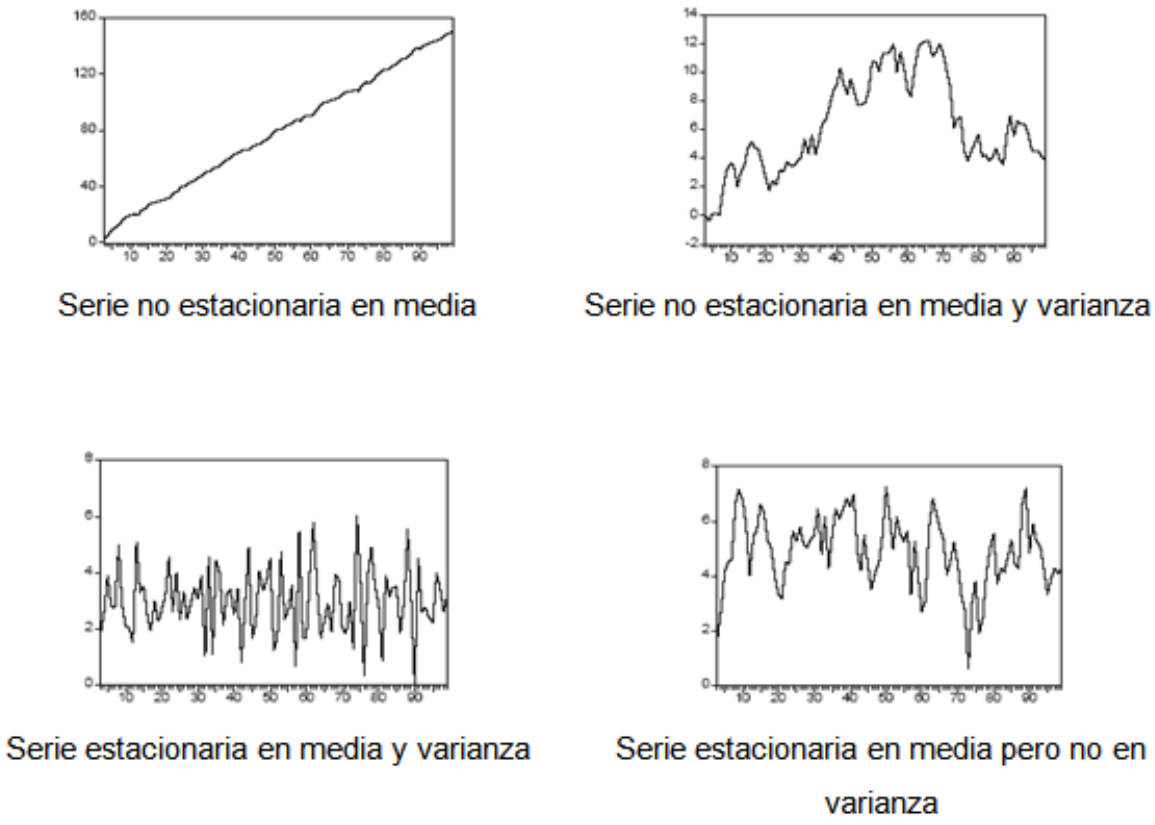

Figura 6.22: Estacionariedad contra no estacionariedad

6.5. Estructura del Desarrollo y Estrategia de Entrenamiento

Con el objetivo de poder realizar al completo el proyecto del TFM, se ha escogido el apoyo de plataformas en la nube pues estas reducen el tiempo de preparación del entorno de desarrollo al contar con gran cantidad de software ya instalado y listo para emplearse. Otra ventaja de emplear este tipo de plataformas es la portabilidad y posibilidad de implementación y testeo desde cualquier terminal con acceso a internet.

La herramienta en la nube escogida para el desarrollo del TFM es Google Colab, siendo esta la plataforma de desarrollo en la nube más empleada actualmente. Google Colab es empleado para realizar todo tipo de proyectos de Machine Learning ya que ofrece capacidad de cómputo y almacenamiento temporal permitiendo así la creación y el almacenamiento de modelos.

Dadas las características de los modelos que se van a emplear, se realizarán pruebas con el objetivo de verificar su correcto funcionamiento. Es por ello que durante el desarrollo de todo el trabajo, se mostrará tanto el código como las gráficas de los diferentes resultados y comparativas que se vayan obteniendo.

Capítulo 7

Desarrollo de los modelos

Este capítulo recoge los modelos implementados, sus partes y el proceso que llevo a su selección. Al tratarse de dos series temporales muy similares, se han empleado los mismos modelos para ambas calles con un ajuste de los parámetros independiente para cada una de ellas (Un entrenamiento independiente para cada una). En primer lugar se van a explicar las métricas empleadas para determinar que modelos son mejores.

7.1. Métricas empleadas

Las métricas empleadas para comparar modelos y para refinar desde un modelo incial hasta el empleado son dos:

- MAE: Esta métrica denominada Error Absoluto Medio, mide la media de los errores en el conjunto de predicciones sin tener en cuenta la dirección de los mismos. Solo puede usarse sobre las diferencias de la predicción y la observación real pues las diferencias individuales de cada instante de tiempo tienen el mismo peso. Se calcula de la siguiente forma:

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

  Donde y_j es la medición real en el isntante j y $\hat{y}_j$ es la predicción en el isntante j.

- Loss Function: En un algoritmo inteligente la función que se emplea para evaluar la solución se denomina función objetivo. El objetivo en los algoritmos de ML es minimizar el error, es decir, minimizar la función de pérdida cuyo valor calculado se denomina "loss".

Como conclusión, cuanto menor MAE y loss tengan nuestros modelos tras el entrenamiento, mejor precisión tendran para predecir las series temporales. Es por ello, que el proceso para crear los modelos consistirá en partir de un modelo base e ir modificandolo para reducir ambas métricas hasta alcanzar márgenes de error que consideremos aceptables para la predicción del flujo de tráfico.

7.2. Primer modelo construido: RNN

El primero de los modelos construido se trata de una RNN, explicada con detalle en el Capítulo 4. A continuación se expone el proceso de creación del modelo y los refinamientos que se fueron realizando hasta dar con un resultado adecuado.

7.2.1. Proceso de Construcción del Modelo

En primer lugar se construyo el primero de los modelos RNN el cual puede verse en la figura 7.1.

```
model1 = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),input_shape=[None]),
  tf.keras.layers.SimpleRNN(30, return_sequences=True),
  tf.keras.layers.Dense(1),
])

optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)
model1.compile(loss=tf.keras.losses.Huber(),optimizer=optimizer,metrics=["mae"])
model1.summary()
```

Figura 7.1: Primer modelo de RNN

Este modelo esta formado por las siguientes capas:

1. Lambda: Esta capa se emplea para aplicar a cada entrada la función lambda especificada, siendo en este caso devolviendo un tensor con un eje de longitud 1.

2. RNN: La segunda de las capas empleadas es una capa RNN con una profundidad de 30.

3. Dense: La última de las capas es una capa Fully Connected la cual nos vale para determinar una única salida en este caso, siendo esta salida la predicción del modelo.

Tras entrenar este modelo se vio que ambas métricas eran muy elevadas y por tanto se decidió refinar el modelo añadiendo y modificando capas dando lugar al segundo modelo RNN implementado, el cual puede verse en la figura 7.2.

```
model2 = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),input_shape=[None]),
  tf.keras.layers.SimpleRNN(30, return_sequences=True),
  tf.keras.layers.SimpleRNN(60),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)
model2.compile(loss=tf.keras.losses.Huber(),optimizer=optimizer,metrics=["mae"])
model2.summary()
```

Figura 7.2: Segundo modelo de RNN

Este modelo esta formado por las siguientes capas (Solo se explican las que se diferencian del anterior y los cambios en las ya existentes):

1. RNN: Se añade esta tercera capa RNN con una profundidad de 60.

2. Lamda: Se añade esta nueva función de salida que se encarga de aplicar la función lambda

$$lambda\ \mathbf{x}{:}x * 100$$

   La cual se aplica a todos los datos que pasan por la red.

La comparativa de la métrica MAE puede verse en la figura 7.3 y la métrica loss en la figura 7.4.

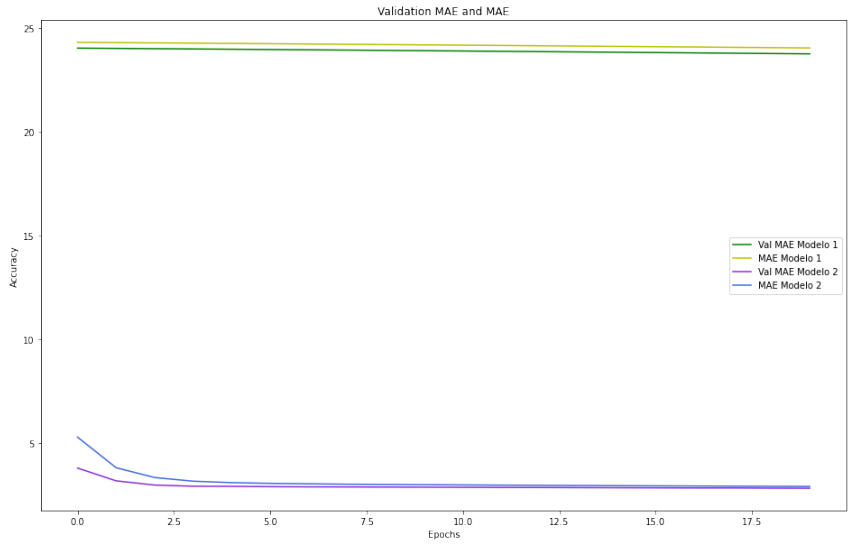

Figura 7.3: Comparativa del MAE entre el modelo 1 y 2 de RNN

Como puede verse en ambas figuras el segundo modelo implementado es muy superior al primero en cuanto a precisión. A pesar de ello, se decidió realizar un tercer modelo que mejorase las capas de salida del mismo. Este tercer modelo desarrollado puede verse en la figura 7.5. Este modelo esta formado por las siguientes capas (Solo se explican las que se diferencian del anterior y los cambios en las ya existentes):

1. RNN: La primera de las RNN se ha modificado para aumentar su profundidad a 150.

2. RNN: La segunda de las RNN se ha modificado tambien para aumentar su profundidad a 150 al igual que la anterior.

3. Dense: Se ha añadido una primera capa Fully Connected de 128 unidades (Este entero determina la dimensión del espacio de salida).

4. Dense: Se ha añadido una primera capa Fully Connected de 56 unidades (Este entero determina la dimensión del espacio de salida) para ir disminuyendo la dimensión de salida gradualmente como una especie de embudo hasta la siguiente capa de 1 sola unidad.

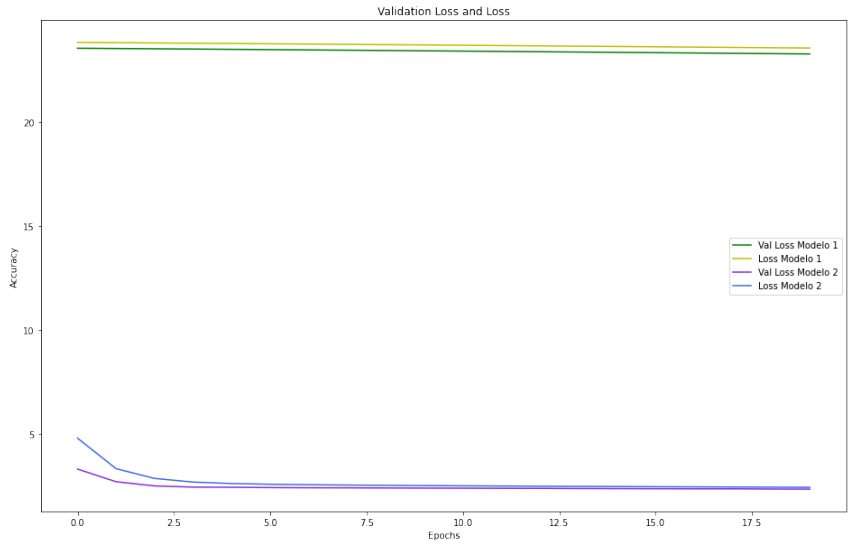

Figura 7.4: Comparativa del loss entre el modelo 1 y 2 de RNN

```
model3 = tf.keras.models.Sequential([
  tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1),input_shape=[None]),
  tf.keras.layers.SimpleRNN(150, return_sequences=True),
  tf.keras.layers.SimpleRNN(150),
  tf.keras.layers.Dense(128),
  tf.keras.layers.Dense(56),
  tf.keras.layers.Dense(1),
  tf.keras.layers.Lambda(lambda x: x * 100.0)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=1e-8, momentum=0.9)
model3.compile(loss=tf.keras.losses.Huber(),optimizer=optimizer,metrics=["mae"])
model3.summary()
```

Figura 7.5: Tercer modelo de RNN

Este modelo se comparó con el modelo 2 para determinar cual usar. La comparativa de la métrica MAE puede verse en la figura 7.6 y la métrica loss en la figura 7.7. Como puede verse en ambas figuras, las métricas son prácticamente idénticas siendo ligeramente menor las del tercer modelo a medida que avanza el entrenamiento, es por ello que se concluye que este tercer modelo es el adecuado para realizar las predicciones temporales con ambas series.

