UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Predicción de ancho de banda disponible en redes inalámbricas altamente dinámicas

Autor:

D. Juan Manuel Espinosa Lerma

Tutor:

Dr. D. Eduardo Gómez Sánchez

Dr. D. Miguel Luis Bote Lorenzo

Valladolid, 21 de junio de 2024

TÍTULO:	**Predicción de ancho de banda disponible en redes inalámbricas altamente dinámicas.**
AUTOR:	**D. Juan Manuel Espinosa Lerma**
TUTOR:	**Dr. D. Eduardo Gómez Sánchez** **Dr. D. Miguel L. Bote Lorenzo**
DEPARTAMENTO:	**Teoría de la Señal y las Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. D. Juan I. Asensio Pérez**
VOCAL:	**Dr. D. Juan Pablo Casaseca de la Higuera**
SECRETARIO:	**Dr. D. Ignacio de Miguel Jiménez**
FECHA:	**28 de junio de 2024**
CALIFICACIÓN:	

RESUMEN

Las redes móviles son redes inalámbricas que proporcionan acceso a servicios como el acceso a Internet, entre otros. Los enlaces de estas redes son poco fiables, dado fundamentalmente por su carácter inalámbrico, y la capacidad de la que se va a disponer depende de muchos factores como la potencia de señal recibida o la calidad. Diversas aplicaciones de usuario deben ajustar ciertos parámetros en función de la capacidad disponible en cada momento, con el fin de proporcionar la mejor experiencia final al usuario. El uso de técnicas de aprendizaje automático puede resultar útil para predecir la capacidad que se dispondrá en los próximos instantes y, consecuentemente, las aplicaciones podrán ajustarse a las circunstancias de la red. A este respecto, trabajos previos han estudiado cómo el aprendizaje automático puede predecir la capacidad disponible en una red. Sin embargo, los modelos creados implican una carga computacional no asumible en muchos entornos reales. El objetivo de este TFG es analizar varios tipos de predictores, con especial énfasis en los basados en redes neuronales LSTM, buscando un compromiso entre carga computacional y una buena predicción. Para ello, se explorarán diversos conjuntos de datos, así como diversas

variantes de LSTM. Se programará un predictor LSTM base con el objetivo de entenderlo y se intentará reproducir los resultados de la literatura. Por último, se discutirá el beneficio de la predicción con LSTM frente al elevado coste computacional de los modelos.

PALABRAS CLAVE

Redes móviles, 4G, LTE, redes neuronales, LSTM, aprendizaje automático, coste computacional.

ABSTRACT

Mobile networks are wireless networks that provide access to services such as Internet access, among others. The links of these networks are unreliable, mainly due to their wireless nature, and the capacity that will be available depends on many factors such as the received signal power or quality. Various user applications must adjust certain parameters based on the capacity available at any given time, in order to provide the best final user experience. The use of machine learning techniques can be useful to predict the capacity that will be available in the next few moments and, consequently, applications can adjust to the circumstances of the network. In this regard, previous work has studied how machine learning can predict the available capacity in a network. However, the models created imply a computational load that is not acceptable in many real environments. The objective of this TFG is to analyze various types of predictors, with special emphasis on those based on LSTM neural networks, seeking a compromise between computational load and good prediction. To do this, various data sets will be explored, as well as various variants of LSTM. A base LSTM predictor will be programmed with the objective of understanding it and an attempt will be made to reproduce the results from the literature. Finally, the benefit of prediction with LSTM compared to the high computational cost of the models will be discussed.

KEYWORDS

Cellular network, mobile network, 4G, LTE, neural networks, LSTM, machine learning, computational cost.

Agradecimientos

Son muchas las personas a las que tengo que agradecer el apoyo y confianza que me han permitido llegar hasta aquí. En primer lugar, me encantaría agradecer a mis tutores, Eduardo y Miguel, por haberme ofrecido la oportunidad de hacer este trabajo con ellos y por estar constantemente encima de mí a lo largo del desarrollo del trabajo. También quiero agradecer a Juan Ignacio Asensio que, pese a no ser tutor, ha estado igualmente implicado en nuestros avances y cuando nos quedábamos atascados con algo.

Gracias a todos mis compañeros de carrera por el apoyo que nos hemos ido proporcionando a lo largo de estos cuatro años. Quiero agradecer también a Pablo García por la gran ayuda que me ha proporcionado, sobretodo en temas de presentación y diseño, y por sus consejos.

Por último, quiero agradecer a mi familia. Sin vuestro apoyo y comprensión no habría sido capaz de llegar hasta aquí.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Las redes de comunicación móviles

Las comunicaciones móviles, fundamentales hoy en día, siguen el principio general de la telefonía: conectar dos usuarios remotos a través de un equipo de red de un operador responsable de la gestión del servicio [1]. A diferencia de las comunicaciones por teléfono fijo, en la red móvil no existe soporte físico, y las transmisiones de radio constituyen el enlace final. El teléfono móvil del usuario se comunica mediante ondas electromagnéticas que se propagan a través del aire con una antena que se comunica con la central del operador, la cual encamina la comunicación hacia la parte correspondiente en la red fija o a través de las antenas. Para que esta comunicación sea efectiva, el usuario móvil ha de estar en el área de alcance de una antena. Esta área tiene un alcance limitado y cubre una pequeña área alrededor, área que se conoce como «celda» [2]. Para cubrir el máximo territorio y garantizar cobertura al mayor número de usuarios posibles, los operadores despliegan miles de celdas, cada una equipada con estaciones base, asegurándose de que no haya huecos entre ellas para nunca perder la comunicación con los usuarios [1].

Lo que ha vuelto a las redes móviles más importantes en el mundo actual no es exclusivamente el acceso en movilidad a un servicio de comunicación de voz (llamadas), sino el acceso a Internet, lo que facilita el trabajo y la educación a distancia en aquellos lugares donde no hay accesible ninguna red cableada, por ejemplo. El tráfico de Internet en redes móviles se ha visto increíblemente aumentado en los últimos años. Esto es, en parte, debido a los grandes avances en las tecnologías de acceso inalámbricas y el desarrollo de aplicaciones multimedia en los ámbitos del *streaming*, videoconferencias, Realidad Virtual (*Virtual Reality* – VR), Realidad Aumentada (*Augmented Reality* – AR), Realidad Mixta (*Mixted Reality* – MR), conducción autónoma, etc [3].

Si bien es cierto que, en la actualidad, están operativas redes inalámbricas de última generación, como las redes 5G, diseñadas para proporcionar una elevada capacidad, alta fiabilidad y muy baja latencia, la calidad de servicio (QoS) que se proporciona a los usuarios es vulnerada por la calidad del canal físico existente entre los dispositivos de usuario y los puntos de acceso de las operadoras [4] [3].

Las redes móviles no otorgan la fiabilidad de las redes cableadas en términos de garantías de calidad ni aseguramiento de capacidad disponible. Además, el espectro del que dispone una operadora es limitado. El espectro es un recurso natural que se distribuye

en diferentes bandas de frecuencia y, por tanto, tiene un número máximo de usuarios que pueden conectarse y tramitar sus peticiones simultáneamente. Con un acceso compartido (por ejemplo, contienda) el número de usuarios es ilimitado. Sin embargo, con un acceso multiplexado, no. En 1G y 2G el acceso era de contienda en el canal de señalización y multiplexado en los canales de voz [2]. La congestión normalmente es debida a la coincidencia simultánea de numerosas peticiones de acceso y transmisión (datos o voz) por parte de un número de usuarios que supera la capacidad del canal. La capacidad [5] («*troughput*», o tasa de transferencia efectiva) es la cantidad de datos (sin sobrecargas) que se puede transmitir a través de una red en un momento dado. El límite superior de la tasa de transferencia es una consecuencia de compartir la capacidad del enlace entre varias conexiones. Se trata de un valor fundamental para medir el rendimiento de una conexión de red (también se puede utilizar con protocolos no orientados a conexión como UDP): un gran número de paquetes perdidos o errores durante la transmisión de mensajes pueden llegar a producir una tasa de transferencia efectiva mucho menor, lo que se traduce en un peor rendimiento.

Los principales factores que afectan a la tasa de transferencia efectiva son la congestión, la latencia y la sobrecarga de los protocolos, entre otros [5]. La razón más básica asociada a estas variaciones en la capacidad es la distancia y movilidad [6]. La calidad en la transmisión de datos mediante una red móvil depende de la distancia a la antena y de la situación: no es lo mismo un usuario andando que un usuario que viaja en coche, o que uno que esté dentro de un edificio. Si se encuentra viajando en coche, se alejará más rápidamente de la estación base, con lo que la señal se atenuará más, dando una peor relación señal a ruido. Es cierto que los terminales son capaces de adaptar la potencia de transmisión, pero es posible que la capacidad de adaptación de la que disponen no sea suficiente en casos de movilidad rápida. Otra razón obvia es la cantidad de usuarios pues, a mayor número de usuarios, peor tasa de transferencia efectiva se experimentará. También influye la banda de frecuencias utilizada, pues las ondas radioeléctricas se atenúan menos a frecuencias bajas (banda de 800 MHz) que a frecuencias altas (banda de 2600 MHz). Obviamente el *hardware* del usuario también va a influir en estas variaciones. En [7] se ha demostrado que la red mayoritariamente usada, 4G/LTE, puede ser mucho más volátil e impredecible que una red WiFi y, con mayor razón, que una red cableada. Si bien es cierto que el 5G proporciona una capacidad 10 veces superior comparado con el 4G tradicional, y una latencia 10 veces menor, las variaciones de la capacidad que experimentaron los usuarios que participaron en las primeras pruebas tras el despliegue comercial del 5G fueron mucho más dramáticas que las variaciones en el 4G/LTE [8].

Uno de los retos a los que se tienen que enfrentar los desarrolladores de aplicaciones móviles y los desarrolladores de aplicaciones de los operadores multimedia es proporcionar un alto nivel de calidad de experiencia [5] (*Quality of Experience*, QoE) bajo condiciones de acceso a la red volátiles e inestables. Uno de los tipos de aplicaciones que existen son las conocidas como «aplicaciones tolerantes al retardo», las cuales pueden absorber las variaciones en la capacidad a costa de empeorar la latencia. Este es el caso de las aplicaciones de *streaming* multimedia, las cuales generalmente mantienen un búfer de vídeo y/o de audio, permitiendo una reproducción continua y sin pausa siempre que no se vacíe dicho búfer, absorbiendo así ligeras variaciones de la capacidad instantánea disponible. Otro tipo de aplicaciones, sin embargo, son aquellas que no admiten grandes

búferes dado que requieren la menor latencia posible. Este es el caso de las aplicaciones de vídeo o televisión en directo, videoconferencias… Sin estos búferes, se debe elegir muy bien la tasa de vídeo pactada para evitar los llamados «congelamientos». En particular, en televisión en directo en línea, esta tasa de vídeo se ha de ajustar a la capacidad del enlace descendente del usuario. En videoconferencias, hay un usuario que envía su vídeo, y el resto de los participantes lo han de descargar: es necesario ajustar la tasa del vídeo transmitido a la capacidad del enlace ascendente del usuario que lo transmite, y ajustar la tasa de descarga de vídeo a la capacidad del enlace descendente de los usuarios que van a descargar dicho vídeo [7].

Para proporcionar una alta QoE para estas aplicaciones, se ha de estimar de la manera más precisa posible la capacidad de los enlaces de subida y de bajada. Si se sobreestima esta capacidad, el vídeo sufrirá «congelamientos». Sin embargo, si se subestima la capacidad del enlace, entonces la calidad del vídeo será inferior a la que podría llegar a disfrutarse [3]. En ninguno de estos dos casos el usuario estará plenamente satisfecho con el servicio que se le proporciona, pues buscará obtener la mayor calidad (resolución) del vídeo sin sufrir ningún corte por falta de datos en el búfer. En el compromiso entre ambos estará la mayor virtud. Los sistemas de *streaming* de vídeo actuales han adoptado diversos predictores en tiempo real de la capacidad disponible en los próximos segundos, los cuales permiten adaptar la tasa de vídeo del *streaming* y de las videoconferencias. Uno de los más comunes es el DASH [9] (*Dynamic Adaptative Streaming over HTTP*, Transmisión Dinámica Adaptativa sobre HTTP), que optimiza la selección de la tasa de vídeo basándose en una predicción de la capacidad TCP en una ventana de tiempo futuro de unos pocos segundos.

La predicción de la capacidad no es una tarea fácil, principalmente porque la capacidad disponible varía con el tiempo, puede ser o no ser periódica, no lineal e interdependiente a lo largo del tiempo. Además, el origen de las fluctuaciones tampoco es el mismo: desde requerimientos *hardware* o interferencias del canal inalámbrico hasta relaciones competitivas entre múltiples usuarios [10].

La estimación de la capacidad disponible en el futuro puede verse como un problema de series temporales. Las series temporales son una forma de datos que se encuentran a menudo en el mundo real. El propósito del análisis de las series temporales es descubrir información y patrones implicados en dichas series temporales, y usarlos para evaluar la serie de datos y predecir los subsiguientes valores de la serie. De hecho, los modelos estadísticos tradicionales de predicción de series temporales pueden capturar de manera precisa las variaciones en el tiempo y la naturaleza periódica de la capacidad disponible, por lo que son ampliamente utilizados en la práctica. Sin embargo, presentan más dificultades al analizar las series con «picos» y al capturar las relaciones no lineales.

Los modelos basados en redes neuronales (NN, *Neural Networks*) son muy flexibles para combatir el problema de las series temporales. Las redes neuronales recurrentes (RNN, *Recurrent Neural Networks*) suelen ser consideradas el método más efectivo para la predicción de series temporales [11]. Sin embargo, si las series temporales son de gran longitud, los modelos de redes neuronales tradicionales presentan el problema de la desaparición del gradiente, que se caracteriza por la disminución exponencial de los gradientes a medida que se propagan hacia atrás a través de la red durante el proceso de

retropropagación (*backward*). Pueden capturar fuertemente la correlación a largo plazo de las series temporales. LSTM (*Long Short Term Memory*, Memoria a Largo y Corto Plazo) se basa en una simple RNN a la que se han añadido algunas estructuras llamadas «puertas» que, principalmente, permiten resolver los problemas de la desaparición del gradiente durante el entrenamiento de largas secuencias. Esto es: LSTM permite obtener un mejor rendimiento en secuencias largas comparado con las RNN clásicas. Las RNN clásicas capturan bien las periodicidades de período corto que se presentan de manera constante durante toda la señal, pero no lo hacen con las variaciones de la señal a más largo plazo [12]. Los modelos LSTM se consideran uno de los métodos más avanzados para abordar los problemas de predicción de series temporales, por lo que son el centro de atención de muchos investigadores [12]. De acuerdo con las características y necesidades de predicción de diferentes series temporales, los modelos LSTM han ido mejorando con el paso del tiempo, y han surgido varias versiones.

Predecir la capacidad en redes WiFi y cableadas con un modelo de predicción de series temporales simple es algo factible, pero no lo es cuando se trata de predecir la capacidad en redes 4G/LTE o 5G, donde la naturaleza inestable y variante con el tiempo de dichas series hace que un simple modelo de predicción lo tenga muy difícil para capturar las características de la capacidad en todos los escenarios posibles [13]. Además, las mejoras que se aplican a estos modelos se asocian comúnmente a requerimientos de entornos específicos, lo cual dificulta que los modelos avanzados mantengan su alto rendimiento en el resto de escenarios [10]. Y todo esto teniendo en cuenta que los modelos LSTM generan una mayor carga computacional comparado con otros [14]. En trabajos como [3] o [9] se ha tratado la predicción de la capacidad de enlace con métodos clásicos, pero también con redes neuronales. En [9] llevan a cabo un análisis de la precisión de diferentes predictores, en concreto: RL (*Recursive Least Square adaptative algorithm*, Algoritmo adaptativo Recursivo de Mínimos Cuadrados), HM (*Harmonic Mean*, Media Armónica), y LSTM (*Long Short Term Memory*, Memoria a Corto y Largo Plazo), y proponen un método que elige el mejor predictor en función del escenario. En [3] utilizan RLS, RF (*Random Forest*, Bosque Aleatorio) y LSTM. En [9] proponen un modelo de predicción basado en LSTM y, además, dado que se realizan predicciones en diferentes escenarios, proponen entrenar una red LSTM para cada escenario. Igualmente, también proponen la opción de crear una red LSTM válida para todo escenario, y la opción de «entrenamiento cruzado», consistente en entrenar la red LSTM con datos de un escenario para predecir datos de otro escenario diferente. Para la opción de entrenar una red para cada escenario, proponen también un algoritmo de conmutación de modelo que selecciona el modelo que ha dado mejores resultados en el pasado reciente, aunque una forma más sofisticada que también proponen es realizar una fusión de información (*Bayes model fusion*) de múltiples sensores para dar un resultado más consistente, preciso y completo, de manera que se aproveche la información complementaria de salida de los diferentes modelos LSTM. En [3] utilizan TPA-LSTM: una extensión de LSTM con un mecanismo de atención que consulta la información de los instantes de tiempo previos y usa lo más relevante para mejorar la precisión. En [10] utilizan GRU, IndRNN, DSTP-RNN (basados en redes neuronales), StemGNN (*Spectral Temporal Graph Neural Network*, Red Neuronal de Gráfico Temporal Espectral) y diferentes variantes de LSTM (LSTM+Zoneout, EA-LSTM, TG-LSTM y TPA-LSTM).

Existen métodos alternativos, como el método de los *Transformers* [15], una arquitectura de modelo en el campo del aprendizaje automático y procesamiento del lenguaje natural caracterizada por abordar problemas como la traducción automática, el análisis de sentimientos o la generación de texto. Presenta grandes ventajas frente a las RNN en lo que respecta a la paralelización en el entrenamiento, ya que no dependen del procesamiento secuencial de la entrada, a la escalabilidad, y al rendimiento.

1.2. Objetivos

El principal objetivo de este trabajo de fin de grado es reproducir los resultados de predicción obtenidos con LSTM en la literatura [3] y entender su adecuación para el problema de predicción de capacidad de enlace.

Los objetivos parciales de este trabajo son:

- Explorar los conjuntos de datos disponibles para la predicción de capacidad del enlace y seleccionar el más adecuado.
- Explorar las variantes de LSTM utilizadas en la literatura para la predicción de capacidad del enlace, y seleccionar las más adecuadas.
- Por otro lado, también se busca entender y programar la variante LSTM seleccionada.
- Reproducir los experimentos con el conjunto de datos y variante de LSTM seleccionados comparando con algunos algoritmos básicos.
- Analizar los resultados y discutir el beneficio de la predicción con LSTM frente al coste computacional de los modelos.

Los avances realizados en este Trabajo de Fin de Grado allanarán el camino en la búsqueda de algoritmos alternativos al LSTM para mejorar la predicción de la capacidad del enlace en trabajos futuros.

1.3. Metodología

La metodología seguida es la que propone el método de ingeniería [16], un enfoque sistemático para alcanzar la solución deseada a un problema. El método de ingeniería es, por naturaleza, un proceso iterativo. Consta de las siguientes fases:

1. **Idea**. Consiste en plantear el problema. Problema que suele estar definido vagamente y es necesario evaluar su viabilidad y factibilidad. La parte más crítica es definir el problema, validar su valor, e identificar al cliente que pide una solución. En nuestro caso, el problema inicial fue la predicción de la capacidad disponible. Este problema es viable porque puede proporcionar un nuevo estado del arte y es factible porque se dispone de las herramientas software necesarias así como un servidor[1] con suficiente potencia de tarjeta gráfica.

[1] Servidor perteneciente al grupo de investigación GSIC/EMIC de la Universidad de Valladolid.

2. **Concepto**. Consiste en generar numerosos modelos que transmitan que la solución cumple con las expectativas o requisitos que impone el cliente. Se combinan los elementos de varios conceptos para encontrar un único concepto. En esta fase tenemos que tener en cuenta que ya existen trabajos previos y no se necesitan generar cosas nuevas, sólo adaptarlas al contexto de interés (si bien hemos programado ciertas cosas desde cero: predictores simples y modelo LSTM, para lograr un mejor entendimiento y aprendizaje de lo que se está llevando a cabo).
3. **Planificación**. Consiste en definir el plan de implementación: definir las tareas, duración de las mismas, dependencias de las tareas, interconexiones de las tareas, presupuesto… En este caso concreto, puesto que se trata de un Trabajo de Fin de Grado, es un trabajo individual y no tiene presupuesto. La planificación se realizó al comenzar el curso académico 2023/2024, donde se establecieron las horas necesarias de formación para poder abordar con éxito este trabajo. En concreto, los primeros meses fueron puramente formativos: realización de cursos de aprendizaje automático, lectura y comprensión de artículos científicos, aprendizaje y familiarización con el lenguaje *Python* y sus librerías de aprendizaje automático y manipulación de datos… El plan de estudios[2] asigna 6 ECTS al Trabajo de Fin de Grado: aproximadamente 150 horas, si bien la dedicación total en horas ha sido aproximadamente el doble por la obtención de una beca de investigación. La planificación fue la siguiente:
    a. Introducción al aprendizaje automático, mediante la realización de los correspondientes cursos y tutoriales.
    b. Estudio a fondo de los artículos que tratan modelos de predicción de la capacidad disponible.
    c. Estudio de los subconjuntos de datos disponibles públicamente.
    d. Programación de modelos de predicción clásicos y basados en redes neuronales (LSTM). Reutilización de trabajos previos y adaptación al contexto actual. Realización de gráficas y pruebas convenientes para entender lo mejor posible todo lo que está sucediendo.

    Para llevar cuenta de las horas dedicadas, se ha hecho uso de una hoja de cálculo que permite contabilizarlas e indicar la tarea hecha en ellas, para así llevar una buena planificación y poder darse cuenta de tareas que están llevando un tiempo excesivo.

4. **Diseño**. Consiste en especificar detalles y establecer especificaciones. El propósito de esta fase es traducir los requisitos del cliente y el modelo en especificaciones de ingeniería con las que un ingeniero pueda trabajar para diseñar y construir un prototipo funcional. Es en esta fase donde se diseña cada uno de los experimentos que se abordan en este trabajo.
5. **Desarrollo**. Consiste en generar documentación de ingeniería: esquemas, código fuente… y otra información en un prototipo funcional que demuestre ser una solución al problema. La solución puede ser un prototipo funcional tangible o una simulación funcional [16]. Esta etapa ha sido, sin duda, la más larga. Una vez obtenidos los subconjuntos de datos, se realiza de manera iterativa:

[2] Grado en Ingeniería de Tecnologías de Telecomunicación, Universidad de Valladolid, plan 460.

a. Diseño del código del modelo predictor de estudio.
b. Depuración de errores.
c. Conclusiones de los resultados. En función de estas conclusiones, reajuste de parámetros o modelo y vuelta a empezar.

6. **Lanzamiento**. Consiste en la entrega de documentación y diseño de ingeniería a las instalaciones de fabricación para su producción, con un prototipo funcional. En nuestro caso, no se tiene por objetivo el hacer un producto final, sino que el objetivo es explorar nuevas técnicas de predicción que podrían usarse posteriormente en sistemas reales.

Es importante recalcar que en este caso hay etapas que no se han llegado a completar. El objetivo no es crear un producto, sino estudiar diferentes modelos y conseguir un aprendizaje sólido en este ámbito.

1.4. Estructura del documento

En cuanto a la organización de este documento, en el Capítulo 2 se comienza explicando el fundamento teórico de las redes móviles. Tras una breve perspectiva histórica de la evolución de las redes de telefonía hasta el 4G y 5G (explicando los fundamentos de cada generación) se detallan con mayor profundidad los detalles de la arquitectura de la red 4G. También se dan nociones sobre las técnicas de modulación que se usan. Se comenta el problema de variabilidad de la capacidad en redes móviles, y trabajos relacionados. A continuación, se especifica cómo se puede ver la predicción de la capacidad como un problema de series temporales. Se especifican diferentes técnicas de predicción clásicas (*Naive*, EWMA, *Harmonic Mean*). Se dan también unas nociones sobre el aprendizaje automático y técnicas de aprendizaje automático basadas en redes neuronales. Se describen los conjuntos de datos públicos que se pueden encontrar para la predicción de la capacidad del enlace. Por último, se especifican las métricas de evaluación de rendimiento que se van a utilizar.

El Capítulo 3 comienza describiendo el *dataset* que se va a utilizar en los sucesivos experimentos. Se comenta qué *software* y qué *hardware* se emplea, y se detalla cómo se ha llevado a cabo la implementación de los algoritmos de predicción. A continuación, se muestran los resultados de los diferentes métodos y arquitecturas de predicción para cada línea de transporte público. Por último, se realizan unas discusiones y se extraen conclusiones.