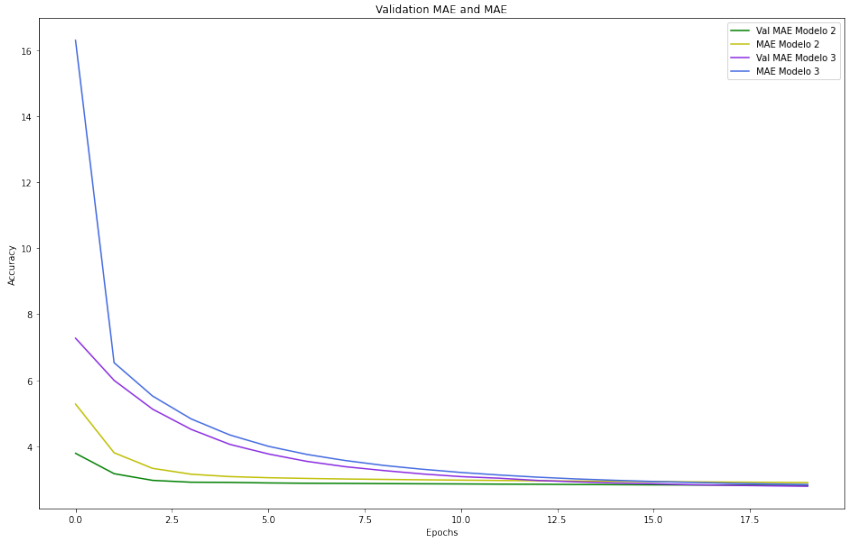

Figura 7.6: Comparativa del MAE entre el modelo 2 y 3 de RNN

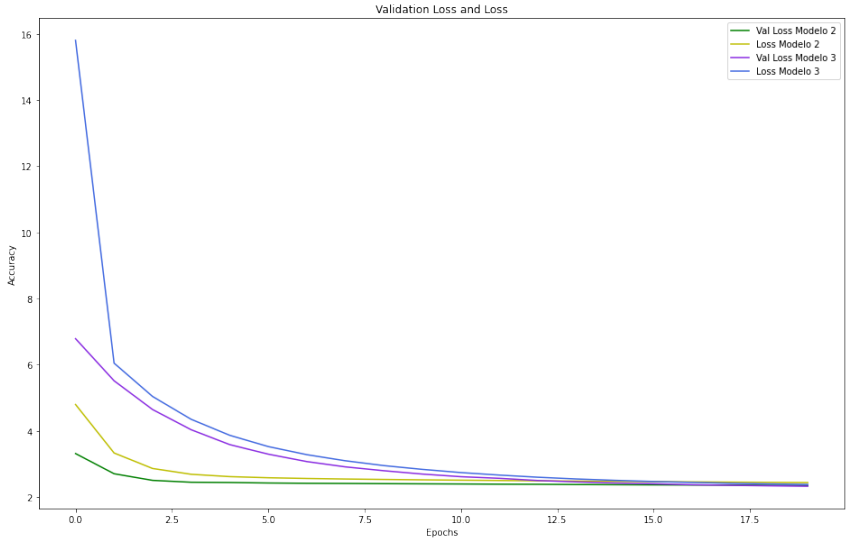

Figura 7.7: Comparativa del loss entre el modelo 2 y 3 de RNN

7.3. Segundo modelo construido: LSTM

El segundo de los modelos construido se trata de una lstm, explicada con detalle en el Capítulo 4. A continuación se expone el proceso de creación del modelo y los refinamientos que se fueron realizando hasta dar con un resultado adecuado.

7.3.1. Proceso de Construcción del Modelo

En primer lugar se construyo el primero de los modelos LSTM el cual puede verse en la figura 7.8.

```
model1 = tf.keras.models.Sequential()
model1.add(tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1), input_shape=[None]))
model1.add(tf.keras.layers.LSTM(30, return_sequences = True))
model1.add(tf.keras.layers.Dense(1))

model1.compile(loss=tf.keras.losses.Huber(),optimizer="Adam",metrics=["mae"])
model1.summary()
```

Figura 7.8: Primer modelo de LSTM

Este modelo esta formado por las siguientes capas:

1. Lambda: Esta capa se emplea para aplicar a cada entrada la función lambda especificada, siendo en este caso devolviendo un tensor con un eje de longitud 1.

2. LSTM: La segunda de las capas empleadas es una capa LSTM con una profundidad de 30.

3. Dense: La última de las capas es una capa Fully Connected la cual nos vale para determinar una única salida en este caso, siendo esta salida la predicción del modelo.

Tras entrenar este modelo se vio que ambas métricas eran muy elevadas y por tanto se decidió refinar el modelo añadiendo y modificando capas dando lugar al segundo modelo LSTM implementado, el cual puede verse en la figura 7.9.

```
model2 = tf.keras.models.Sequential()
model2.add(tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1), input_shape=[None]))
model2.add(tf.keras.layers.LSTM(50, return_sequences = True))
model2.add(tf.keras.layers.LSTM(80, return_sequences = True))
model2.add(tf.keras.layers.Dense(64))
model2.add(tf.keras.layers.Dense(1))

model2.compile(loss=tf.keras.losses.Huber(),optimizer="Adam",metrics=["mae"])
model2.summary()
```

Figura 7.9: Segundo modelo de LSTM

Este modelo esta formado por las siguientes capas (Solo se explican las que se diferencian del anterior y los cambios en las ya existentes):

1. LSTM: Se modifica la profundidad de la primera capa LSTM de 30 a 50.

2. LSTM: Se añade esta tercera capa LSTM con una profundidad de 80.

3. Dense: Se ha añadido una primera capa Fully Connected de 64 unidades (Este entero determina la dimensión del espacio de salida).

La comparativa de la métrica MAE puede verse en la figura 7.10 y la métrica loss en la figura 7.11.

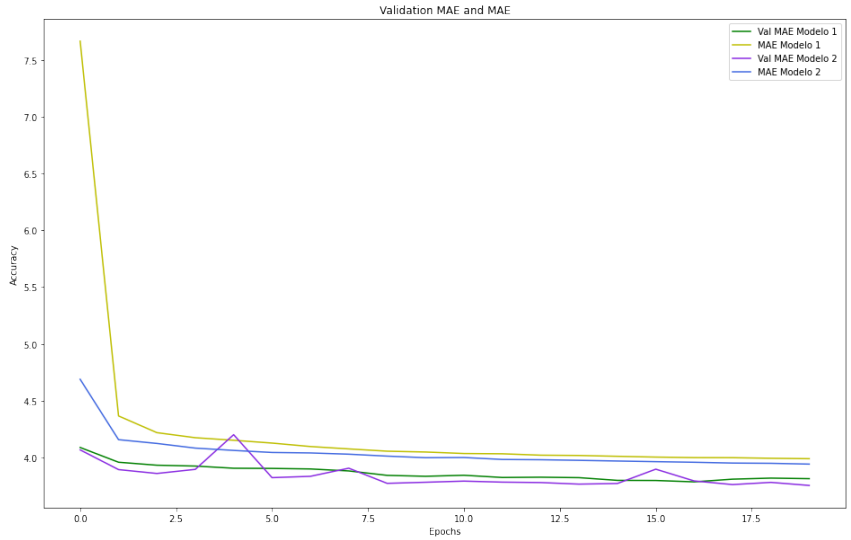

Figura 7.10: Comparativa del MAE entre el modelo 1 y 2 de LSTM

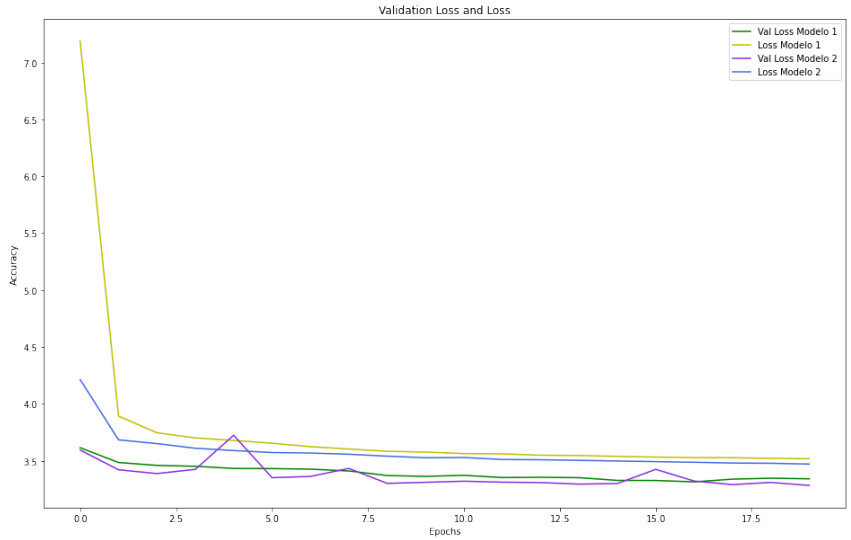

Figura 7.11: Comparativa del loss entre el modelo 1 y 2 de LSTM

Como puede verse en ambas figuras el segundo modelo implementado es superior al primero en cuanto a precisión. A pesar de ello, se decidió realizar un tercer modelo que mejorase las capas de salida del mismo y las LSTM existentes. Este tercer modelo desarrollado puede verse en la figura 7.12. Este modelo esta formado por las siguientes capas (Solo se explican las que se diferencian del anterior y los cambios en las ya existentes):

```
model3 = tf.keras.models.Sequential()
model3.add(tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=-1), input_shape=[None]))
model3.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(150, return_sequences = True)))
model3.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(200)))
model3.add(tf.keras.layers.Dropout(0.3))
model3.add(tf.keras.layers.Dense(128))
model3.add(tf.keras.layers.Dense(64))
model3.add(tf.keras.layers.Dense(1))
```

Figura 7.12: Tercer modelo de LSTM

1. LSTM: La primera de las LSTM se ha modificado para aumentar su profundidad a 150 y se le ha añadido un wrapper Bidirectional.

2. LSTM: La segunda de las LSTM se ha modificado para aumentar su profundidad a 200 y se le ha añadido un wrapper Bidirectional.

3. Dropout: Con un rate de 0.3. Pone aleatoriamente las unidades de entrada a 0 con un rate en cada paso durante el tiempo de entrenamiento, lo que ayuda a evitar el sobreajuste.

4. Dense: Se ha añadido una primera capa Fully Connected de 128 unidades (Este entero determina la dimensión del espacio de salida) para ir disminuyendo la dimensión de salida gradualmente como una especie de embudo hasta la siguiente capa de 64 unidades y luego a la de 1.

Este modelo se comparó con el modelo 2 para determinar cual usar. La comparativa de la métrica MAE puede verse en la figura 7.13 y la métrica loss en la figura 7.14. Como puede verse en ambas figuras, las métricas del tercer modelo son considereablemente mejores, es por ello que se concluye que este tercer modelo es el adecuado para realizar las predicciones temporales con ambas series.

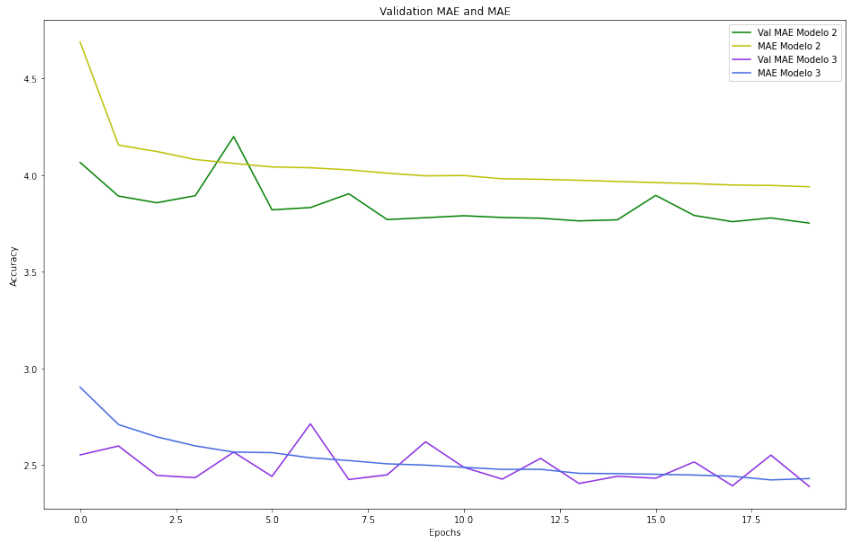

Figura 7.13: Comparativa del MAE entre el modelo 2 y 3 de LSTM

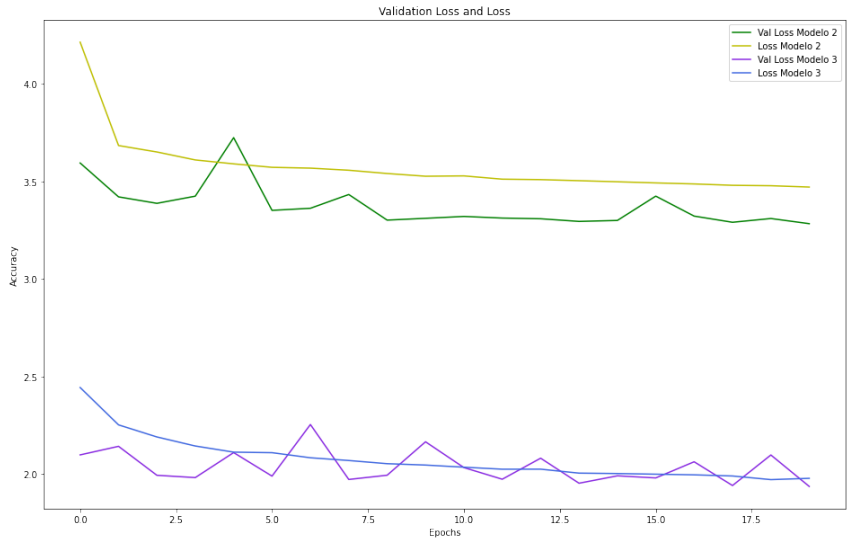

Figura 7.14: Comparativa del loss entre el modelo 2 y 3 de LSTM

7.4. Tercer modelo construido: GRU

El tercero de los modelos construido se trata de una GRU. A continuación se expone el proceso de creación del modelo y los refinamientos que se fueron realizando hasta dar con un resultado adecuado.

7.4.1. Proceso de Construcción del Modelo

En primer lugar se construyo el primero de los modelos GRU el cual puede verse en la figura 7.15.

```
model1 = tf.keras.models.Sequential()
model1.add(tf.keras.layers.GRU (units = 16, return_sequences = True, input_shape = [trainX.shape[1], trainX.shape[2]]))
model1.add(tf.keras.layers.Dense(units = 1))

model1.compile(loss = tf.keras.losses.Huber(), optimizer= 'adam',metrics=['mae'])
model1.summary()
```

Figura 7.15: Primer modelo de GRU

Este modelo esta formado por las siguientes capas:

1. GRU: Primera capa GRU con una profundidad de 16
2. Dense: La última de las capas es una capa Fully Connected la cual nos vale para determinar una única salida en este caso, siendo esta salida la predicción del modelo.

Tras entrenar este modelo se vio que ambas métricas eran muy elevadas y por tanto se decidió refinar el modelo añadiendo y modificando capas dando lugar al segundo modelo GRU implementado, el cual puede verse en la figura 7.16.

```
model2 = tf.keras.models.Sequential()
model2.add(tf.keras.layers.GRU (units = 150, return_sequences = True, input_shape = [trainX.shape[1], trainX.shape[2]]))
model2.add(tf.keras.layers.Dropout(0.2))
model2.add(tf.keras.layers.Dense(units = 128))
model2.add(tf.keras.layers.Dense(units = 64))
model2.add(tf.keras.layers.Dense(units = 1))

model2.compile(loss = tf.keras.losses.Huber(), optimizer= 'adam',metrics=['mae'])
model2.summary()
```

Figura 7.16: Segundo modelo de GRU

Este modelo esta formado por las siguientes capas (Solo se explican las que se diferencian del anterior y los cambios en las ya existentes):

1. GRU: Se modifica la profundidad de la primera capa GRU de 16 a 150.

2. Dropout: Con un rate de 0.2. Pone aleatoriamente las unidades de entrada a 0 con un rate en cada paso durante el tiempo de entrenamiento, lo que ayuda a evitar el sobreajuste.

3. Dense: Se ha añadido una primera capa Fully Connected de 128 unidades (Este entero determina la dimensión del espacio de salida).

4. Dense: Se ha añadido una primera capa Fully Connected de 64 unidades (Este entero determina la dimensión del espacio de salida) para ir disminuyendo la dimensión de salida gradualmente como una especie de embudo hasta la siguiente capa de 1 sola unidad.

La comparativa de la métrica MAE puede verse en la figura 7.17 y la métrica loss en la figura 7.18.

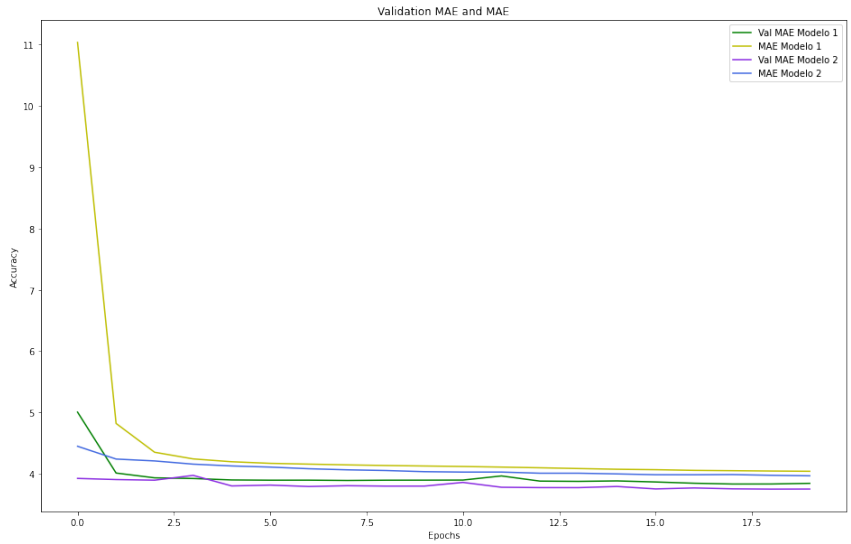

Figura 7.17: Comparativa del MAE entre el modelo 1 y 2 de GRU

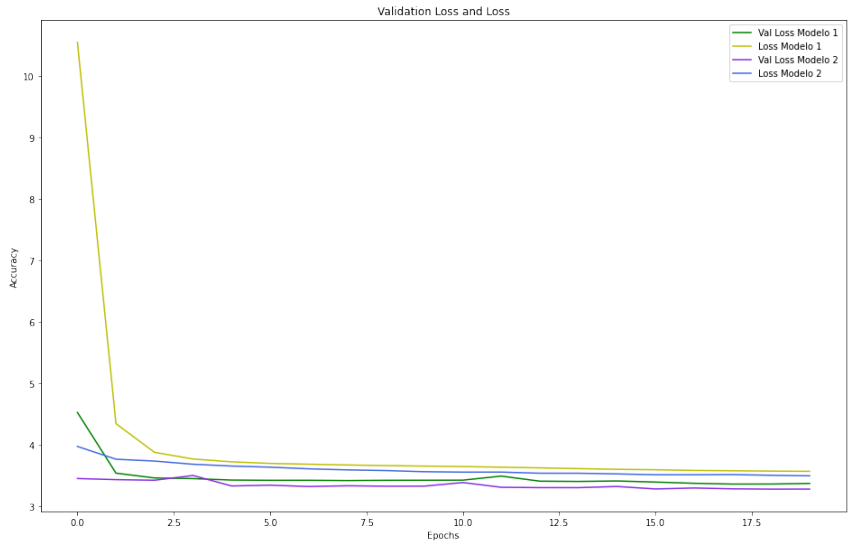

Figura 7.18: Comparativa del loss entre el modelo 1 y 2 de GRU

Como puede verse en ambas figuras el segundo modelo implementado es superior al primero en cuanto a precisión. A pesar de ello, se decidió realizar un tercer modelo que mejorase las capas de salida del mismo y las GRU existentes. Este tercer modelo desarrollado puede verse en la figura 7.19.

```
model3 = tf.keras.models.Sequential()
model3.add(tf.keras.layers.GRU (units = 150, return_sequences = True, input_shape = [trainX.shape[1], trainX.shape[2]]))
model3.add(tf.keras.layers.Dropout(0.2))
model3.add(tf.keras.layers.GRU(units = 200))
model3.add(tf.keras.layers.Dropout(0.2))
model3.add(tf.keras.layers.Dense(units = 128))
model3.add(tf.keras.layers.Dense(units = 64))
model3.add(tf.keras.layers.Dense(units = 1))

model3.compile(loss = tf.keras.losses.Huber(), optimizer= 'adam',metrics=['mae'])
model3.summary()
```

Figura 7.19: Tercer modelo de GRU

Este modelo esta formado por las siguientes capas (Solo se explican las que se diferencian del anterior y los cambios en las ya existentes):

1. Dropout: Con un rate de 0.2. Pone aleatoriamente las unidades de entrada a 0 con un rate en cada paso durante el tiempo de entrenamiento, lo que ayuda a evitar el sobreajuste.

2. GRU: La segunda de las GRU con una profundidad de 200.

Este modelo se comparó con el modelo 2 para determinar cual usar. La comparativa de la métrica MAE puede verse en la figura 7.20 y la métrica loss en la figura 7.21. Como puede verse en ambas figuras, las métricas del tercer modelo son considereablemente mejores, es por ello que se concluye que este tercer modelo es el adecuado para realizar las predicciones temporales con ambas series.

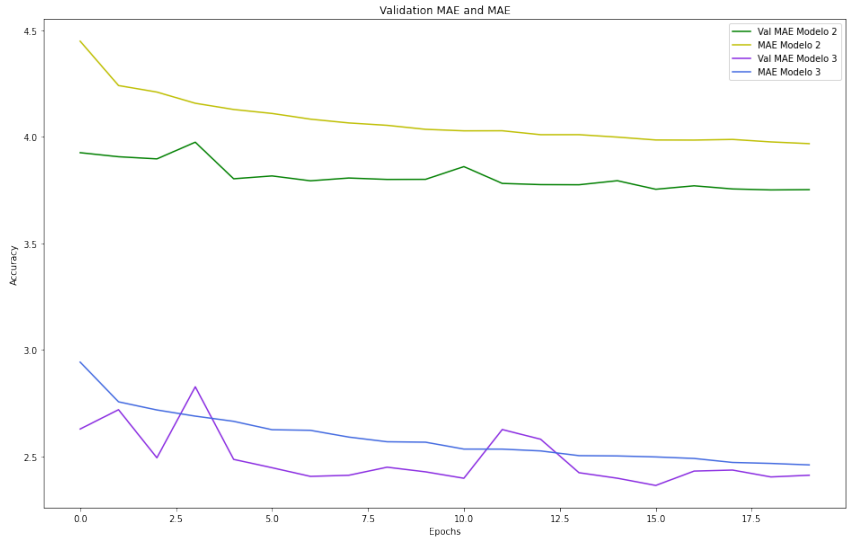

Figura 7.20: Comparativa del MAE entre el modelo 2 y 3 de GRU

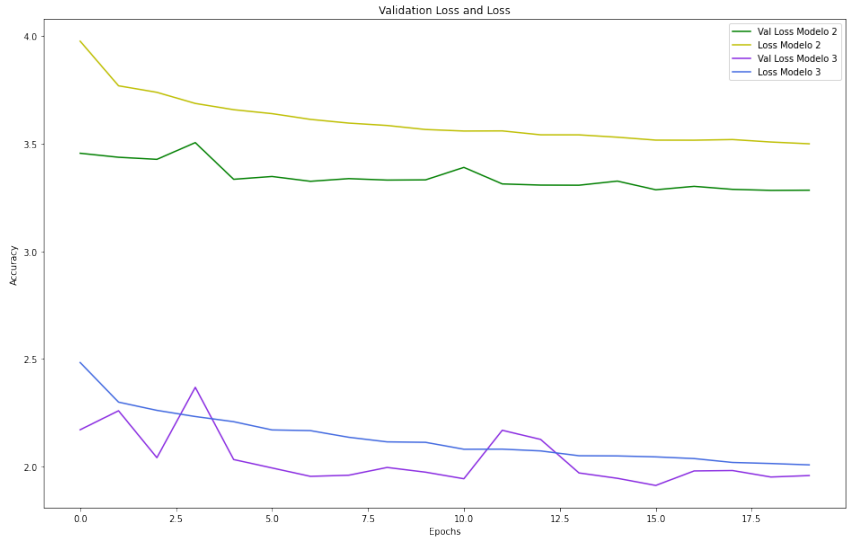

Figura 7.21: Comparativa del loss entre el modelo 2 y 3 de GRU

Capítulo 8

Resultados

Este capítulo recoge los resultados fruto de entrenar y probar los modelos definitivos para cada una de las calles. Tras exponerlos se determinará cual de los modelos planteados es más preciso en base a las métricas establecidas.

8.1. Resultados tras emplear el modelo RNN en la calle Fernando el Católico

El MAE resultante al entrenar para la serie temporal de Fernando el Católico puede verse en la figura 8.1. El loss resultante para Fernando el Católico puede verse en la figura 8.2.