En el Capítulo 4 se presentan las conclusiones y posibles líneas futuras de este trabajo.

Capítulo 2

Estado del arte

2.1. Introducción

Tal y como se indica en el capítulo de introducción, en las redes móviles surge el problema de predecir la capacidad disponible en los siguientes instantes de tiempo. El objetivo de este proyecto es realizar una predicción de la capacidad disponible la manera más precisa posible buscando un compromiso con la carga computacional. El principal objetivo es reproducir los resultados obtenidos por [3]. Para poder cumplir con este objetivo, es necesario, en primer lugar, profundizar un poco más en las características del problema en cuestión.

Este trabajo forma parte de la hipótesis de que los predictores basados en LSTM son buenos predictores, por lo que el grueso de nuestro estudio se centra en predictores LSTM, si bien es cierto que, dada la elevada carga computacional que requieren [12], se verá que en determinadas situaciones convendría conformarse con predictores más simples.

En primer lugar, en este capítulo se va a hablar de las redes móviles inalámbricas y sus problemáticas. Por otra parte, se va a profundizar en el problema de la predicción de la capacidad disponible, para lo cual se van a estudiar diversos métodos y métricas para evaluar el rendimiento de cada uno.

2.2. Las redes móviles

2.2.1. Introducción a las redes móviles

Las redes de comunicación celulares (comúnmente conocidas como redes móviles) [2] son parte de nuestro día a día. Su crecimiento ha sido increíble y se considera uno de los principales logros tecnológicos del siglo XX [2]. No hay más que observar a nuestro alrededor: en algunos países cada persona tiene incluso dos o más teléfonos móviles. El crecimiento ha sido tan grande que ha roto todas las expectativas de los analistas. El origen de las redes móviles se remonta a 1947 en los laboratorios Bell, donde un documento interno propuso la idea de elaborar un sistema usando cierto número de transmisores de baja potencia en «celdas» [2], con el objetivo de reutilizar frecuencias en un sistema móvil de radio o telecomunicaciones.

La idea fundamental es dividir el área que se pretende cubrir en células cuyo tamaño varía dependiendo del lugar [2]. Las zonas de difícil acceso (como los túneles), para evitar la pérdida de cobertura, conforman microcélulas. Cada célula emplea un rango diferente de canales de frecuencia para evitar colisiones con las células adyacentes. Existen tres técnicas clásicas de acceso radio: Acceso Múltiple por División en Frecuencia (FDMA, *Frequency Division Multiple Access*), Acceso Múltiple por División en el Tiempo (TDMA, *Time Division Multiple Access*) y Acceso Múltiple por División en el Código (CDMA, *Code Division Multiple Access*) (se detallarán posteriormente), las cuales permiten combinar señales de diferentes fuentes en un medio de transmisión común, de forma que, en los destinos, los diferentes canales puedan ser separados sin interferencia mutua.

En dicho documento interno [2] se mencionó la necesidad de encontrar un método para lo que se conoce como traspaso o *handing over*, esto es, el traspaso de la estación móvil de una celda a la siguiente a medida que se desplaza, aunque no especificó cómo llevarlo a cabo. Durante unos años, esta propuesta se ignoró. Si bien es cierto que ya se usaban algunos teléfonos móviles, estos conformaban un sistema de telecomunicaciones muy simple basado en transmitir en una determinada frecuencia, y en recibir en otra. Naturalmente existía una limitación: el número de canales disponibles. En la década de los 1960 y 1970, otros países empezaron a considerar seriamente la posibilidad de crear un sistema de telecomunicaciones celular. Tras varios logros intermedios, no fue hasta 1983 cuando se comenzó a comercializar completamente un sistema bajo el estándar «Servicio Avanzado de Telefonía Móvil» (AMPS, *Advanced Mobile Phone Service*) en EEUU. En la región escandinava, el primer sistema lanzado comercialmente fue bajo el estándar «Teléfono Móvil Nórdico» (NMT, *Nordic Mobile Telephone*), lanzado en 1980, utilizando una banda de frecuencias en torno a 450 Hz. Paralelamente, en otros países se utilizó la propuesta de Motorola: «Sistema de Comunicaciones de Acceso Total» (TACS, *Total Access Communications System*). Las redes de telecomunicación celulares se expandieron por todo el mundo, y otros muchos estándares aparecieron.

Todos estos se pueden englobar dentro de la **Primera Generación (1G)**. Se trataba de sistemas analógicos con ciertas desventajas, como la muy baja calidad o la baja capacidad (pocos usuarios). Únicamente se podían prestar servicios de voz. La técnica de acceso al medio que usaba fue FDMA/FDD. **FDMA** (Acceso Múltiple por División en Frecuencia, *Frequency Division Multiple Access*) [17] (ver Figura 2.1) es una técnica de acceso al medio que consiste en asignar una frecuencia diferente a cada transmisión durante todo el tiempo. **FDD** (Dúplex por División en Frecuencia, *Frequency Division Dúplex*) es una técnica de funcionamiento de dispositivos dúplex que permite que ambos sentidos transmitan simultáneamente, pero en bandas de frecuencia distintas. Por las mencionadas desventajas, se fue dando paso a la idea de utilizar sistemas digitales.

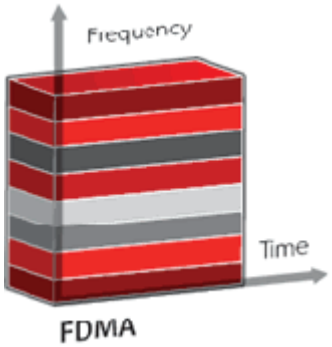

Figura 2.1. Acceso Múltiple por División en Frecuencia (FDMA). Figura tomada de [17].

En Europa, el principal problema de 1G era el gran número de estándares incompatibles entre sí. En EEUU la principal queja era relativa a la baja capacidad de usuarios. Como consecuencia, aparecieron nuevos estándares que, mediante sus propuestas, solventaban los problemas previamente mencionados, llegando la **Segunda Generación (2G)**. Una de las mayores propuestas fue la europea «Grupo Especial Móvil» (*Groupe Spéciale Mobile*), renombrada posteriormente a «Sistema Global para Comunicaciones Móviles» (GSM, *Global System for Mobile Communications*). Se comenzó a trabajar en esta idea en el año 1982, y fueron un total de 26 compañías de telecomunicaciones las que cooperaron en su desarrollo. Comenzó a operar a mediados de 1991 en la banda de 900 MHz. Se trataba ya de un **sistema digital**. La técnica de acceso al medio que utilizaba era **TDMA** (Acceso Múltiple por División en el Tiempo, *Time Division Multiple Access*, ver Figura 2.2), que consiste en segmentar el tiempo en que los usuarios pueden acceder al medio para transmitir datos, con lo que todos los usuarios comparten la totalidad del ancho de banda durante los períodos de tiempo en los que se les permite. Se consiguió proporcionar una mayor calidad en las transmisiones de voz y una mayor capacidad de usuarios. Con GSM apareció el servicio de mensajería «Servicio de Mensajes Cortos» (SMS, *Short Message Service*), e incluso existía la posibilidad de navegar mediante WAP (Protocolo de Aplicaciones Inalámbricas, *Wireless Application Protocol*), pero la máxima velocidad de transmisión datos no superaba los 9.6 Kbps.

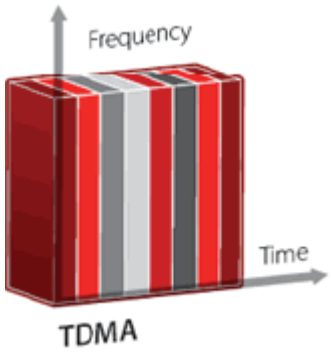

Figura 2.2. Acceso Múltiple por División en Tiempo (TDMA). Figura tomada de [17].

En la década de 1990 la industria de la telefonía móvil había alcanzado una madurez notable, y los analistas determinaron que, debido al creciente uso de Internet, la gente querría disponer de más servicios de datos. La primera extensión de la segunda

generación se conoce como *General Packet Radio System* (GPRS, Sistema General de Radio por Paquetes), y su mejora *Enhaced Data rates for Global Evolution* (EDGE, Tarifas de Datos Mejoradas Para la Evolución Global). Estos sistemas se constituyen las **Generaciones 2.5G y 2.75G**. La novedad que introdujo GPRS fue la de permitir tarificación por tráfico, además de permitir una velocidad neta de 14.4 Kbps por ranura temporal (*time slot*, ranuras en las que el tiempo se divide de acuerdo con TDM - *Time Division Multiplex* o Multiplexación por División en el Tiempo), lo que, combinando 8 ranuras temporales, permitiría una capacidad total de 115.2 Kbps. Por su parte, EDGE permitía una capacidad de 59.2 Kbps por ranura temporal, y definió 9 esquemas de modulación (la velocidad por ranura temporal variaba de un esquema de modulación a otro). Además, los cambios *hardware* y *software* de adaptación a EDGE fueron mínimos: únicamente hubo que adaptar la red para el uso de la nueva modulación 8-PSK.

Posteriormente emergió la **Tercera Generación (3G)**: en Europa apareció un sistema llamado «Sistema de Telecomunicaciones Móviles Universal» (UMTS, *Universal Mobile Telecommunications System*), utilizando CDMA de banda ancha, mientras que en EEUU se utilizó CDMA2000. En China se usó «CDMA por División en el Tiempo Síncrona» (TD-SCDMA, *Time Division Synchronous CDMA*). Todas eran técnicas de acceso basadas en **CDMA** (*Code Division Multiple Access*, Acceso Múltiple por División en el Código, ver Figura 2.3), con lo que se asignaba todo el espectro durante todo el tiempo a cada usuario, diferenciado del resto mediante la utilización de códigos ortogonales que se utilizaban para codificar la señal de información a transmitir. El receptor, conociendo estas secuencias, decodifica las señales y regenera los datos originales. Permitía un mayor número de usuarios por célula que sus predecesores. Al igual que en la generación anterior, surgieron mejoras que se conforman como parte de la **Generación 3.5G**, entre las que se incluyen *High Speed Downlink Packet Access* (HSDPA, Acceso de Descarga de Paquetes a Alta Velocidad, con velocidades de bajada de hasta 14 Mbps), *High Speed Uplink Packet Access* (HSUPA, Acceso de Subida de Paquetes a Alta Velocidad, con velocidades de subida de hasta 7.2 Mbps) y HSPA+ (o *Evolved HSPA*, con velocidades de hasta 84 Mbps de bajada y 22 Mbps de subida).

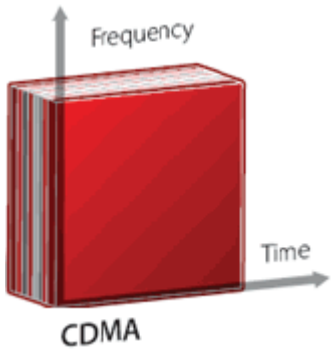

Figura 2.3. Acceso Múltiple por División en el Código (CDMA). Figura tomada de [17].

La **Cuarta Generación (4G)** llegó a finales de la década de los 2000. La Unión Internacional de Telecomunicaciones (UIT) creó un comité para definir las especificaciones del 4G, basada completamente en el protocolo IP (usa conmutación de paquetes en todos los servicios). En dicho comité se empezaron a estudiar a los distintos candidatos a estándar, entre los que estaba el *Long Term Evolution* (LTE, Evolución a

Largo Plazo) de la norma 3GPP (*3rd Generation Partnership Project*). En torno a 2018 y 2019 llegó la quinta generación (5G).

El *dataset* utilizado en este Trabajo de Fin de Grado (véase el Capítulo 3) contiene medidas tomadas de una red 4G, por lo que, a continuación, se detalla un poco más la arquitectura de una red LTE [17]. La diferencia fundamental con respecto a las generaciones anteriores es que, en LTE, todos los servicios utilizan conmutación de paquetes. Es importante destacar que LTE es compatible con tecnologías anteriores, lo cual supone una enorme ventaja. Utiliza tecnología MIMO (*Multiple Input Multiple Output*, Múltiples Entradas Múltiples Salidas), esto es, varias antenas en transmisor y receptor. Una configuración MIMO común es 2x2, permitiendo capacidades de hasta 173 Mbps de descarga y de hasta 86 Mbps de subida. Admite dos modos de duplexión: FDD (*Frequency Division Duplex* o «Dúplex por División en Frecuencia», en el que ambos sentidos de comunicación se transmiten simultáneamente, pero en bandas de frecuencias distintas) y TDD (*Time Division Duplex* o «Dúplex por División en Tiempo», en el que cada sentido de la comunicación transmite en instantes diferentes de tiempo y en intervalos suficientemente pequeños como para no producir discontinuidades en la comunicación). Para la descarga, se utiliza la tecnología OFDMA (*Orthogonal Frequency Division Multiple Access*, Acceso Múltiple por División en Frecuencia Ortogonal, se detallará más adelante) y, para la subida, SC-FDMA (*Single Carrier – Frequency Division Multiple Access*, Acceso Múltiple por División en Frecuencia de Única Portadora, se detallará más adelante). En cuanto a la arquitectura [17], la conforman (véase la Figura 2.7):

- **Equipo de usuario (UE)**. Permite a los usuarios del sistema LTE acceder a los servicios de la red a través de la interfaz radio. Cada usuario se identifica en la red (independientemente del equipo móvil utilizado) mediante la tarjeta inteligente UICC (*Universal Integrated Circuit Card*): SIM (*Suscriber Identity Module*), USIM (*User Services Identity Module*) o ISIM (*IP Multimedia Services Identity Module*). El equipo móvil ME (*Mobile Equipment*) se puede dividir en dos partes para mayor flexibilidad: el MT (*Mobile Terminal*, que alberga las funciones propias de la comunicación) y el TE (*Terminal Equipment*, que se ocupa de la interacción del usuario con los servicios de red).

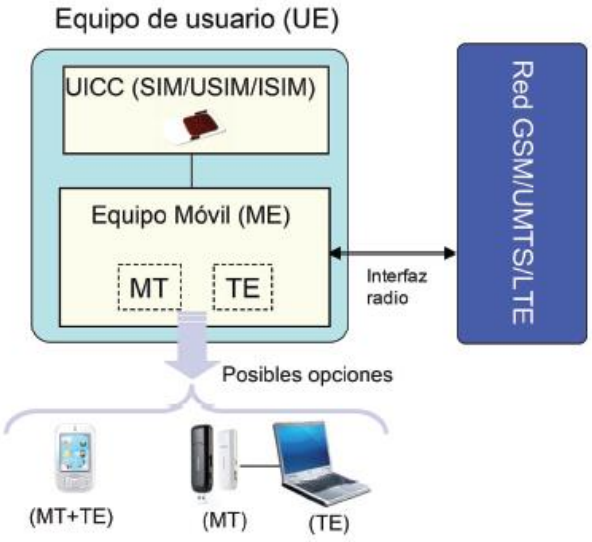

Figura 2.4. El equipo de usuario. Figura tomada de [17].

- **Red de acceso (E-UTRAN)**. La red de acceso se compone por una única entidad de red: los eNB (*evolved NodeB*), que se comunican con el resto de elementos a través de tres interfaces:
    - **Uu**. Para la transferencia de información por el canal radio entre eNB y UE.
    - **S1**. Para la conexión entre eNB y red troncal EPC (incluye S1-U para la transferencia de paquetes IP y S1-MME para mensajes de control).
    - **X2**. Opcional, para la conexión entre eNB y para transferir mensajes de control y tráfico de usuarios durante un proceso de *handover*[3].

  Los eNB integran todas las funciones de la red de acceso. Almacenan la información sobre todos los UE que tienen conectados, gestionan los recursos radio (decisiones de *handover*, asignación dinámica de recursos radio en los enlaces ascendente y descendente…), seleccionan la entidad MME (*Mobile Management Entity*, Entidad de Gestión Móvil, explicado en el siguiente párrafo) de la red troncal EPC (véase el siguiente párrafo también) para balancear la carga y aumentar la robustez, y permiten el envío y recepción de paquetes IP de los usuarios a diferentes pasarelas S-GW (véase el siguiente párrafo) de la red troncal EPC. Por estas razones la red LTE no puede asegurar una capacidad constante al igual que sí puede hacerlo una red cableada.

[3] Se denomina *handover* o «traspaso» [2] al sistema que permite la transferencia del servicio de una estación base a otra cuando la calidad del enlace con una de las estaciones es insuficiente.

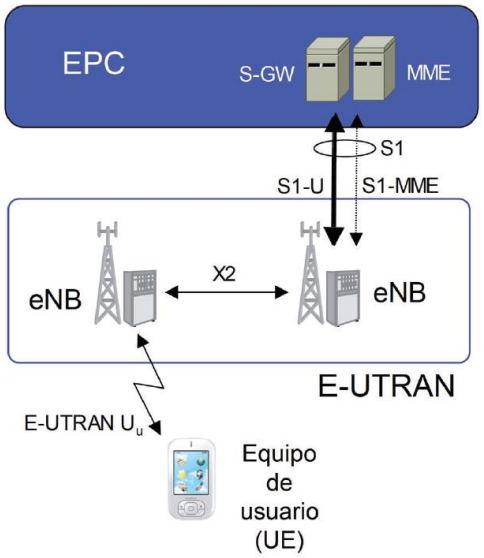

Figura 2.5. Red de acceso. Figura tomada de [17].

- **Red troncal (EPC)**. Su misión es proporcionar un servicio de conectividad IP, facilitando la interconexión con otras redes. Su núcleo lo conforman tres entidades lógicas:

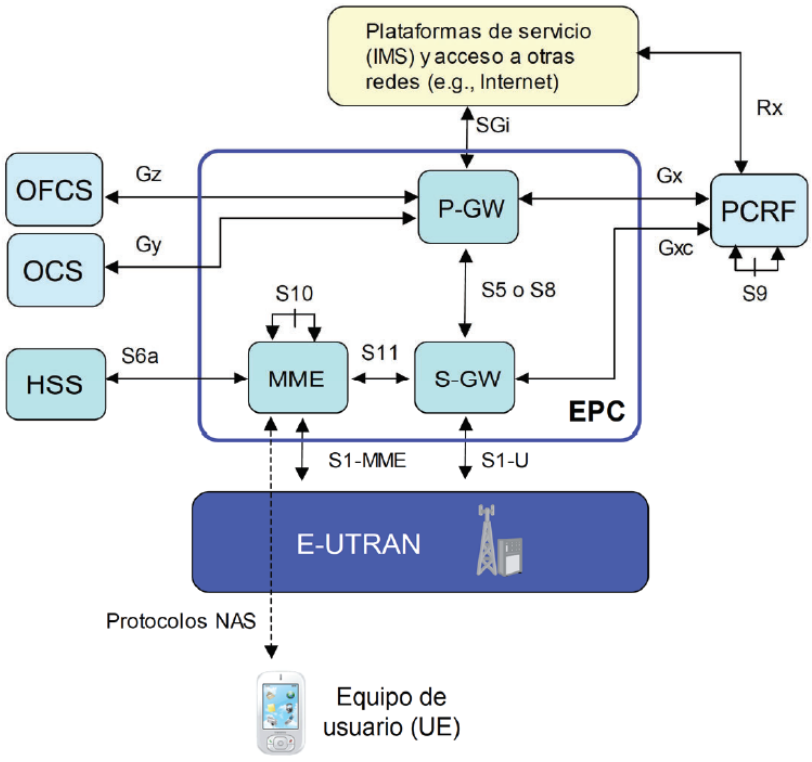

Figura 2.6. Red troncal. Figura tomada de [17].

  - **MME** (*Mobile Management Entity*). Es el elemento principal del plano de control. Cada UE tiene un MME asociado, aunque puede ir cambiando debido a la movilidad. Su objetivo es mantener un contexto de datos de usuario (identificadores, servicios portadores activos, claves de

seguridad, localización…) para autenticar y autorizar el acceso de los terminales a la red, gestionar los servicios portadores sobre los que se sustenta el envío de paquetes IP, gestionar la movilidad de los usuarios que no tienen ninguna conexión de control establecida con la red E-UTRAN (es decir, los que están en modo *idle*), y realizar funciones de control para el soporte de movilidad con otras redes.

- **S-GW** (*Serving Gateway*). Es la pasarela del plano de usuario entre E-UTRAN y la red EPC. Cada UE tiene una S-GW asociada. Se encarga de la gestión de movilidad con otras redes y de encaminar el tráfico de usuario. Proporciona un punto de anclaje en la red EPC respecto a la movilidad de los terminales entre eNBs. Almacena, temporalmente, los paquetes IP de usuarios en modo *idle*.
- **P-GW** (*Packet Data Network Gateway*). Proporciona conectividad entre la red LTE y otras redes externas. Cada usuario tiene asignada como mínimo una pasarela P-GW desde su registro en la red LTE.

La base de datos del sistema se conoce como HSS (*Home Subscriber Server*), que almacena identificadores universales del usuario, identificadores de servicio, información de seguridad y cifrado, información de localización del terminal en la red e información necesaria para la provisión de los servicios de la red. Existen, además, otras tres entidades relacionadas con la tarificación: PCRF (*Policy and Charging Rules Function*), OFCS (*Offline Charging System*) y OCS (*Online Charging System*).

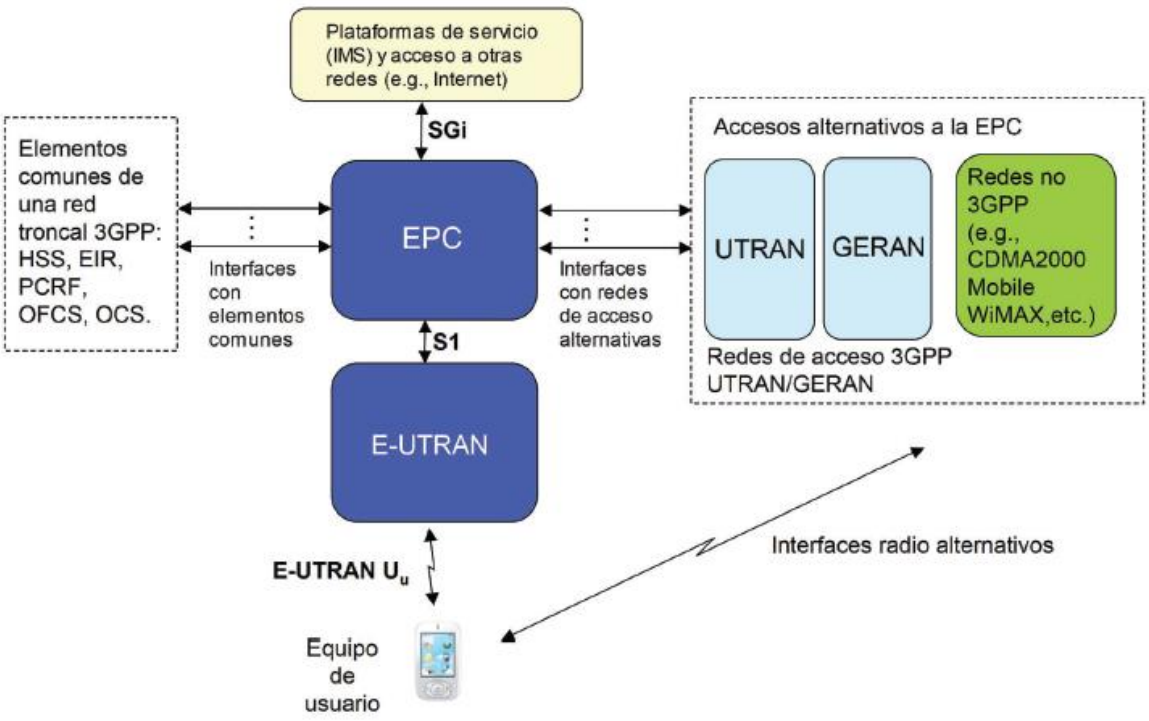

Figura 2.7. Arquitectura de la red 4G. Figura tomada de [17].