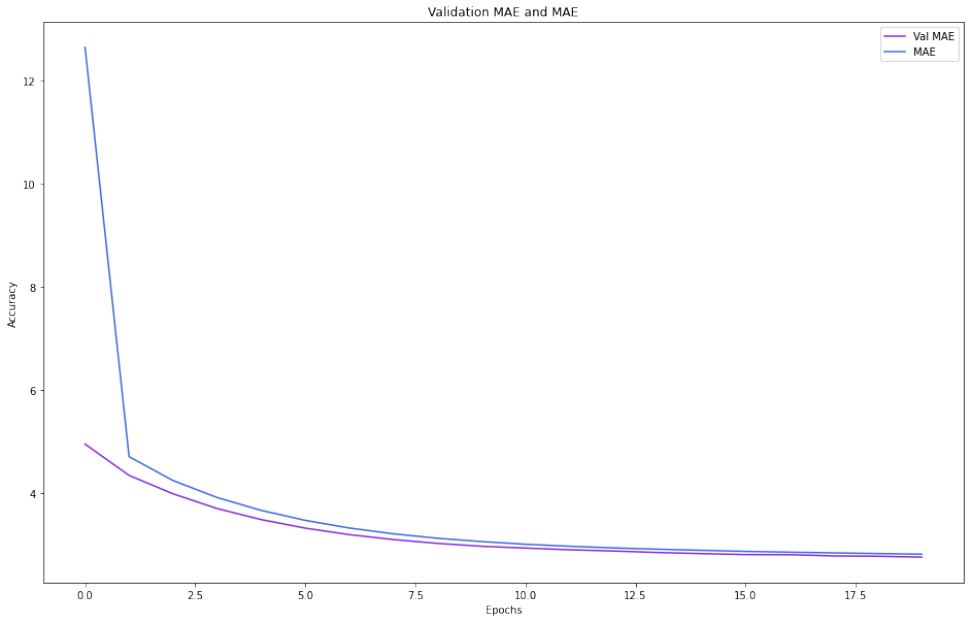

Figura 8.1: MAE resultante tras entrenar la RNN para Fernando el Católico

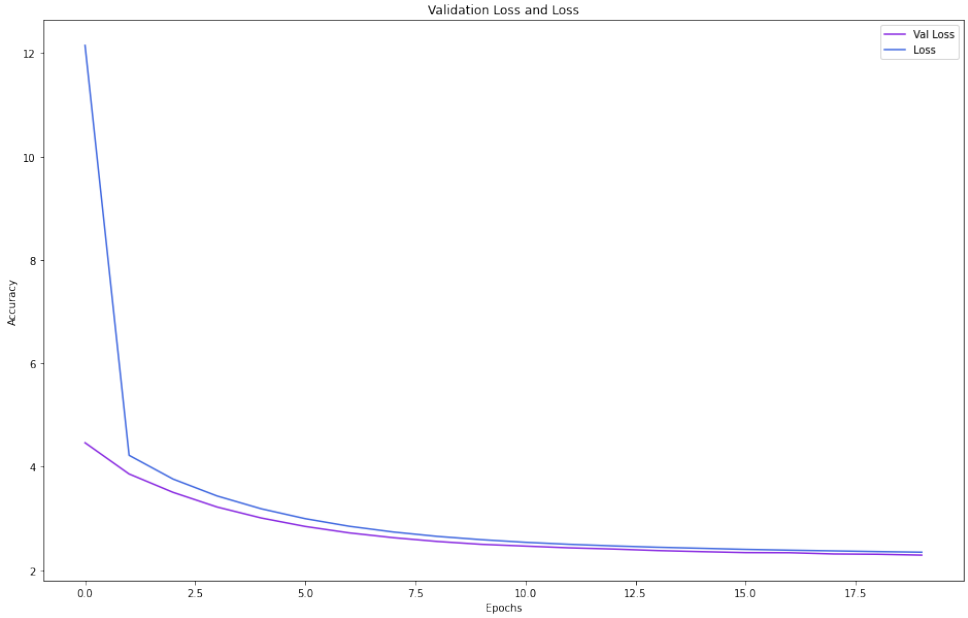

Figura 8.2: loss resultante tras entrenar la RNN para Fernando el Católico

Una vez entrenado el modelo, se prueba para el 20 % de los datos reservados de Fernando el Católico como test, pudiendo ver los resultados generales en la figura 8.3. Para que se vea de forma más precisa, se pueden observar 100 mediciones y sus respectivas predicciones en la figura 8.4. Como se puede ver en ambas imágenes, el modelo se ajusta con un error aceptable, sin embargo, no es preciso a la hora de predecir cambios bruscos en el flujo de tráfico produciendo incluso valores negativos.

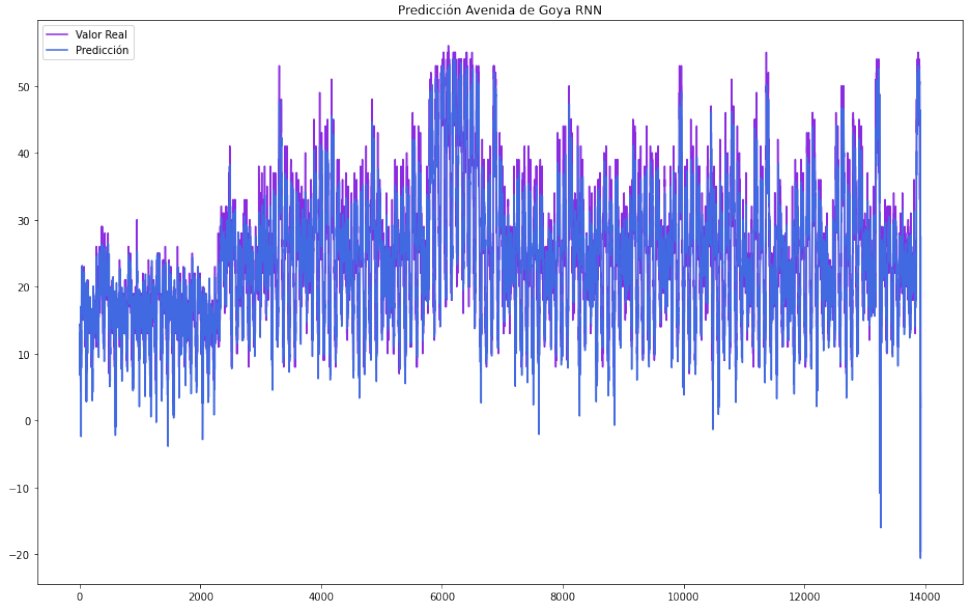

Figura 8.3: Resultado general tras entrenar la RNN para Fernando el Católico

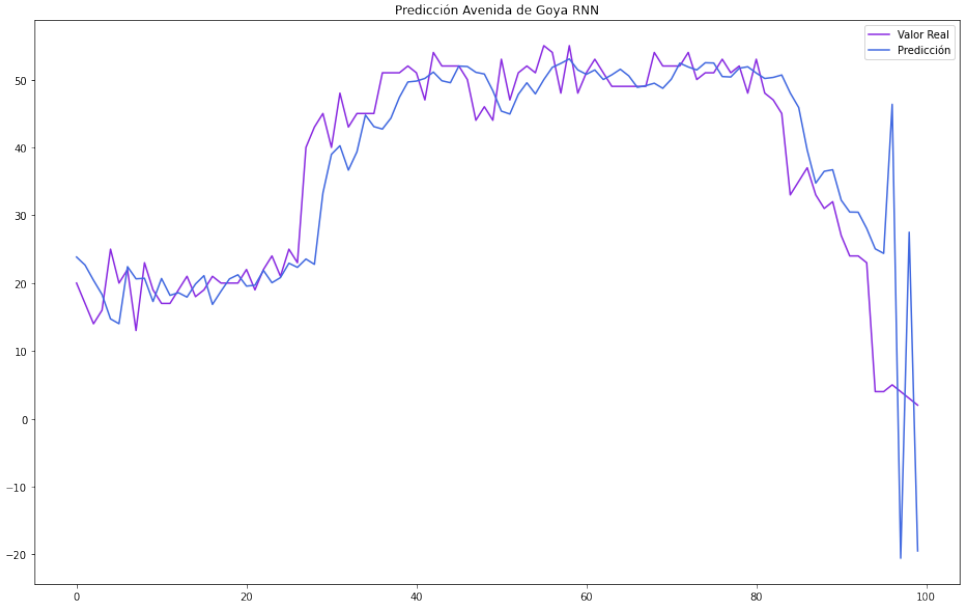

Figura 8.4: Resultado ampliado tras entrenar la RNN para Fernando el Católico

8.2. Resultados tras emplear el modelo RNN en la calle Avenida de Goya

El MAE resultante al entrenar para la serie temporal de Avenida de Goya puede verse en la figura 8.5. El loss resultante para Avenida de Goya puede verse en la figura 8.6.

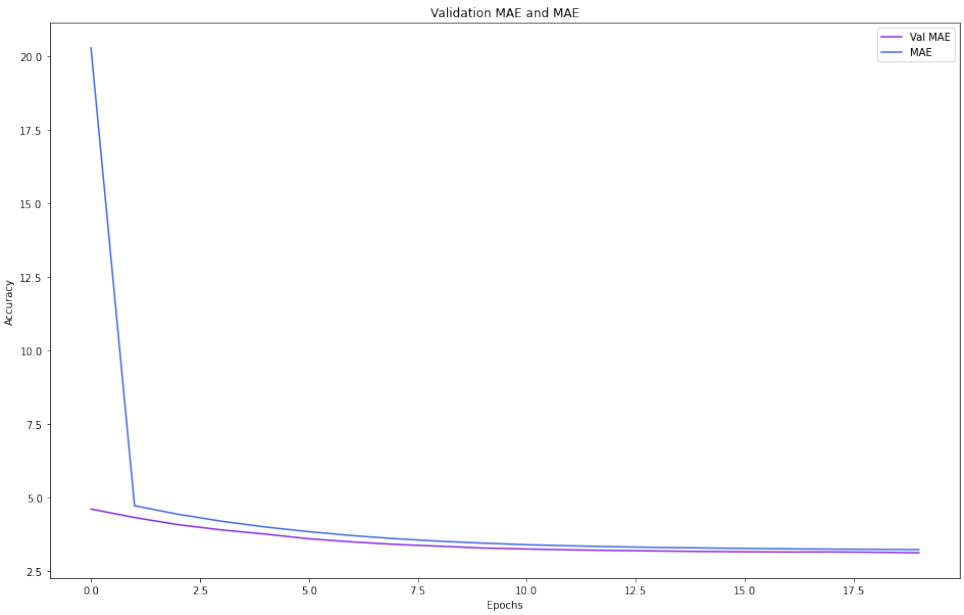

Figura 8.5: MAE resultante tras entrenar la RNN para Avenida de Goya

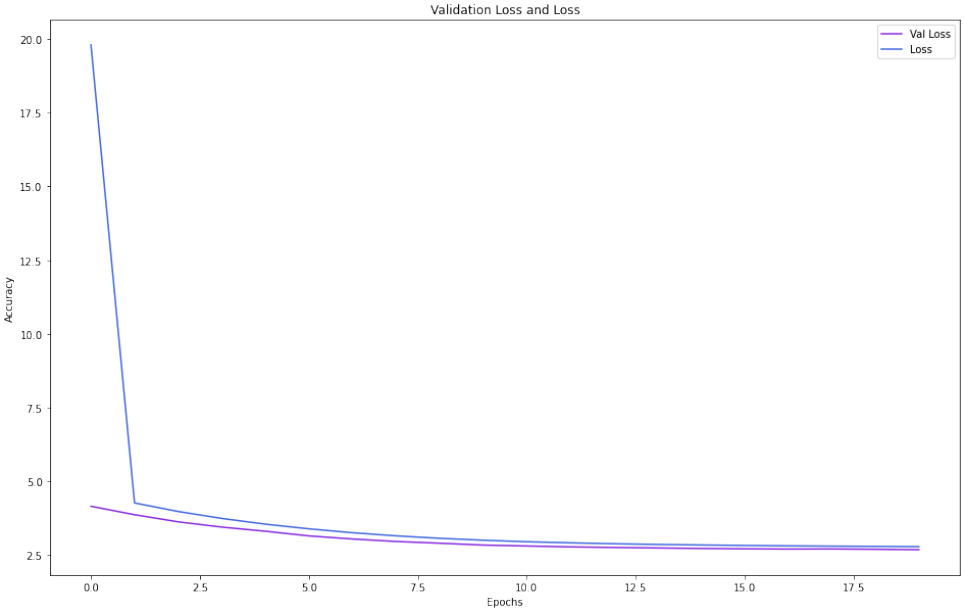

Figura 8.6: loss resultante tras entrenar la RNN para Avenida de Goya

Una vez entrenado el modelo, se prueba para el 20 % de los datos reservados de Avenida de Goya como test, pudiendo ver los resultados generales en la figura 8.7. Para que se vea de forma más precisa, se pueden observar 100 mediciones y sus respectivas predicciones en la figura 8.8. Como se puede ver en ambas imágenes, el modelo se ajusta con un error aceptable, sin embargo, no es preciso a la hora de predecir cambios bruscos en el flujo de tráfico produciendo incluso valores negativos.

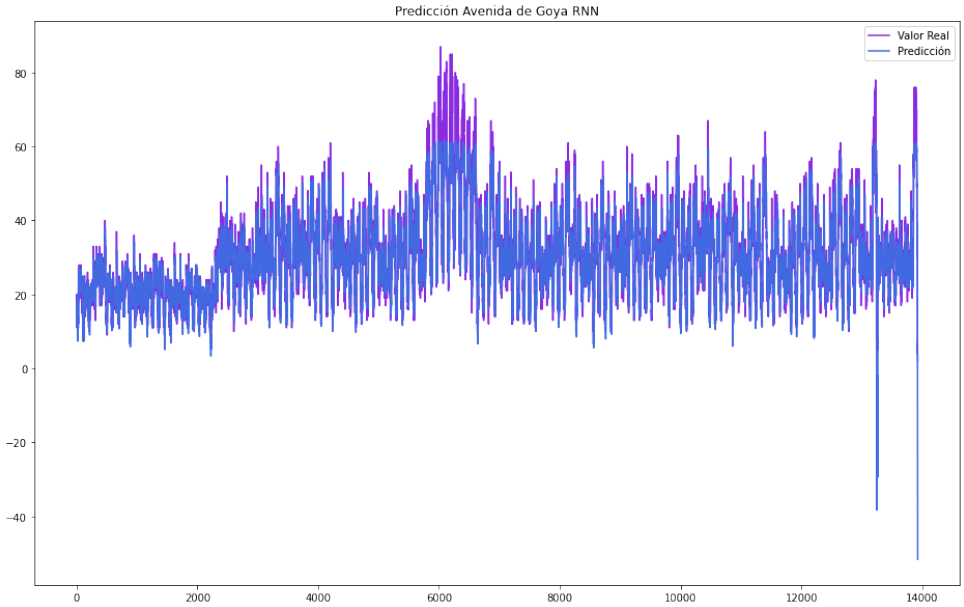

Figura 8.7: Resultado general tras entrenar la RNN para Avenida de Goya
Avenida de Goya

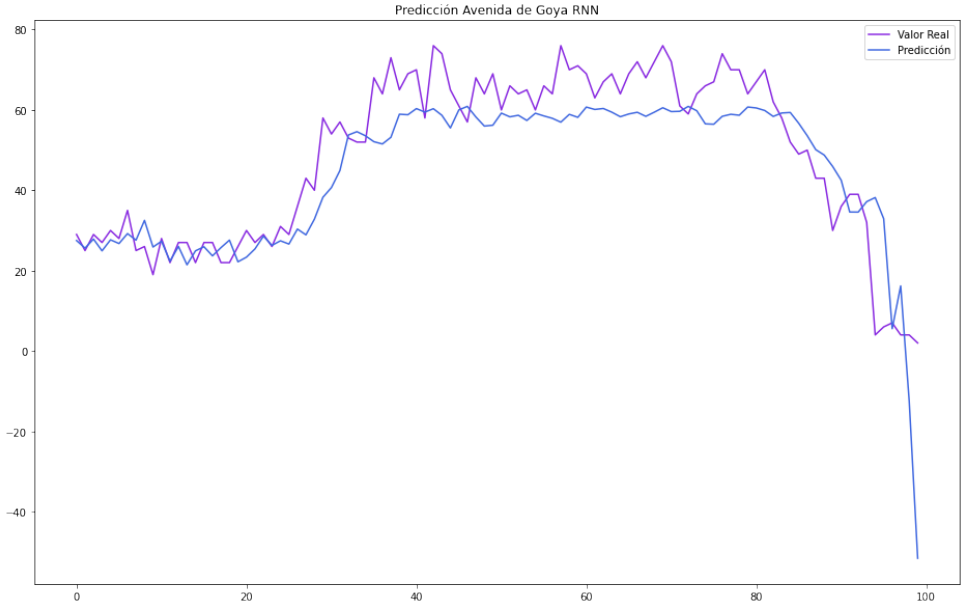

Figura 8.8: Resultado ampliado tras entrenar la RNN para Avenida de Goya

8.3. Resultados tras emplear el modelo LSTM en la calle Fernando el Católico

El MAE resultante al entrenar para la serie temporal de Fernando el Católico puede verse en la figura 8.9. El loss resultante para Fernando el Católico puede verse en la figura 8.10.

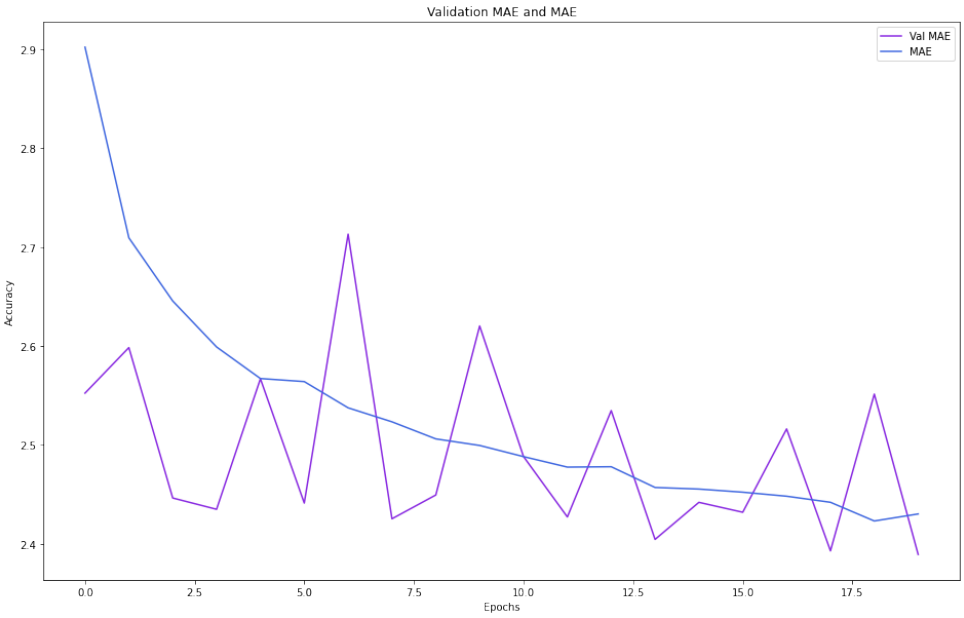

Figura 8.9: MAE resultante tras entrenar la LSTM para Fernando el Católico

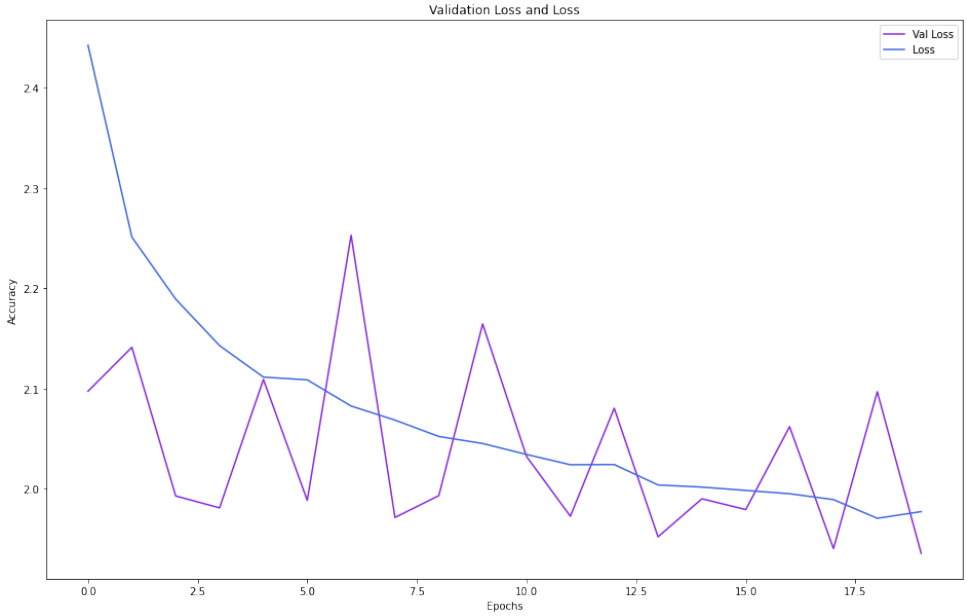

Figura 8.10: loss resultante tras entrenar la LSTM para Fernando el Católico

Una vez entrenado el modelo, se prueba para el 20 % de los datos reservados de Fernando el Católico como test, pudiendo ver los resultados generales en la figura 8.11. Para que se vea de forma más precisa, se pueden observar 100 mediciones y sus respectivas predicciones en la figura 8.12.

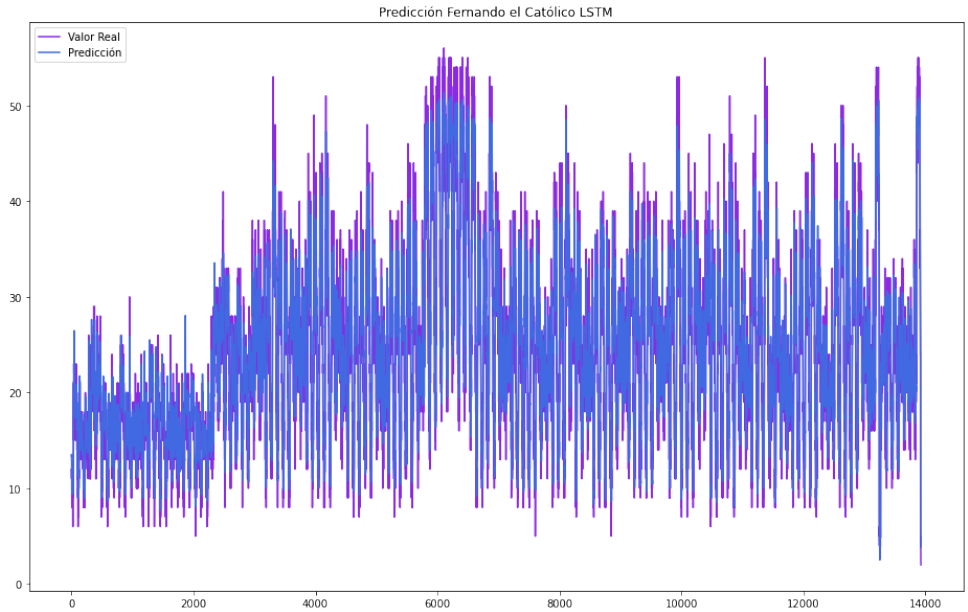

Figura 8.11: Resultado general tras entrenar la LSTM para Fernando el Católico

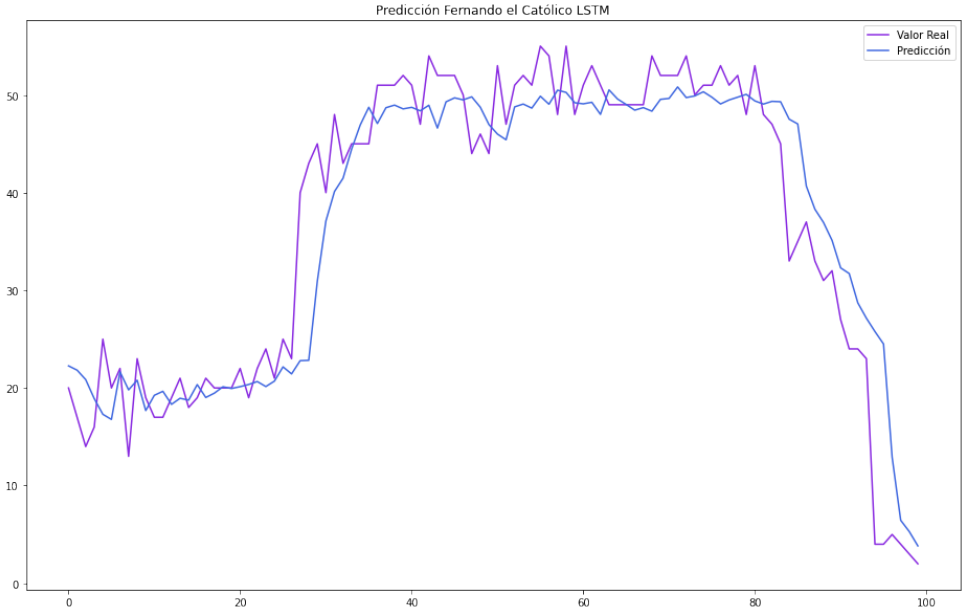

Figura 8.12: Resultado ampliado tras entrenar la LSTM para Fernando el Católico

8.4. Resultados tras emplear el modelo LSTM en la calle Avenida de Goya

El MAE resultante al entrenar para la serie temporal de Avenida de Goya puede verse en la figura 8.13. El loss resultante para Avenida de Goya puede verse en la figura 8.14.

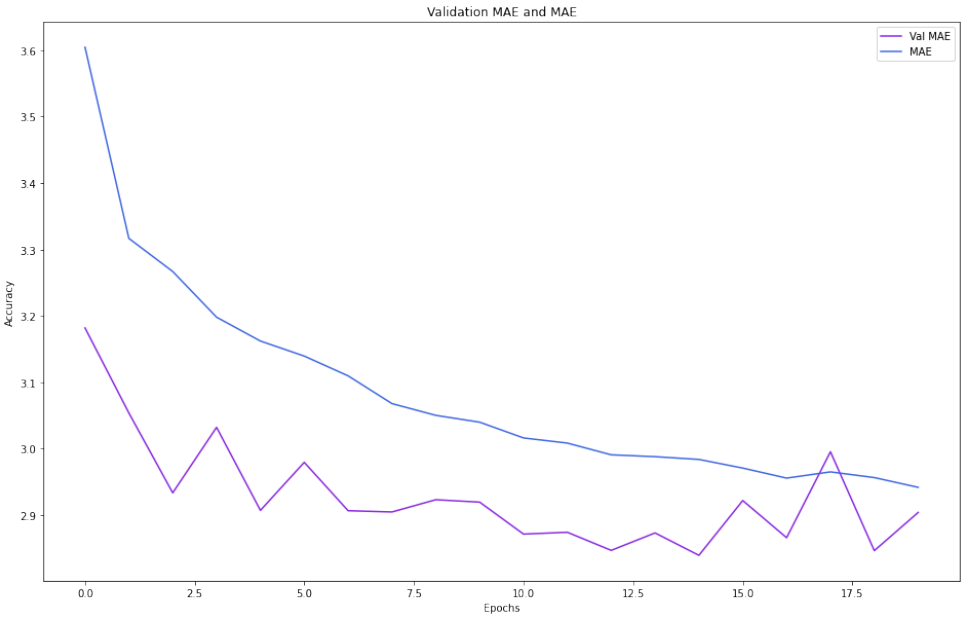

Figura 8.13: MAE resultante tras entrenar la LSTM para Avenida de Goya

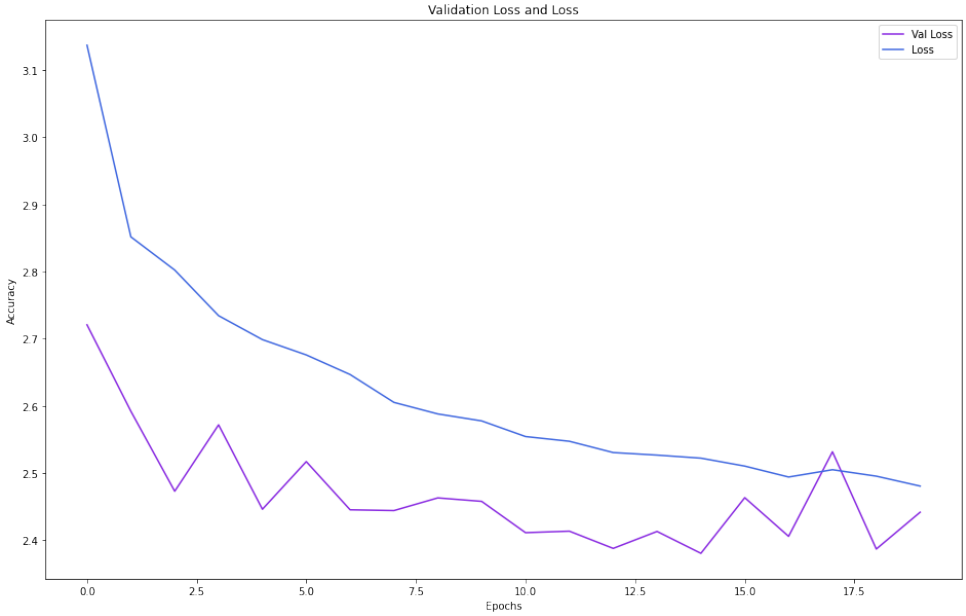

Figura 8.14: loss resultante tras entrenar la LSTM para Avenida de Goya

Una vez entrenado el modelo, se prueba para el 20 % de los datos reservados de Avenida de Goya como test, pudiendo ver los resultados generales en la figura 8.15. Para que se vea de forma más precisa, se pueden observar 100 mediciones y sus respectivas predicciones en la figura 8.16.