En cuanto a la técnica de acceso, LTE utiliza OFDMA (*Orthogonal Frequency Division Multiple Access*). OFDMA es una técnica de acceso basada en OFDM (*Orthogonal Frequency Division Multiplex*), que es una técnica de transmisión multiportadora que consiste en multiplexar un conjunto de símbolos sobre un conjunto de subportadoras ortogonales. De esta forma, es posible efectuar la transmisión simultánea de todos los símbolos manteniendo la capacidad de separación de los mismos en recepción.

$$x_k(t) = e^{j2\pi k\Delta f t} rect_{Ts}(t) \text{ con } 0 \leq k \leq K-1$$

Siendo $f_k = k\Delta f$ la frecuencia de la subportadora k-ésima, y $rect_{Ts}$ representa un pulso rectangular de duración Ts. Nótese que dos subportadoras son ortogonales si su producto escalar es nulo en el intervalo temporal *Ts*. LTE utiliza OFDMA, que es igual que OFDM pero considera que los diferentes símbolos modulados sobre las subportadoras pertenecen a usuarios distintos. OFDMA permite diversidad multiusuario, diversidad frecuencial, mayor robustez, flexibilidad en la banda asignada (permite adaptar **diferentes velocidades de transmisión** a diferentes usuarios asignando más o menos portadoras por usuario), elevada granularidad en los recursos asignables (pues subdivide la banda total en subportadoras de banda estrecha que se asignan dinámicamente), elevado uso de la banda asignada, y gran sencillez de implementación. Para la gestión de los recursos radio, posee mecanismos que permiten conseguir un uso eficiente de estos recursos: ***scheduling* de paquetes** (responsable de asociar las parejas «subportadora / período de tiempo» a cada usuario, teniendo en cuenta requisitos de QoS de los usuarios e información sobre el estado del canal), y **adaptación del enlace** (estrategia cuyo objetivo es extraer el máximo rendimiento del canal, entendiéndose «rendimiento» como velocidad de transmisión, mediante la selección de la modulación que permita el mayor número de bits por símbolo). En sistemas celulares, se utilizan técnicas de reutilización de frecuencias en las que las subportadoras se dividen en F grupos, siendo F el factor de reuso. Por ejemplo, si F=3 y las células son hexagonales, véase la Figura 2.8. Aumentando F, se reduce la interferencia intercelular, pero también el número de subportadoras por célula.

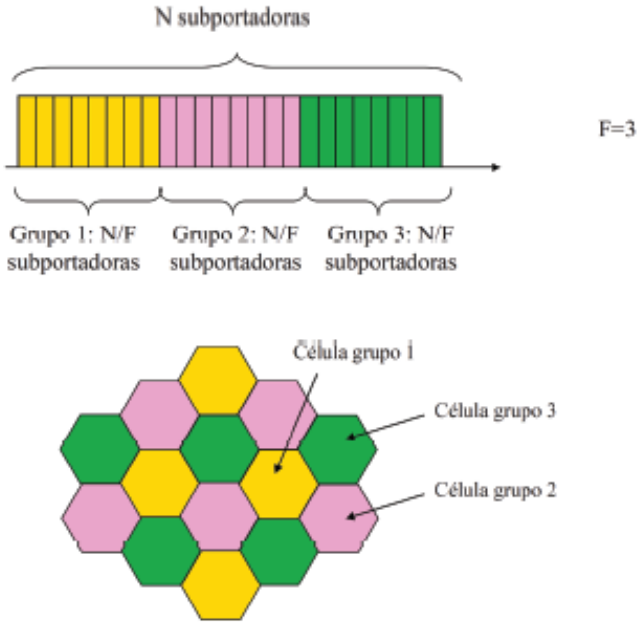

Figura 2.8. OFDMA en sistemas celulares. Figura tomada de [17].

En el enlace ascendente de LTE se utiliza **SC-FDMA**, que consiste en una versión precodificada con DFT de los símbolos a transmitir previa al proceso de transmisión OFDM. Mientras que en OFDM cada símbolo se transmite en paralelo, en SC-FDMA cada símbolo de transmite secuencialmente. Su rendimiento se mide con el factor de cresta o PAPR (*Peak to Average Power Ratio*): a mayor factor, mayor eficiencia, menor consumo de potencia y, por tanto, mayor duración de las baterías. Permite las mismas modulaciones que OFDMA (QPSK, 16QAM y 64QAM).

2.2.2. Variabilidad de la capacidad disponible para el usuario en las redes móviles

Las redes móviles no brindan la estabilidad que sí pueden ofrecer las redes cableadas. La razón más básica de esta inestabilidad es la distancia del usuario a la antena [18]. Es obvio que, a mayor distancia, peor será la relación señal a ruido en el dispositivo de usuario, empeorando la calidad del enlace. La movilidad de los usuarios también influye, pues a medida que el usuario se desplaza no va a disponer de la misma capacidad: si se aleja de la antena, el enlace empeora, disminuyendo la capacidad del enlace. Si el usuario es un peatón, tardará más en perder la máxima capacidad de conexión que si el usuario se traslada en un vehículo. Si el usuario se está moviendo, va a salir más rápido del área de cobertura de una celda, empeorándose la calidad de la conexión. Dependiendo de la velocidad a la que se mueva, es posible que el proceso de *handover* no sea lo suficientemente rápido como para hacer el traspaso correctamente, tardando más y, por tanto, afectando a la capacidad disponible.

Por otro lado, también va a influir la cantidad de usuarios a los que la misma antena está proporcionando servicio. Si la estación base tiene que asignar canales a cada usuario (con FDM o TDM) puede no haber canal. Con OFDM pasa lo mismo, pero con las portadoras. Además, la capacidad del enlace de salida de la estación base también tiene que ser compartida, lo que afecta al rendimiento percibido en el terminal. Si hay muchos usuarios móviles conectados a la antena de telefonía, se podría perder capacidad de bajada

y/o subida. Si bien es cierto que el enlace radio es el más propenso a introducir estas variaciones, el «cuello de botella» puede no estar en este enlace radio, y estar en la conexión de la antena con la red central. En ese caso también se ve afectada la capacidad de la que dispone el usuario de la red móvil.

La capacidad disponible está íntimamente relacionada con indicadores [19] que se toman como información de contexto en muchos de los *datasets* disponibles. En concreto, el RSSI (*Received Signal Streght Indicator*, Indicador de Fuerza de Señal Recibida) da cuenta de la potencia de señal recibida por un dispositivo, siendo crucial en tecnologías inalámbricas (entre las que se incluyen las redes móviles) para evaluar la calidad de conexión y, por consiguiente, la capacidad que puede brindar dicha conexión. Generalmente se expresa en dBm (decibelios de milivatio). El RSRP (*Reference Signal Received Power*, Potencia de Señal de Referencia Recibida) es otro indicador clave de la calidad en redes LTE (*Long Term Evolution*) que mide la potencia recibida de las señales de referencia que son transmitidas por una celda específica. A diferencia de RSSI, que mide la potencia total de la señal recibida, incluyendo la interferencia y el ruido, RSRP mide específicamente la potencia de las señales de referencia (a efectos prácticos, la señal «útil»). Durante la movilidad del usuario, es el RSRP el indicador que ayuda a decidir cuándo el dispositivo tiene que hacer un traspaso (*handover*) de una celda a otra. EL TA (*Timing Advace*, Avance de Tiempo) indica el tiempo necesario para alcanzar el siguiente *ENodeB* (interfaz de conexión de la red móvil con los equipos de usuario), y asegura que las señales enviadas por los dispositivos móviles lleguen a la estación base en el momento adecuado sin que se produzcan colisiones. Por tanto, también tiene su efecto en la capacidad disponible. El RSRQ (*Reference Signal Received Quality*) es un indicador que evalúa la calidad de la señal recibida, y complementa otros indicadores como RSRP y RSSI. Es adimensional (se mide en dB). Tiene en cuenta la potencia de la señal y la interferencia y ruido presentes. La banda de frecuencia en la que se efectúa la conexión también influye en la potencia de la señal recibida, pues hay frecuencias en las que la señal se atenúa menos al propagarse y otras en las que se atenúa más, influyendo en la capacidad [2].

2.2.3. Predicción de la capacidad en la literatura

La predicción de la capacidad en tiempo real es un problema serio que diferentes áreas de investigación relacionadas con las telecomunicaciones y el aprendizaje automático están intentando solucionar. En [9] abordan la predicción de la capacidad en diferentes rutas de transporte público de Nueva York en diferentes momentos del día. El objetivo es predecir la capacidad a 1 segundo vista, por un lado, y por otro lado a varios segundos vista, utilizando en ambos casos solamente valores anteriores de la capacidad. Para llevar a cabo la predicción, utilizan trazas de duración entre 500 y 1000 segundos. Proponen un modelo de predicción basado en LSTM y, además, dado que se realizan predicciones en diferentes escenarios, proponen entrenar una red LSTM para cada escenario. Plantean la opción de crear una red LSTM válida para todo escenario, y la opción de «entrenamiento cruzado», consistente en entrenar la red LSTM con datos de un escenario para predecir datos de otro escenario diferente. Para la opción de entrenar una red para cada escenario, proponen también un algoritmo de conmutación de modelo que selecciona el modelo que ha dado mejores resultados en el pasado reciente, aunque

una forma más sofisticada que también proponen es realizar una fusión de información (*Bayes model fusion*) de múltiples sensores para dar un resultado más consistente, preciso y completo, de manera que se aproveche la información complementaria de salida de los diferentes modelos LSTM. En [3] utilizan un *dataset* multivariable, por lo que llevan a cabo la predicción utilizando los valores anteriores de la capacidad junto con otras variables de estado de la red como la potencia de señal recibida (RSSI, *Received Signal Strength Indicator*), calidad de señal recibida (RSRQ, *Reference Signal Received Quality*), banda de frecuencia… Como novedad, utilizan TPA-LSTM: una extensión de LSTM con un mecanismo de atención que consulta la información de los instantes de tiempo previos y usa lo más relevante para mejorar la precisión.

En [10] abordan la predicción de la capacidad en una red 5G en la que los usuarios se mueven en transporte público (metro) y en coche, combinando diferentes *datasets* públicos. Encuentran que el mismo modelo de predicción exhibe diferente rendimiento en función de la traza considerada. Esto es: demuestran que la capacidad en los diferentes escenarios tiene sus propias características, por lo que proponen un esquema basado en predictores en tiempo real multivariables y clasificación de escenario con conmutación al modelo óptimo. Utilizan GRU, IndRNN, DSTP-RNN (basados en redes neuronales), StemGNN (*Spectral Temporal Graph Neural Network*) y diferentes variantes de LSTM (LSTM+Zoneout, EA-LSTM, TG-LSTM y TPA-LSTM).

En [20] [21] se usa la media armónica de la capacidad o *thoughput* TCP para descargar los segmentos previos y predecir los siguientes segmentos. En [22] se usa un filtro adaptativo, *Recursive Least Squared* (RLS), para hacer predicciones en escenarios celulares. En cuanto a modelos estadísticos y de aprendizaje automático convencionales, en [23] los autores proponen entrenar un modelo SVR (*Support Vector Regression*) para estimar la capacidad TCP. En el contexto del *streaming* DASH, en [24] utilizan un algoritmo de predicción para seleccionar la tasa binaria y un modelo SVR personalizado para selección de servidor DASH. En el contexto de las videoconferencias, en [25] se modela el enlace móvil como una cola de único servidor conducida por un proceso doble-estocástico, y la predicción de la capacidad futura se genera por inferencia estadística basada en el modelo de cola de único servidor. En [26] se plantea utilizar un modelo basado en *Random Forest* para realizar predicciones en tiempo real sobre la red LTE basándose en información de contexto. Dado que los modelos estadísticos tradicionales no pueden capturar patrones implícitos en estructuras de información ricas y complejas, otros autores también han trabajado con modelos basados en aprendizaje profundo. Así, por ejemplo, en [27] también se ha desarrollado un método basado en LSTM para estimar la capacidad basándose en las medidas previas. Diversos autores han discutido también la fiabilidad de los modelos LSTM en escenarios generalizados.

2.3. Predicción de series temporales

2.3.1. Introducción a las series temporales

Tal y como se ha comentado, sería conveniente conocer con cierta antelación (1 a 3 segundos) la capacidad del enlace que va a estar disponible, de manera que las aplicaciones puedan tenerla en cuenta y ajustar la cantidad de información transmitida

(reduciendo o aumentando la calidad del vídeo, por ejemplo) ya que, de no hacerlo, la experiencia final del usuario empeorará. Sin embargo, como ya se ha discutido, la capacidad del enlace es variable, debido, en parte, al carácter inalámbrico de los enlaces y a su carácter asimétrico en ocasiones [28], lo que dificulta conocer de antemano qué capacidad estará disponible.

Trabajos previos [9] afirman que el seguimiento de la capacidad permite a las aplicaciones tomar las decisiones más adecuadas para proporcionar la mejor QoE (*Quality of Experience*, Calidad de Experiencia). No obstante, habrá otros factores que influyan en la capacidad, no solo relativos a la separación temporal entre el instante actual y el instante en el que se produzca la predicción.