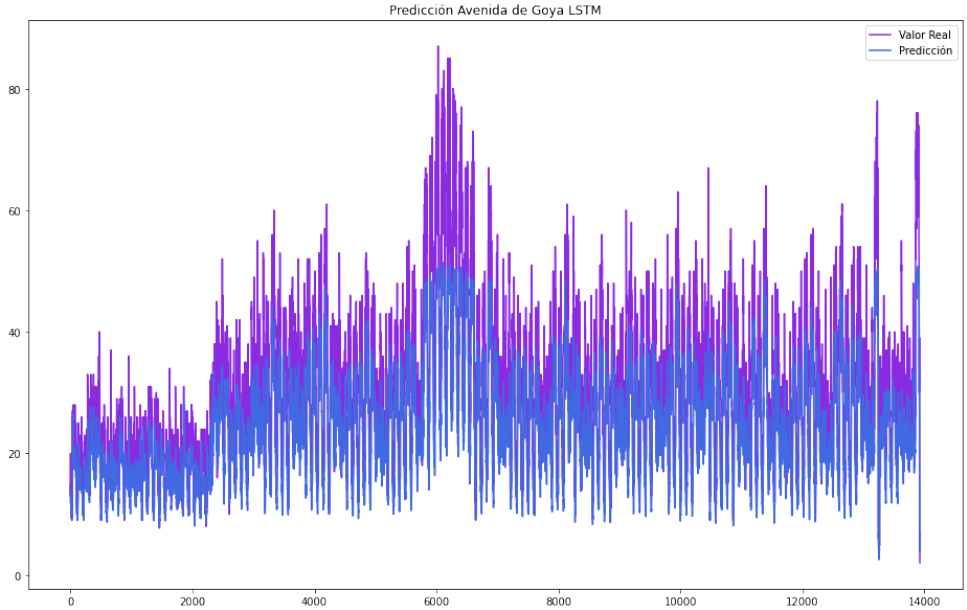

Figura 8.15: Resultado general tras entrenar la LSTM para Avenida de Goya

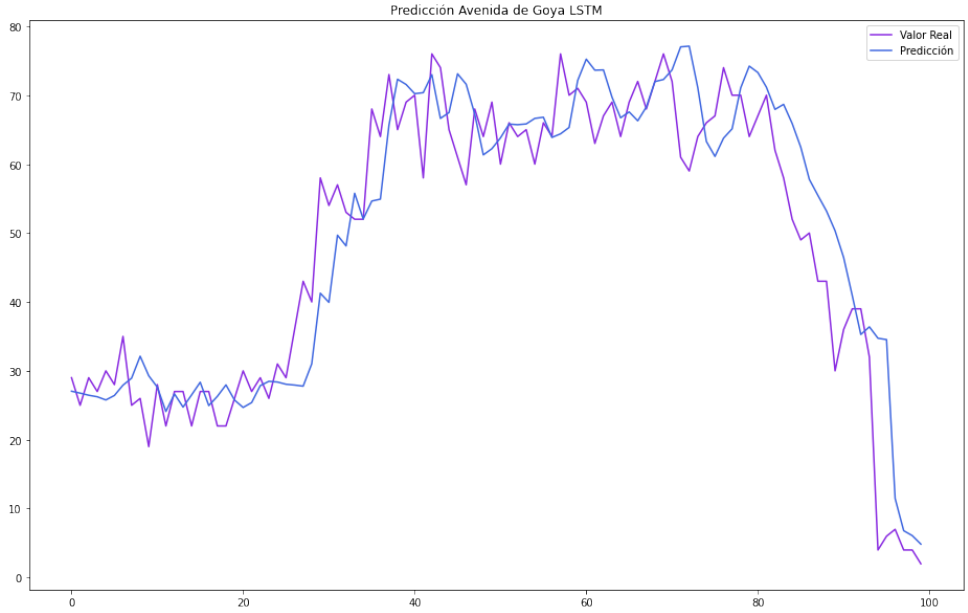

Figura 8.16: Resultado ampliado tras entrenar la LSTM para Avenida de Goya

8.5. Resultados tras emplear el modelo GRU en la calle Fernando el Católico

El MAE resultante al entrenar para la serie temporal de Fernando el Católico puede verse en la figura 8.17. El loss resultante para Fernando el Católico puede verse en la figura 8.18.

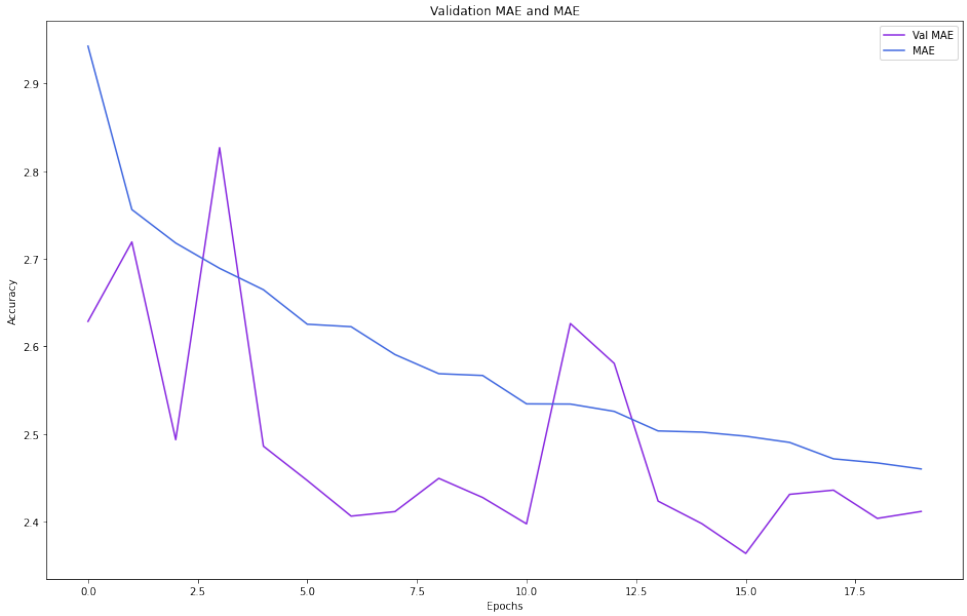

Figura 8.17: MAE resultante tras entrenar la GRU para Fernando el Católico

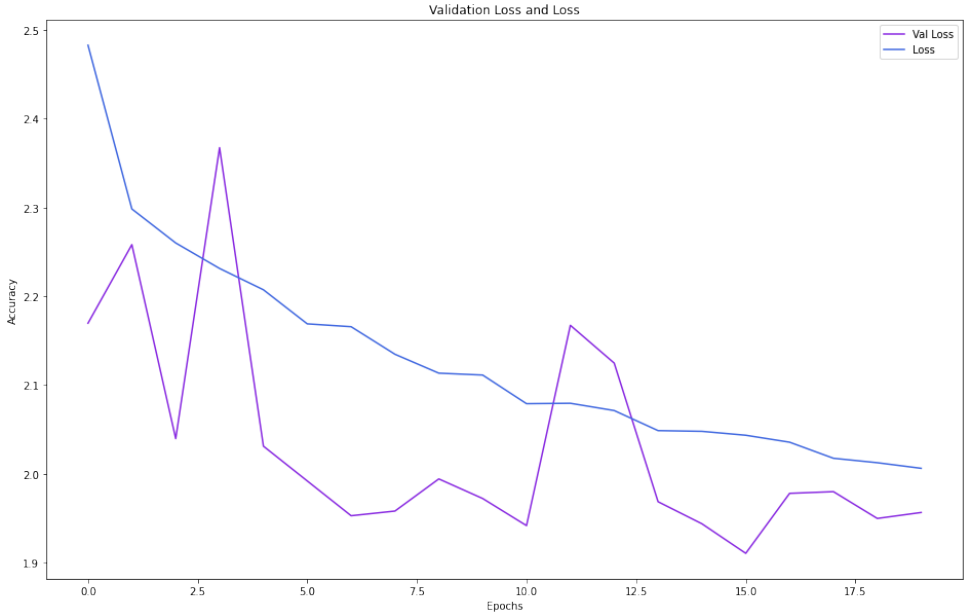

Figura 8.18: loss resultante tras entrenar la GRU para Fernando el Católico

Una vez entrenado el modelo, se prueba para el 20 % de los datos reservados de Fernando el Católico como test, pudiendo ver los resultados generales en la figura 8.19. Para que se vea de forma más precisa, se pueden observar 100 mediciones y sus respectivas predicciones en la figura 8.20.

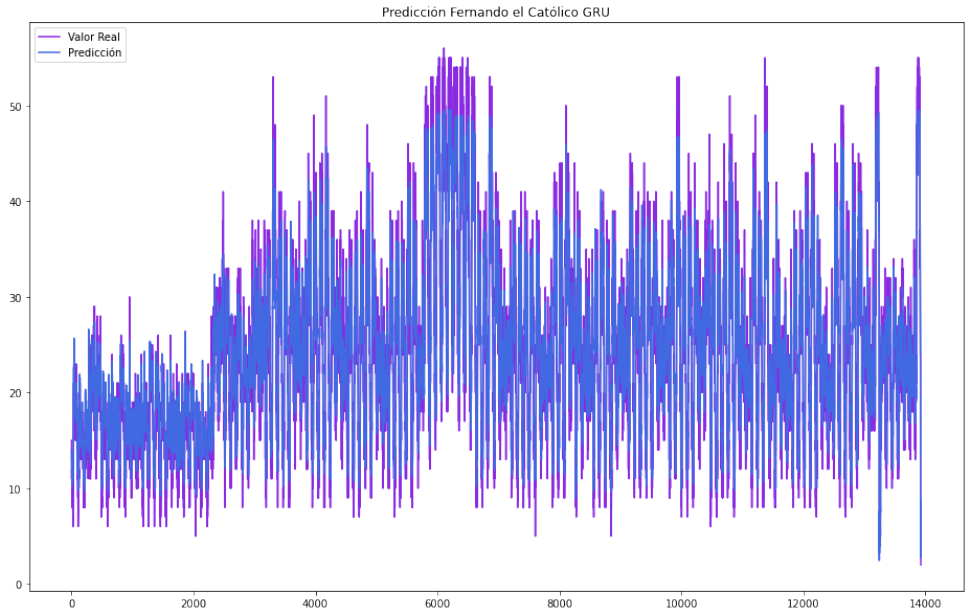

Figura 8.19: Resultado general tras entrenar la GRU para Fernando el Católico

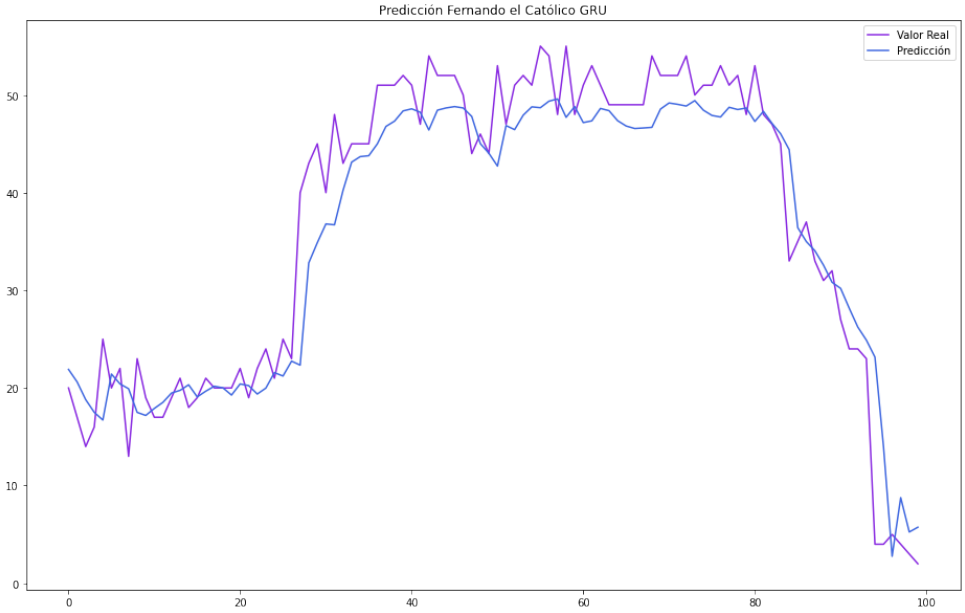

Figura 8.20: Resultado ampliado tras entrenar la GRU para Fernando el Católico

8.6. Resultados tras emplear el modelo GRU en la calle Avenida de Goya

El MAE resultante al entrenar para la serie temporal de Avenida de Goya puede verse en la figura 8.21. El loss resultante para Avenida de Goya puede verse en la figura 8.22.