En este sentido, parece razonable que, si la capacidad varía a lo largo del tiempo, se puedan realizar predicciones a corto plazo, ya que mejoraría la toma de decisiones por parte de aplicaciones con el conocimiento del futuro próximo de la red, que es cuando se necesita optimizar el *bitrate*, por ejemplo, y no con la información actual, que es posible que no sea válida en el futuro.

Como el problema que se pretende abordar es el de predecir la capacidad disponible en el futuro a partir de la capacidad disponible en el pasado y, posiblemente, de los valores de otras métricas del estado de la red en el pasado, no resulta difícil formular este problema de predicción como un problema de series temporales. Sea *b(t)* la capacidad disponible para un usuario en el instante de tiempo *t*. El dispositivo móvil del usuario mide periódicamente la calidad del enlace de acceso a la red móvil con cierta frecuencia, por ejemplo, cada segundo. Se obtiene, por consiguiente, una serie temporal discreta:

$$\{X(t), t = 1, 2, 3, \dots\} \text{ con } X(t) \in \mathbb{R}^n$$

X(t) es un vector de *n* tipos de información medida, incluyendo *b(t)* y otras métricas sobre la conexión. El verdadero problema de la predicción de la capacidad en tiempo real en el instante *t* es estimar la capacidad disponible en algún instante de tiempo futuro $t+\tau$ conocida la colección de medidas hasta el instante *t*:

$$\hat{b}(t+\tau) = \boldsymbol{f}(\{X(k), k = 1, 2, 3, \dots, t\})$$

Siendo τ el valor del horizonte de tiempo futuro. Para la predicción basada en el historial reciente, se usará solo $\{X(t-w+1), \dots, X(t)\}$ *p*ara predecir $\mathrm{b}(\mathrm{t}+\tau)$, siendo *w* el tamaño de la ventana deslizante. En predicción de capacidad univariable solo se usarán las medidas de capacidad del pasado para predecir capacidad futura. Esto es:

$$\hat{b}(t+\tau) = \boldsymbol{f}^{(\boldsymbol{u})}(\{b(k), k = 1, 2, 3, \dots, t\})$$

En predicción de capacidad con entrada multivariable se usará, además de las medidas de la capacidad en el pasado, información de canal y de contexto, para llevar a cabo la predicción.

En cuanto a la predicción univariable, hay muchos métodos para construir la función de predicción $\mathbf{f}^{(\mathbf{u})}(\{\mathrm{b(k)}, \mathrm{k} = 1, 2, 3, \dots, \mathrm{t}\})$, desde un predictor *Naive* muy sencillo y con apenas carga computacional (en [3] se le denomina «*simple history-repeat*»), que se rige por la idea de asignar a la predicción el valor anterior, hasta otros como *Exponential Weighted Moving Average* (EWMA):

$$\hat{b}(t+1) = (1-\alpha) \cdot \hat{b}(t+1) + \alpha \cdot b(t)$$

o *Harmonic Mean* (HM):

$$\hat{b}(t+\mathcal{C}) = \frac{h}{\sum_{k=0}^{h-1}\left(\frac{1}{b(t-k)}\right)}$$

que pueden limitar el impacto de los grandes *outliers* (valores atípicos, valores observados en un conjunto de datos que se diferencian significativamente del resto). En la sección 2.3.2 se comentan con más detalle.

En cuanto a la predicción multivariable, existen diversos algoritmos que hacen uso de *machine learning* como *Support Vector Machine* o *Random Forest*. En [9] y [29] se ha demostrado que las redes neuronales LSTM (véase la sección 2.4.2) pueden proporcionar una mayor precisión que los métodos de predicción de capacidad convencionales.

2.3.2. Técnicas clásicas (Naive, EWMA, Harmonic Mean)

Un predictor con mínima carga computacional es aquel que predice que la capacidad del enlace en el siguiente instante de tiempo es la misma que en el instante de tiempo actual.

Más formalmente, sea *t* el instante de tiempo actual, instante en el cual se conoce la capacidad disponible. El objetivo es predecir la capacidad disponible en el instante de tiempo t+$\mathcal{C}$. El predictor *Naive* (o, en la terminología de [3], un simple predictor *history-repeat*) establece que la capacidad predicha en el instante t+$\mathcal{C}$ es:

$$\hat{b}(t+\mathcal{C}) = b(t)$$

Entre los algoritmos clásicos de predicción de series temporales, se encuentran EWMA (*Exponentially Weighted Moving Average*, Media Móvil Exponencialmente Ponderada), y *Harmonic Mean*.

EWMA [30] [31] es una medida estadística cuantitativa usada para modelar o describir series temporales. Su filosofía se basa en asignar menos peso a las observaciones más antiguas. Los pesos caen exponencialmente a medida que los datos son más antiguos, de ahí su adjetivo «exponencialmente ponderada». La única decisión que se debe tomar en su diseño es el parámetro α. Este parámetro determina cuánta importancia tiene la observación más reciente en el cálculo de la EWMA. A mayor valor de α, más cerca estará la predicción EWMA de la serie temporal original. Su formulación matemática es simple:

$$x_{EWMA}(t) = \alpha * x(t) + (1-\alpha) * x_{EWMA}(t-1)$$

Siendo α: el peso elegido, x(t): el valor de la serie en el instante *t*, y $x_{EWMA}(t)$ la predicción obtenida mediante EWMA en el instante *t*. Se trata de una función recursiva, es decir, la observación actual se calcula usando las previas. Por esta razón, es demostrable [31] que el decaimiento de los pesos es exponencial. Si se desarrollase la

expresión hasta el término de índice 0 (inicial), se demostraría que el peso de la observación *x(t-k)* es:

$$\alpha * (1 - \alpha)^k$$

Nótese que, como α está definido entre 0 y 1, el peso se hace más pequeño a medida que *k* aumenta, es decir, se asigna un menor peso a las observaciones más antiguas.

Por otro lado, la Media Armónica (*Harmonic Mean*) [32] es un tipo especial de media que se calcula dividiendo el número de valores que hay en la serie temporal entre la suma de los recíprocos para cada valor en la serie temporal. Conforma una de las tres medias Pitagóricas, junto con la media aritmética y la geométrica. La media armónica siempre muestra el menor valor de estas tres medias. A continuación, se muestra su expresión matemática.

$$x_{Harmonic\ Mean} = \frac{n}{\sum_{i=0}^{n-1} x(i)}$$

Siendo *n* el número de valores del *dataset*, y *x(i)* un punto del *dataset*. También se puede calcular de manera eficiente manteniendo el valor del denominador calculado hasta un instante determinado.

2.4. Aprendizaje automático

2.4.1. Introducción al aprendizaje automático

Arthur L. Samuel, pionero en el campo de la inteligencia artificial y considerado como el creador del término *Machine Learning* [28], definió el aprendizaje automático (*Machine Learning*) el «campo de estudio que da a los ordenadores la capacidad de aprender sin ser explícitamente programados para ello» [33]. Por otro lado, de manera más formal, Tom Mitchell [34] afirmó que «un programa de ordenador aprende de la experiencia *E* respecto a una tarea *T* con una medida del rendimiento *P*, si su rendimiento en la tarea *T* medido por *P* mejora con *E*». En definitiva, el aprendizaje automático se trata de una serie de algoritmos capaces de extraer modelos partiendo de datos u observaciones de un sistema de cualquier tipo. El objetivo de estos algoritmos es llevar a cabo tareas de clasificación o regresión («estimación», si es de valores presentes, o bien «predicción» si es de valores futuros). Son muchas las aplicaciones que tiene actualmente el aprendizaje automático, en ámbitos de la robótica, la minería de datos, el reconocimiento de voz o escritura, entre otros muchos. Debido al crecimiento de la cantidad de datos disponible en los últimos años y de la capacidad de cómputo, el aprendizaje automático ha tomado especial relevancia [35]. A principios de los 2000, la cantidad total de datos disponibles en la Web rondaba entre los 25 y 50 terabytes [36]. En 2005, el tamaño aproximado era de 600 terabytes. Actualmente la cantidad total de datos disponibles es del orden de zetabytes [37].

Como primera aproximación, se puede comentar una primera distinción entre los distintos tipos de aprendizaje automático que existen [28]. Por un lado, estarían aquellos que se ejecutan sobre un conjunto de datos fijo, que se conocen como métodos de aprendizaje en lote (*batch*), y por otro lado estarían los que se ejecutan sobre un flujo de

datos (*stream* u *online*). El aprendizaje en lote se emplea para aprender modelos sobre un conjunto de datos fijo que se ha recolectado a lo largo de un período de tiempo. Para algunas aplicaciones este enfoque puede ser más que suficiente, pero para otras no. Por un lado, los datos pueden llegar a una velocidad suficientemente elevada como para que el hecho de almacenarlos conlleve un gasto de almacenamiento no asumible. Además, actualizar los modelos periódicamente puede llegar a ser demasiado costoso en términos computacionales, por lo que una buena solución sería la alternativa en tiempo real, en la que los datos no se almacenan y el modelo se va actualizando progresivamente a medida que se obtienen nuevos datos. Para el caso concreto de la predicción de la capacidad en redes móviles, predecir sobre un conjunto de datos fijo permitiría tener valores de predicción generalizados para ciertas condiciones de contexto dadas. Sin embargo, si se van tomando datos en tiempo real y actualizando el modelo a medida que se obtienen nuevos datos, podría responder mejor a imprevistos que no se hayan dado en el conjunto de datos fijo.

Otra clasificación pasa por distinguir aprendizaje supervisado y no supervisado [38]. El aprendizaje supervisado es el tipo de aprendizaje automático más común, en el cual se le entregan datos correctos de etiquetas al algoritmo, es decir, se conoce la correspondencia entre entrada y salida, y el algoritmo tiene que ser capaz de realizar predicciones cuando se le alimenta con nuevos datos. Requiere datos de entrada y de salida etiquetados durante la fase de entrenamiento del ciclo de vida del *machine learning*. Se llama «supervisado» porque el aprendizaje está guiado por las etiquetas, y esto es como si un humano supervisase. Con el aprendizaje de los patrones existentes entre los datos de entrada y salida, los modelos de aprendizaje supervisado son capaces de predecir resultados a partir de datos nuevos. Esto es lo que se conoce como «generalización». Por otro lado, el aprendizaje no supervisado consiste en el entrenamiento de modelos de datos sin procesar y sin etiquetar. El algoritmo aprende de manera autónoma para descubrir estructuras, patrones o agrupamientos en los datos, sin recibir información sobre etiquetas correctas. Por tanto, es el algoritmo quien tiene que ser capaz de encontrar estructura en los datos que se le entregan, formando distintos grupos.

Por último, cabe destacar la posibilidad de ejecutar muchos de estos algoritmos de manera distribuida. Tradicionalmente, el cuello de botella venía dado por la limitación de los datos disponibles [35], impidiendo desarrollar sistemas más inteligentes. Sin embargo, hoy en día, el factor limitante es la imposibilidad que tienen los algoritmos de aprendizaje de usar todos los datos para aprender en un intervalo de tiempo razonable. En este contexto, el aprendizaje distribuido apunta a ser una prometedora línea de investigación. Si se distribuye el proceso de aprendizaje a lo largo de varias unidades de computación, será más fácil el escalado de los algoritmos.

2.4.2. Técnicas de aprendizaje automático basadas en redes neuronales

Las redes neuronales (RNN) conforman un conjunto de métodos de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera que está inspirada en la forma en que lo hace el cerebro humano. Las redes neuronales surgen en los años 60, y el algoritmo de «propagación hacia atrás» (*back propagation*) en los años 80. El concepto de «aprendizaje profundo» (*Deep Learning*) surge en los años 80 para hablar de

arquitecturas con bastantes capas de neuronas. No fue hasta los años 2000 cuando se populariza este concepto, con la proliferación de las GPUs que permitieron paralelizar los cómputos necesarios para entrenarlas. Utiliza los nodos o las neuronas interconectados en una estructura de capas similar a la estructura del cerebro humano. Crea un sistema adaptable utilizado por las computadoras para aprender de los errores y mejorar continuamente. Así, las redes neuronales intentan resolver problemas complicados [39].

Una red neuronal artificial está formada por neuronas artificiales que trabajan juntas para resolver un problema. Las neuronas artificiales son módulos de *software* llamados nodos [39].

Una red neuronal básica tiene neuronas artificiales interconectadas en tres capas [39]:

- **Capa de entrada**. Punto de entrada de la información exterior a la red neuronal. Los nodos de entrada procesan los datos, los analizan o los clasifican, y los pasan a la siguiente capa.
- **Capa oculta**. Toma su entrada de la capa de entrada o de otras capas ocultas. Una red neuronal puede poseer una gran cantidad de capas ocultas, cada una de las cuales analiza la salida de la capa anterior, la procesa aún más, y la pasa a la siguiente capa.
- **Capa de salida**. Proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal artificial. Puede tener uno o varios nodos.

Las redes neuronales profundas (redes de aprendizaje profundo), sin embargo, tienen varias capas ocultas con millones de neuronas artificiales conectadas entre sí. Las conexiones entre un nodo y otro vienen representadas por un número denominado «peso». Este peso es un número positivo o negativo dependiendo de si un nodo estimula o suprime a otro. Los nodos con mayor peso tienen mayor influencia en los demás nodos. En teoría, las redes neuronales profundas podrían asignar cualquier tipo de entrada a cualquier tipo de salida, aunque, sin embargo, necesitarían mucho más entrenamiento en comparación con otros métodos. [39]

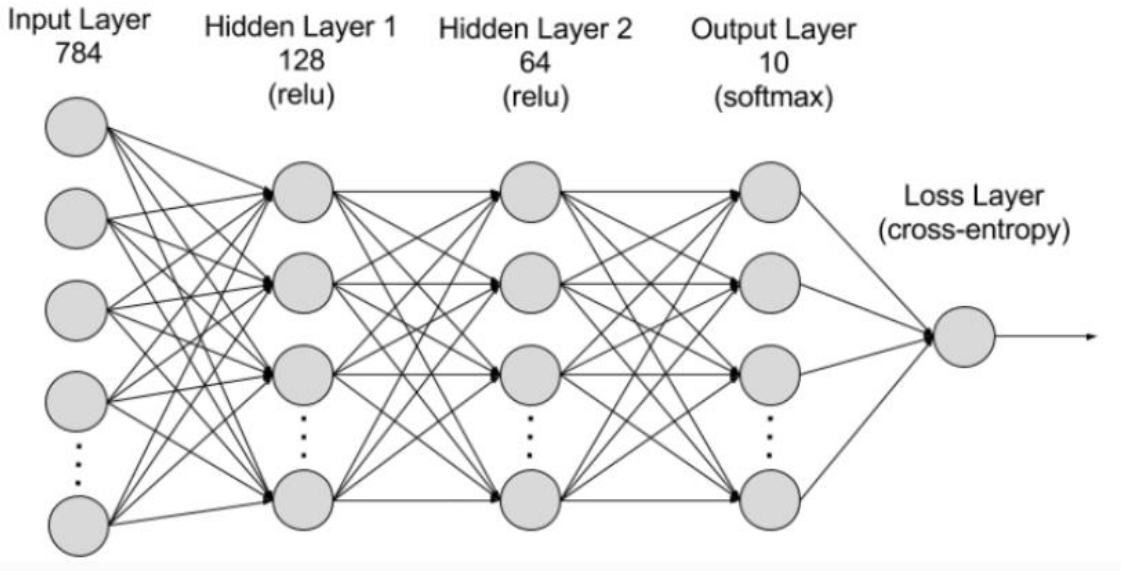

Figura 2.9. Esquema simple de la arquitectura de una red neuronal profunda. Figura tomada de [39].

Los modelos clásicos de redes neuronales tienen un problema, y es que su salida en un momento dado depende únicamente de la entrada presentada en ese momento. Sin embargo, los humanos no empiezan a pensar desde cero cada segundo. En una conversación cualquiera, cada persona comprenderá cada palabra según su comprensión de las palabras anteriores: no «resetea» su mente y empieza a pensar desde cero otra vez, sino que sus pensamientos tienen persistencia [40].

Como solución, surgieron las redes neuronales recurrentes. Se trata de redes con bucles que permiten que la información persista.

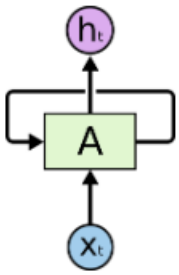

Figura 2.10. Las redes neuronales recurrentes tienen bucles. Figura tomada de [40].

En la Figura 2.10 se muestra un fragmento de red neuronal, ***A***, mirando a una entrada $\boldsymbol{X_t}$, que genera una salida $\boldsymbol{h_t}$. Un bucle permite pasar información de un paso de la red al siguiente. Estos bucles son los que marcan la diferencia fundamental entre las redes neuronales tradicionales y las recurrentes. Se puede descomponer el bucle (véase la Figura 2.11) para apreciar la naturaleza en forma de cadena que revela que las redes neuronales recurrentes están íntimamente relacionadas con secuencias y listas [40].

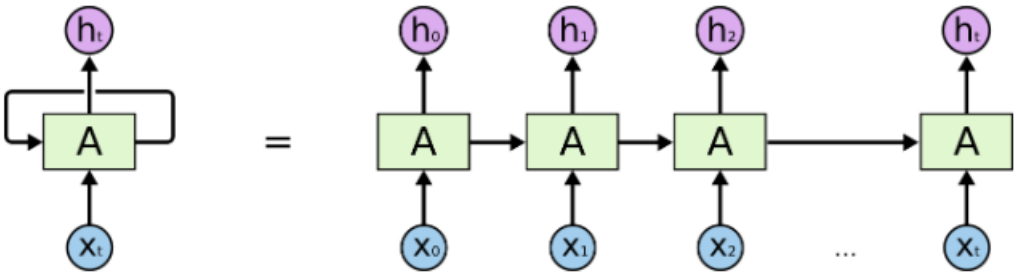

Figura 2.11. Una red neuronal recurrente desarrollada. Figura tomada de [40].

Uno de los grandes atractivos de las RNN es la idea de conexión de la información anterior con la tarea actual. En muchas ocasiones basta con conocer información reciente para poder realizar la tarea actual. Si la separación entre la información relevante y el lugar donde se necesita es pequeña, los RNN pueden funcionar adecuadamente. Sin embargo, otros casos requieren de más contexto. A medida que esta separación se hace mayor, los RNN se vuelven incapaces de realizar una correcta predicción.

Los LSTM (*Long Short Term Memory*) son un tipo muy especial de red neuronal recurrente que carecen del problema previamente mencionado. Como su nombre indica en su traducción al castellano, redes de memoria a largo plazo, son un tipo especial de RNN capaz de aprender dependencias a largo plazo. Están diseñados explícitamente para

evitar el problema de dependencia a largo plazo: su comportamiento predeterminado es, precisamente, recordar información durante largos períodos de tiempo.

Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos de red neuronal repetidos. En las RNN estándar, este módulo tiene una estructura muy simple, mientras que, en los LSTM, el módulo repetido tiene una estructura algo diferente: en lugar de tener una única capa de red neuronal, tienen cuatro capas que interactúan de una forma un tanto especial (véase la Figura 2.12). Con respecto a la notación presentada en dicha figura, cada línea lleva un vector completo (de la salida de un nodo a la entrada de otros). Los círculos de color rosa representan operaciones puntuales. Los cuadros amarillos son capas de redes neuronales aprendidas. Las líneas que se fusionan denotan «concatenación», mientras que las líneas que se bifurcan indican que su contenido se copia, y las copias van a diferentes ubicaciones [40]. La Figura 2.13 resume la notación que se acaba de comentar.

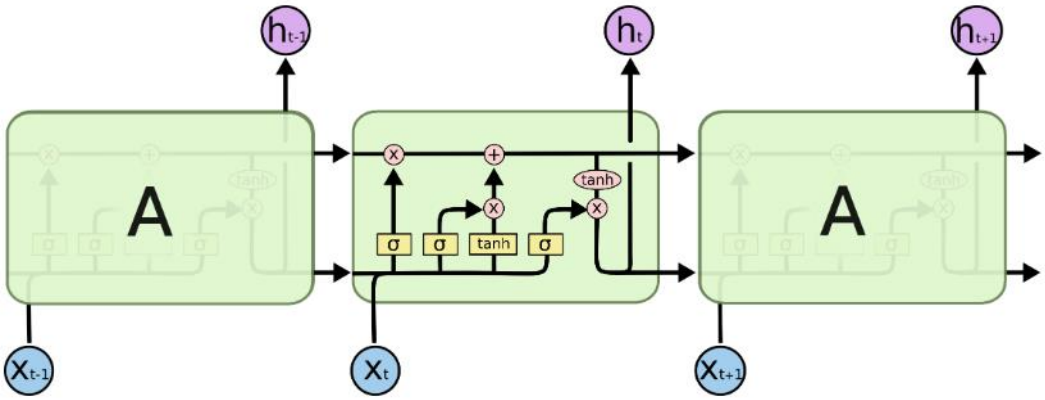

Figura 2.12. Detalle del módulo repetido en un LSTM. Figura tomada de [40].

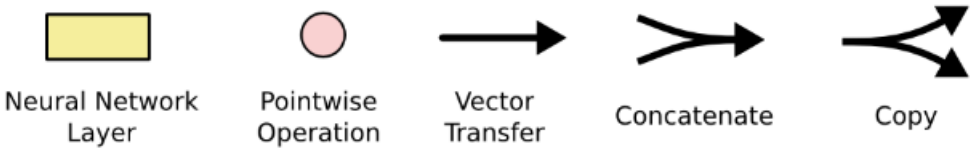

Figura 2.13. Resumen de la notación usada en LSTM. Figura tomada de [40].

La clave en los LSTM es el estado de la celda: es como una «cinta transportadora» que recorre toda la cadena con algunas interacciones lineales menores, por lo que es muy fácil que la información fluya sin cambios. LSTM tiene la capacidad de eliminar o agregar información al estado de la célula, regulado mediante unas estructuras conocidas como «puertas». Las puertas están compuestas por una capa de red neuronal sigmoidea y una operación de multiplicación puntual (véase la Figura 2.14). La capa sigmoidea genera números entre 0 y 1 que indican cuánto de cada componente se debe dejar pasar. Cada celda LSTM tiene 3 puertas.

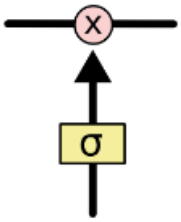

Figura 2.14. Puerta en LSTM. Figura tomada de [40].

El primer paso en LSTM es decidir qué información se va a descartar del estado de la celda, decisión que se toma en una capa sigmoidea conocida como «capa de puerta hacia el olvido». Genera un número entre 0 y 1 para cada número en el estado de la celda, indicando deshacerse completamente de ello (0) o bien mantenerlo por completo (1). El siguiente paso es decidir qué nueva información se va a almacenar en el estado de la celda. Este paso consta de dos partes: una «capa de puerta de entrada» que decide qué valores se actualizarán, y una «capa *tanh*» que crea un vector de valores candidatos que podrían sumarse al estado. En el siguiente paso se combinarán ambas partes para crear una actualización del estado. A continuación, se actualiza el estado de la celda: se olvidan las cosas que se decidieron olvidar, y se agregan los nuevos valores candidatos, escalados según cuánto se haya decidido actualizar cada valor de estado. Finalmente, se decide qué se va a generar, basándose en el estado previo de la celda, pero será una versión filtrada. Primero se ejecuta una capa sigmoidea que decide qué partes del estado de la celda se van a generar, y luego se pasa el estado de la celda *tanh* y se multiplica por la salida de esta puerta sigmoidea, de modo que sólo se genere lo deseado [40].

No todas las variantes LSTM son exactamente iguales a la previamente descrita, pero las diferencias son menores.

2.4.3. Variantes de LSTM para la predicción de la capacidad de enlace

Algunas de las variantes de LSTM que se pueden encontrar en la literatura [10] son las que se mencionan a continuación.

EA-LSTM [41] (*Evolutionary Attention-based LSTM*, Atención Evolucionaria basada en LSTM) consiste en un mecanismo de atención combinado con LSTM, y los pesos de atención se actualizan utilizando búsqueda aleatoria para dar diferentes niveles de atención a las características del conjunto de datos. El mecanismo de atención introducido puede asignar cuantitativamente pesos de importancia para cada paso temporal específico en las características para mejorar las dispersiones típicas de los modelos LSTM tradicionales.

TG-LSTM [42] (*Transformation-Gated LSTM*, LSTM Controlado por Transformación) se basa en la estructura de LSTM, se diseña en la salida de la puerta de olvido una estructura de puerta de transformación, y así la salida de la puerta de entrada y la información del estado de la memoria del momento previo se procesan utilizando la función tangente hiperbólica. De esta manera, se consigue la habilidad de capturar la información de mutación a corto plazo. Durante la propagación hacia atrás, el cambio de gradiente queda completamente reflejado en el rango de valores de la función de la

derivada parcial correspondiente a la puerta de transformación. De esta forma, se obtiene un menor flujo de gradiente de error.