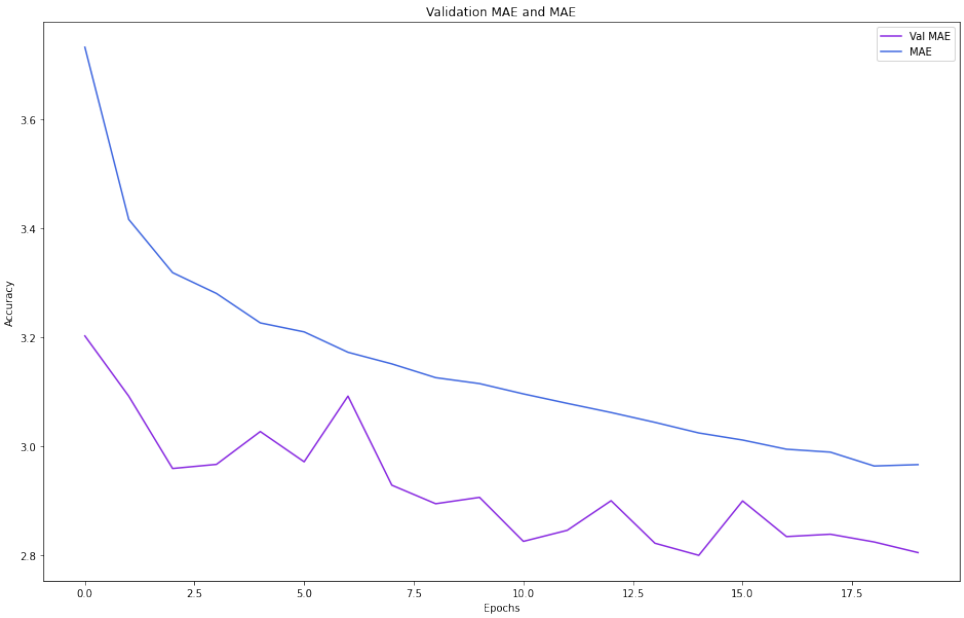

Figura 8.21: MAE resultante tras entrenar la GRU para Avenida de Goya

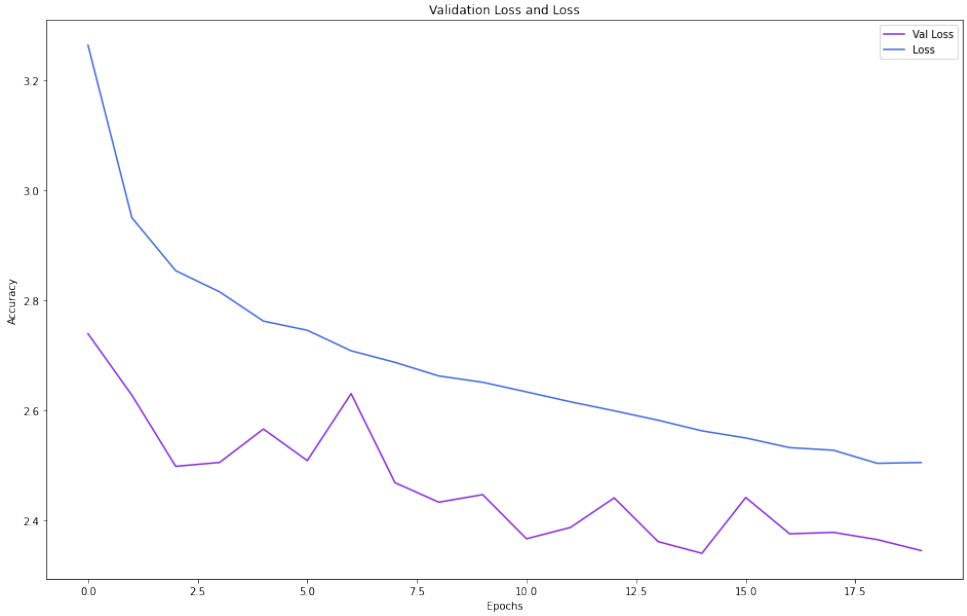

Figura 8.22: loss resultante tras entrenar la GRU para Avenida de Goya

Una vez entrenado el modelo, se prueba para el 20 % de los datos reservados de Avenida de Goya como test, pudiendo ver los resultados generales en la figura 8.23. Para que se vea de forma más precisa, se pueden observar 100 mediciones y sus respectivas predicciones en la figura 8.24.

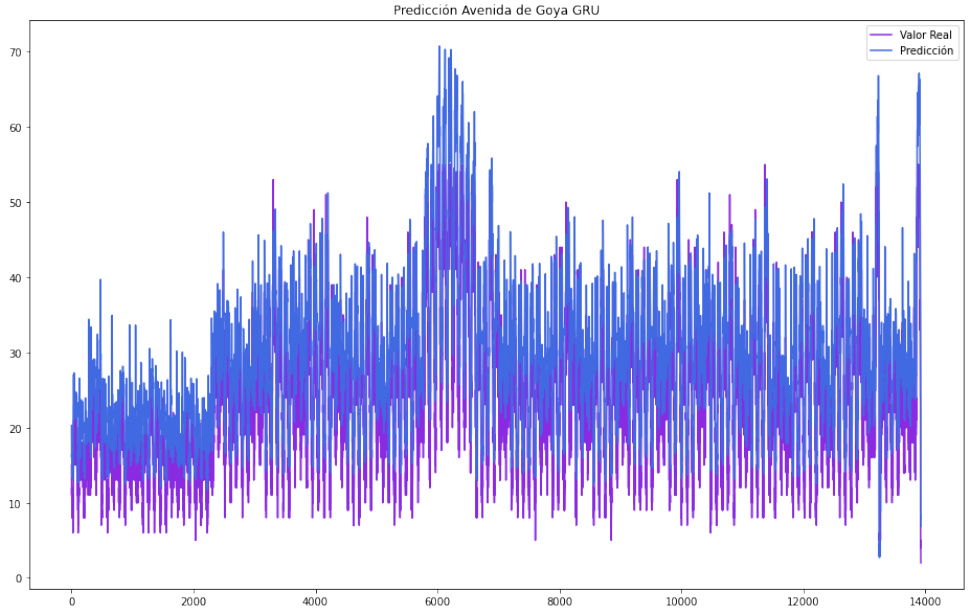

Figura 8.23: Resultado general tras entrenar la GRU para Avenida de Goya

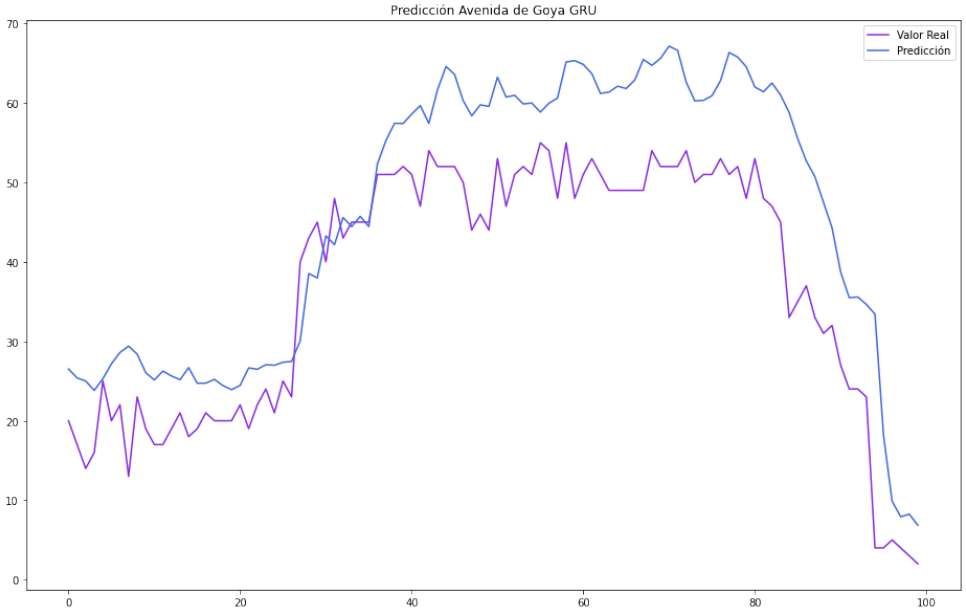

Figura 8.24: Resultado ampliado tras entrenar la GRU para Avenida de Goya

Capítulo 9

Discusión

Este apartado recoge una discusión acerca de los resultados obtenidos de los diferentes modelos. La primera de las dos métricas, es decir, el MAE entre los diferentes modelos para cada una de las dos calles tal y como se ve en las figuras 9.1 y 9.2,

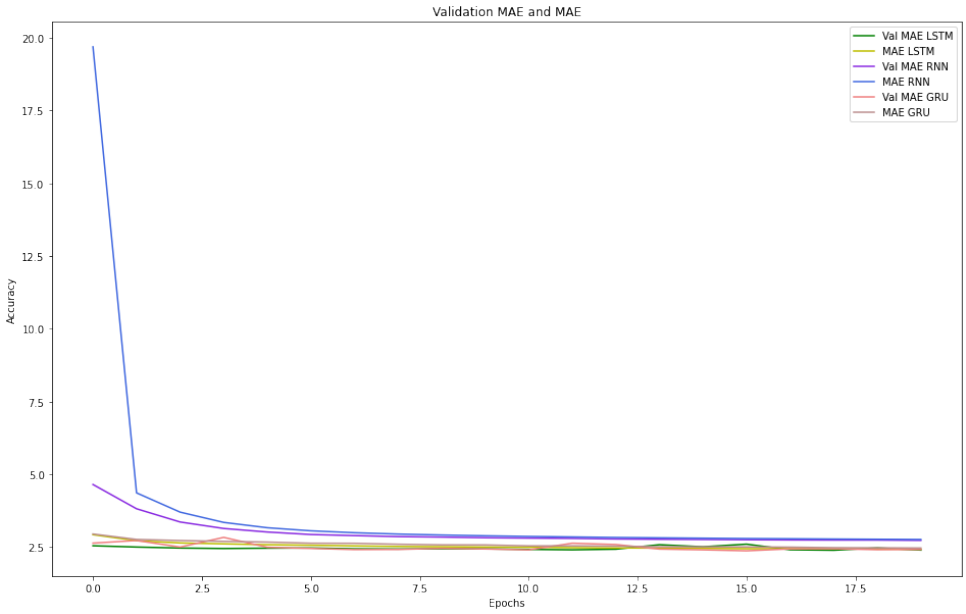

Figura 9.1: Comparativa del MAE de los modelos para Fernando el Católico

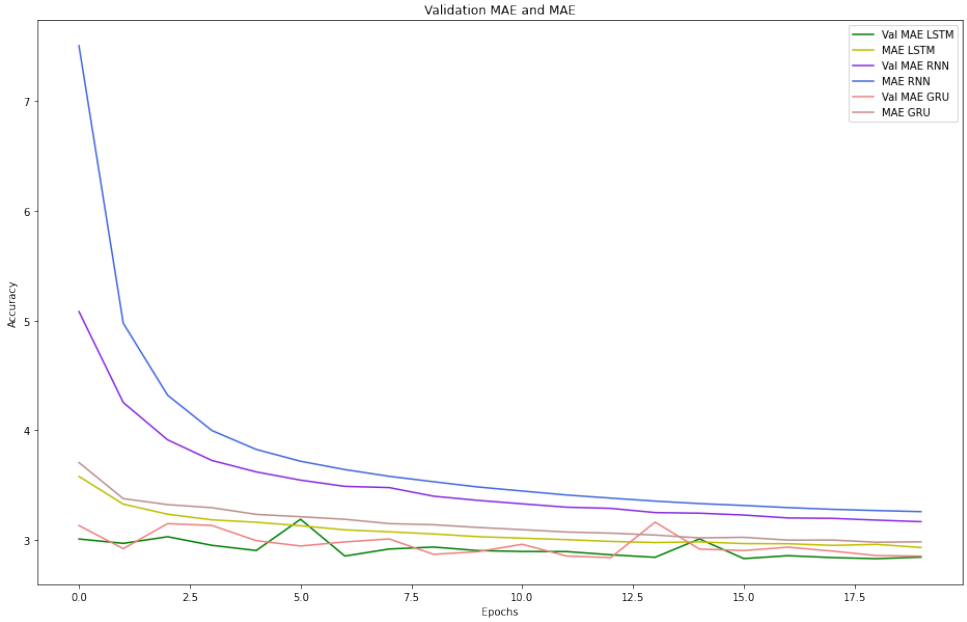

Figura 9.2: Comparativa del MAE de los modelos para Avenida de Goya

podemos ver como el modelo LSTM y el GRU son superiores al RNN en cuanto a la precisión de sus predicciones. Si observamos la segunda de las dos métricas, es decir, el loss entre los diferentes modelos para cada una de las dos calles tal y como se ve en las figuras 9.3 y 9.4,

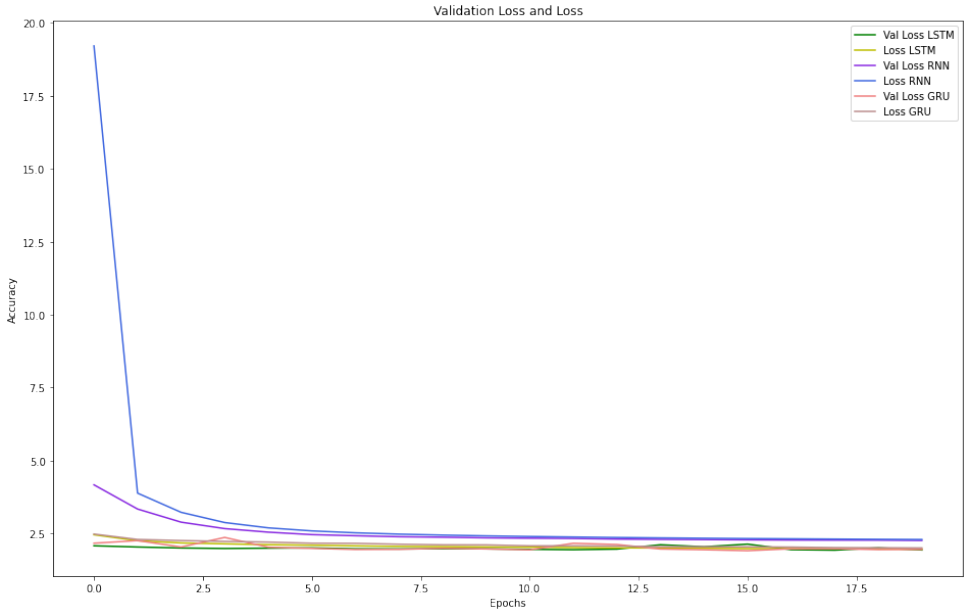

Figura 9.3: Comparativa del loss de los modelos para Fernando el Católico

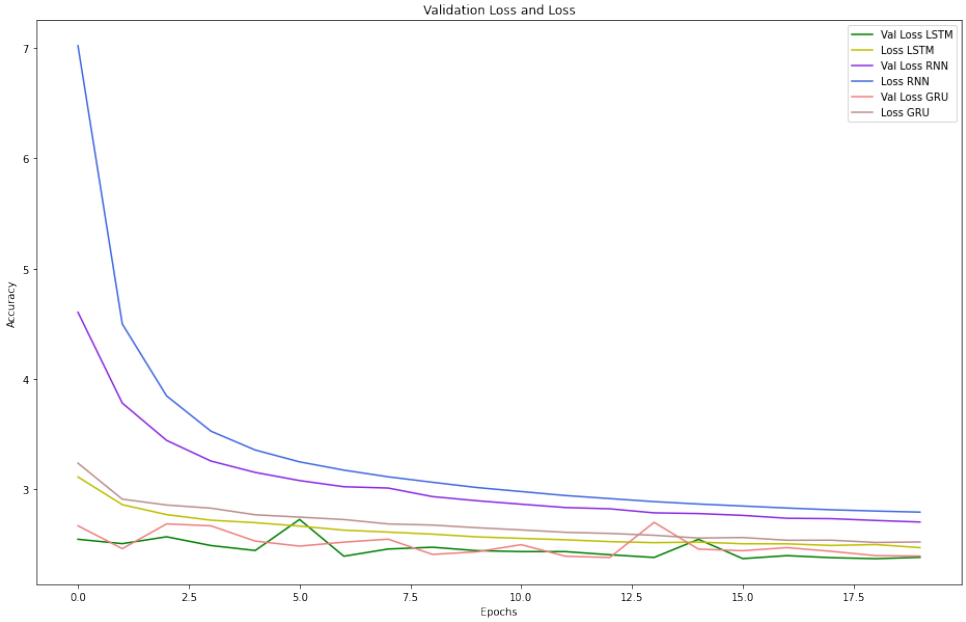

Figura 9.4: Comparativa del loss de los modelos para Avenida de Goya

podemos ver como el modelo LSTM y el GRU son superiores al RNN en cuanto a la precisión de sus predicciones. Teniendo en cuenta estos resultados presentados, podemos concluir que tanto la LSTM como la GRU ofrecen resultados muy similares y ampliamente superiores a la RNN. Si nos fijamos en la gráfica del MAE vemos que tanto la LSTM como la GRU cometen un error medio de menos de dos coches en cada predicción diaria, un error que podemos considerar válido a la hora de tomar decisiones en función del flujo del tráfico pues dos coches por encima o por debajo del flujo real no suponen ningún problema.

Capítulo 10

Conclusiones y trabajo futuro

El siguiente capítulo contiene las conclusiones que han resultado del proceso de desarrollo del presente Trabajo de Fin de Máster. Al final de este capítulo se presentarán los posibles trabajos futuros con una serie de observaciones a tener en cuenta si se quisiera proseguir con la investigación.

10.1. Conclusiones

En los últimos años tanto empresas privadas como públicas han realizado numerosos estudios e investigaciones sobre las predicciones en el flujo de tráfico con el objetvo de poder tomar decisiones de forma premeditada y con suficiente antelación.

El primero de los objetivos planteados fue realizar una revisión bibliográfica sobre cómo se ha solventado este problema desde otros puntos de vista, lo cual permitió observar que no había una única forma eficiente de darle solución. Esta falta de uniformidad en las soluciones, llevó a plantear el TFM de forma que se empleasen los algoritmos de predicción temporal más conocidos y utilizados. Otro de los elementos observado en la literatura existene fue que dada la dificultad que suponía dar con un modelo que fuera válido para cualquier vía, las redes empleadas tenían que tener una gran profundidad para así dar con métricas lo suficientemente buenas como para poder emplear el modelo en casos reales.

Dado lo anteriormente expuesto, el trabajo se centró en dar con una serie de modelos lo suficientemente precisos en base a las métricas explicadas anteriormente y lo suficientemente sencillos como para poder ser empleados en varios tipos de vías. Iniciando así el desarrollo de modelos de tipo RNN y LSTM para predicción de series temporales realizando una serie de pruebas que verificaran su correcto funcionamiento.

Respecto al resto de los objetivos todos ellos se han completado con éxito según lo expuesto en los capítulos 7 y 5, es decir, se ha determinado que ninguna de las variables medidas influia en el número de coches que circulaban por la vía, se ha realizado una correcta preparación de los datos para su uso en los modelos, se han determinado cuales eran los mejores modelos para la predicción de series temporales.

Por último se ha determinado en base a las métricas expuestas cual es el mejor de los modelos empleados en el desarrollo del proyecto. Estos modelos son lo suficientemente genéricos y eficientes como para ser empleados en la predicción del flujo del tráfico de cualquier otra vía.

10.2. Líneas de trabajo futuro

Dado el tiempo limitado para el desarrollo del TFM se plantean las siguientes líneas de trabajo futuras de cara a la continuidad del proyecto para así mejorarlo y obtener mejores resultados.

Respecto a los datos, sería interesante monitorizar otras variables del entorno con el objetivo de encontral algunas dependecias del flujo de tráfico. Además se podrían segmentar los conjuntos de datos para realizar técnicas de entrenamiento distribuido y así aumentar la eficiencia del proceso de ajuste de los pesos.

Por último, sería interasante hacer una investigación en la rama del Deep Learning en su aplicación a la predicción de series temporales. Pues el campo de los algoritmos inteligentes en el ámbito del Deep Learning cuentan con técnicas para conseguir una mayor precisión en un menor tiempo de entrenamiento.

Bibliografía

[1] Juan S Angarita-Zapata, Antonio D Masegosa e Isaac Triguero. «Evaluating automated machine learning on supervised regression traffic forecasting problems». En: *Computational Intelligence in Emerging Technologies for Engineering Applications*. Springer, 2020, págs. 187-204.

[2] Juan S Angarita-Zapata, Isaac Triguero y Antonio D Masegosa. «A preliminary study on automatic algorithm selection for short-term traffic forecasting». En: *International Symposium on Intelligent and Distributed Computing*. Springer. 2018, págs. 204-214.

[3] Moshe Ben-Akiva y col. «DynaMIT: a simulation-based system for traffic prediction». En: *DACCORD short term forecasting workshop*. Citeseer. 1998, págs. 1-12.

[4] Moshe E Ben-Akiva y col. «A dynamic traffic assignment model for highly congested urban networks». En: *Transportation research part C: emerging technologies* 24 (2012), págs. 62-82.

[5] Alireza Ermagun y David Levinson. «Spatiotemporal traffic forecasting: review and proposed directions». En: *Transport Reviews* 38.6 (2018), págs. 786-814.

[6] Xiaochen Fan y col. «Buildsensys: reusing building sensing data for traffic prediction with cross-domain learning». En: *IEEE Transactions on Mobile Computing* (2020).

[7] Andrew Hamilton y col. «The evolution of urban traffic control: changing policy and technology». En: *Transportation planning and technology* 36.1 (2013), págs. 24-43.

[8] Wenhao Huang y col. «Deep architecture for traffic flow prediction: deep belief networks with multitask learning». En: *IEEE Transactions on Intelligent Transportation Systems* 15.5 (2014), págs. 2191-2201.

[9] Maryland Transportation Initiative. «DYNASMART-X Evaluation for Real-Time TMC Application: CHART Test Bed». En: (2005).

[10] Yuhan Jia, Jianping Wu y Yiman Du. «Traffic speed prediction using deep learning method». En: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2016, págs. 1217-1222.

[11] Danqing Kang, Yisheng Lv y Yuan-yuan Chen. «Short-term traffic flow prediction with LSTM recurrent neural network». En: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, págs. 1-6.

[12] Ibai Lana y col. «Road traffic forecasting: Recent advances and new challenges». En: *IEEE Intelligent Transportation Systems Magazine* 10.2 (2018), págs. 93-109.

[13] Zhiheng Li y col. «Building sparse models for traffic flow prediction: An empirical comparison between statistical heuristics and geometric heuristics for Bayesian network approaches». En: *Transportmetrica B: Transport Dynamics* (2017).

[14] Yisheng Lv y col. «Traffic flow prediction with big data: a deep learning approach». En: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2014), págs. 865-873.

[15] Yixuan Ma, Zhenji Zhang y Alexander Ihler. «Multi-lane short-term traffic forecasting with convolutional LSTM network». En: *IEEE Access* 8 (2020), págs. 34629-34643.

[16] Jonathan Mackenzie, John F Roddick y Rocco Zito. «An evaluation of HTM and LSTM for short-term arterial traffic flow prediction». En: *IEEE Transactions on Intelligent Transportation Systems* 20.5 (2018), págs. 1847-1857.

[17] VA Melnikov y ND Kharchenko. «Using a Combination of Recurrent and Convolutional Neural Networks to Forecast the Direction of Financial Instrument Price Movement». En: *8th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2020)*. Atlantis Press. 2020, págs. 209-211.

[18] Alex Sherstinsky. «Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network». En: *Physica D: Nonlinear Phenomena* 404 (2020), pág. 132306.

[19] Ahmed Tealab. «Time series forecasting using artificial neural networks methodologies: A systematic review». En: *Future Computing and Informatics Journal* 3.2 (2018), págs. 334-340.

[20] Ana Isabel Torre-Bastida y col. «Big Data for transportation and mobility: recent advances, trends and challenges». En: *IET Intelligent Transport Systems* 12.8 (2018), págs. 742-755.

[21] Yibing Wang y Markos Papageorgiou. «Real-time freeway traffic state estimation based on extended Kalman filter: a general approach». En: *Transportation Research Part B: Methodological* 39.2 (2005), págs. 141-167.

[22] Yibing Wang, Markos Papageorgiou y Albert Messmer. «RENAISSANCE–A unified macroscopic model-based approach to real-time freeway network traffic surveillance». En: *Transportation Research Part C: Emerging Technologies* 14.3 (2006), págs. 190-212.

[23] Wangyang Wei, Honghai Wu y Huadong Ma. «An autoencoder and LSTM-based traffic flow prediction method». En: *Sensors* 19.13 (2019), pág. 2946.

[24] Yi Yin y Pengjian Shang. «Forecasting traffic time series with multivariate predicting method». En: *Applied Mathematics and Computation* 291 (2016), págs. 266-278. ISSN: 0096-3003.

[25] Bing Yu, Haoteng Yin y Zhanxing Zhu. «Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting». En: *arXiv preprint arXiv:1709.04875* (2017).

Capítulo 11

Anexo 1: Código empleado

Este capítulo final corresponde el Anexo 1 dedicado al código empleado para el TFM. Este código en formato de Jupyter Notebook y PDF junto con los csv empelados puede verse en el repositorio de GitHub correspondiente:

$$https://github.com/MarcosSevert/TFM\_$$