TPA-LSTM [10] (*Temporal Pattern Attention LSTM*, LSTM con Atención al Patrón Temporal) utiliza una serie de filtros para extraer patrones invariantes en el tiempo, y selecciona la serie temporal relevante utilizando un mecanismo de atención al patrón que selecciona las variables relevantes más que los pasos temporales relevantes para dar pesos, y usa su información en el dominio de la frecuencia para predicción multivariable.

2.5. Conjuntos de datos públicos para la predicción de capacidad de enlace

Diferentes autores han generado diversos conjuntos de datos de medidas de la capacidad en diferentes tipos de redes móviles. En [7], los autores han generado un conjunto de datos coleccionando trazas del mayor operador de Irlanda con dos patrones de movilidad: conducción y estático. Además, la estrategia de descarga no comprende solo descarga de ficheros, sino además visualización de vídeo en línea. Lo consiguen haciendo uso de *G-NetTrack Pro*, una herramienta de monitorización de red disponible para dispositivos Android que permite coleccionar muchas métricas relacionadas con el canal, contexto y capacidad (de subida y bajada). Utiliza la librería estándar de Android y no requiere de privilegios de superusuario, pero tiene algunas limitaciones como una granularidad mínima de un segundo en métricas de canal, y no tiene la misma capacidad de medir todas métricas en diferentes chipsets SoC (*System on Chip*, Sistema en Chip). Su *dataset*[4] contiene 83 trazas con una duración total de 3142 minutos. La estrategia seguida para su recolección fue ejecutar experimentos hasta agotar con el plan de datos mensual (80 GB) para cada combinación de aplicación (descarga de ficheros, Netflix o Amazon Prime) y patrón de movilidad (estático o movimiento). Para el escenario estático pudieron capturar 160 minutos (pues la descarga de ficheros produce altas tasas de transferencia, lo que agota rápidamente el plan de datos), pero se pueden dividir en trazas de menor duración dependiendo del escenario que se desee experimentar. Además de los valores de capacidad, contiene otros valores: almacenan el *timestamp*, latitud y longitud (*Latitude* y *Longitude*), velocidad (*Velocity*, en km/h), nombre del operador (*Operatorname*), id de celda (*cellID*), modo de red (*NetworkMode*), tasas de bajada y subida (*DL_bitrate* y *UL_bitrate*), estado del proceso de descarga (*state*), estadísticas de ping, y valores de RSRQ (*Reference Signal Receiving Quality*), SNR, RSRP (*Reference Signal Received Power*), RSRI, CQI y valores de sus vecinos NRxRSRQ y NRxRSRP.

En [10] también coleccionan un *dataset*[5] 5G en escenarios de transporte público (metro) y coche, y lo combinan con otros *datasets* disponibles públicamente. Los datos del metro se capturan en la línea 2 del metro de Ningbo desde la estación de Freng Yuang hasta la estación de Lulin. Tiene una duración total de 120 minutos, lo que resulta en un total de 1000 registros de traza, aproximadamente. Los datos registrados en coche comprenden la sección de Guanghua Road en la ciudad de Ningbo. En este caso, tienen una duración total de 150 minutos, lo que resulta en un total de 1100 registros de traza de

[4] Disponible en http://cs1dev.ucc.ie/misl/5Gframework/5G-production-dataset.zip
[5] Disponible en https://github.com/BKSN964/CapRadar-Dataset.git

capacidad. Utilizan su plataforma de colección de datos *NBCrowd*, que proporciona información de la estación base, del canal y del contexto, transfiriendo los datos en formato JSON a la base de datos. Para coleccionar la información de la capacidad, utilizaron *Iperf*. Contiene el *timestamp*, coordenadas GPS de localización (*Location*), capacidad de descarga (*Bandwidth*), número de celdas LTE a las que se puede conmutar (*CellNumber*), intensidad de señal recibida (*Level*), RSRP, RSRQ, número de canal de radiofrecuencia E-UTRA (*Earfcn*) y número de celda que sirve al dispositivo móvil (*Ci*). De todas estas, solo se seleccionan las más significativas usando ANOVA (*Multi-factor Analysis Of Variance*), pues demuestran obtener mejores resultados de predicción si se hace uso de las características más significativas únicamente.

En [9] utilizan, en primer lugar, un *dataset*[6]HSDPA de la Universidad de Oslo (Noruega) con trazas de capacidad en diferentes medios de transporte (tren, tranvía, ferry, coche, bus y metro), grabando la capacidad y localización cada 1000 ms, obteniendo trazas de 500 a 1000 s. También generaron su propio *dataset*[7] en diferentes rutas de autobús y metro de la ciudad de Nueva York, registrando con *iPerf* la tasa de transferencia efectiva TCP cada 1000 ms, generando trazas de duración 10000 a 20000 segundos.

En [3] generan un *dataset*[8] multivariable LTE centrándose en rutas de movilidad fijas (autobús y metro) en la ciudad de Nueva York. Lo coleccionan desde noviembre de 2019 hasta enero de 2020, haciendo unos 8 viajes de un punto a otro en ambos sentidos (cada viaje duraba unos 30 minutos o más), generando un *dataset* de aproximadamente 30 horas. Midieron la capacidad e información de canal y de contexto usando *NetMonitorPro*, y con *iPerf* descargaron los datos en su servidor. Registraron la capacidad de descarga junto con sus *timesteps* (para relacionar la capacidad con la información de contexto) cada 1000 ms. En la sección 3.2 se describe este *dataset* con mayor profundidad.

La Tabla 2.1 resume los rasgos de los principales conjuntos de datos públicos que se han descrito previamente.

[6] Disponible en http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/
[7] Disponible en https://github.com/NYU-METS/Main
[8] Disponible también en https://github.com/NYU-METS/Main

Ref. conj. datos	[7]	[10]	[9] **Reutilizado**[9]	[9] **Propio**[10]	[3]
Publicaciones conocidas que usan este conj. datos[11]**…**	[7]	[10]	[9]	[9]	[3] [10]
Patrón movilidad	Sí (conducción)	Sí: metro (120 min) y coche (150 min)	Sí: tren, tranvía, ferry, coche, bus, metro	Sí: autobús y metro	Sí: autobús y metro
Patrón estático	Sí (160 min)	No	No	No	No.
Estrategia descarga ficheros	Sí	¿?	¿?	¿?	¿?
Estrategia vídeo en línea	Sí	¿?	¿?	¿?	¿?
Nº de trazas	83	1000(metro) + 1100(coche)	¿?	¿?	¿?
Duración de cada traza	¿?	¿?	500 - 1000s	10000 - 20000s	10000 – 25000s
Duración total	3142 min	270 min	¿?	¿?	30 horas
Características registradas (aparte de la capacidad)	• *Timestamp* • Latitud y longitud • Velocidad • Nombre del Operador • Id de celda • Modo de red • Tasas bajada y subida • Estado del proceso de descarga • Estadísticas de ping • RSRQ • SNR • RSRP • RSRI • CQI NRxRSRQ • NRxRSRP	• *Timestamp* • Coordenadas GPS de localización • Capacidad de descarga • Nº de celdas LTE a las que se puede conmutar • Intensidad de señal recibida • RSRP • RSRQ • Número de canal de radiofrecuencia E-UTRA (*Earfcn*) • Número de celda que sirve al dispositivo móvil (*Ci*)	• Localización	(Univ.)	• Capacidad • Vecinos LTE • RSSI • RSRQ • Echng (Ech) • TA • Velocidad • Banda

Tabla 2.1. Resumen de los conjuntos de datos públicos.

2.6. Evaluación del rendimiento

Para evaluar el rendimiento de los diferentes predictores que se utilizan habitualmente en la predicción de series temporales y que también se han utilizado en

[9] Se refiere al conjunto de datos que no han elaborado sus autores.
[10] Se refiere al conjunto de datos que sí han elaborado sus autores.
[11] Referencia de artículos conocidos en los que se ha usado este conjunto de datos.

este proyecto, se han utilizado las métricas que a continuación se detallan, pues son las que se reportan en el artículo [3] cuyos resultados queremos evaluar.

2.6.1. RMSE

RMSE (*Root Mean Square Error*, Raíz del Error Cuadrático Medio) es la medida de la raíz cuadrada de los errores al cuadrado entre los valores predichos y los valores reales observados [43]. MSE (*Mean Square Error*, Error Cuadrático Medio) y RMSE penalizan más los errores cuyo valor absoluto es mayor, inflando el valor medio del error debido a ese cuadrado del valor del error. A continuación, se muestra la expresión matemática que permite calcularlo:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{x}(i) - x(i)\right)^2}$$

2.6.2. MAE

MAE (*Mean Absolute Error*, Error Absoluto Medio) [44] es una métrica popular de evaluación de rendimiento en la que las unidades del error coinciden con las unidades del valor predicho. A diferencia de RMSE, los cambios en MAE son lineales y, por tanto, más intuitivos. Con MAE los diferentes errores no se ponderan en mayor o menor medida, pero las medidas de error se incrementan linealmente con el incremento de los errores.

La medida del MAE se lleva a cabo promediando el valor absoluto de los errores. Dado que se toma el valor absoluto, la diferencia entre el valor predicho y el esperado (que puede ser positiva o negativa) será necesariamente positiva cuando se calcule el MAE.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|(\hat{x}(i) - x(i))|^2$$

2.7. Conclusiones

En este capítulo se ha visto una breve introducción a las redes móviles, desde su perspectiva histórica hasta detalles de la arquitectura de la red 4G. Se ha comentado el problema de la variabilidad en estas redes, así como trabajos de predicción que se pueden ver en la literatura. Se ha explicado que el problema de predicción de la capacidad se puede ver como un problema de predicción de series temporales, y se han comentado las principales técnicas clásicas de predicción (*Naive*, EWMA y *Harmonic Mean*). Se ha hecho una breve introducción al aprendizaje automático, así como a las técnicas de aprendizaje automático basadas en redes neuronales y diferentes variantes de LSTM utilizadas. Se han descrito los diferentes conjuntos de datos que se pueden encontrar en repositorios públicos. Por último, se han descrito las dos métricas (RMSE y MAE) que se van a utilizar para evaluar el rendimiento de las predicciones.

Capítulo 3

Reproducción de resultados

3.1. Introducción

Para ofrecer servicios de acceso a Internet a usuarios que viajan en transporte público con una calidad de servicio consistente, existe una fuerte necesidad de estimar la capacidad en tiempo real. En el caso de un usuario que está visualizando un vídeo en *streaming*, el reproductor de vídeo puede adaptar la calidad del vídeo a ser descargado basándose en la predicción de la capacidad del enlace de la red móvil. Otro posible escenario es el de un usuario en una videoconferencia: la aplicación de videoconferencia puede ajustar dinámicamente la resolución y la tasa de refresco del vídeo a ser codificado y enviado a otros usuarios, basándose para ello en las predicciones de la capacidad del enlace.

De entre todas las alternativas existentes (presentadas en el capítulo anterior), se ha optado por usar el conjunto de datos de [3]: llevaron a cabo una medición de la capacidad de la red móvil, coleccionando una serie de trazas en Nueva York. Dichas trazas cubren diferentes medios de transporte y diferentes rutas en diferentes instantes del día. En las siguientes secciones se describirán en detalle estas trazas.

Se desarrollarán modelos LSTM para la predicción en tiempo real de la capacidad en los siguientes segundos, basándose en las variantes encontradas en la literatura. No obstante, sin olvidar que LSTM no es el único método de predicción (ni el más sencillo computacionalmente), se harán diversas pruebas con otros predictores que, sin llegar a ser tan sofisticados como el propio LSTM, pueden llegar a ofrecer unos resultados que sean aceptables y permitan ahorrar una gran carga computacional o actualizarse a medida que llegan nuevos datos (entrenamiento en línea versus fuera de línea). No se debe olvidar que existe un compromiso entre carga computacional y exactitud de la predicción. Un empeoramiento pequeño en la precisión con la que se hace una predicción puede tener un impacto despreciable en las implicaciones prácticas, y puede ser preferible una solución más sencilla que consuma menos recursos, especialmente en dispositivos móviles. En el caso del entrenamiento fuera de línea frente al entrenamiento en línea, el modelo puede quedar completamente desfasado con el paso del tiempo, lo cual supone un problema.

El principal esfuerzo consistirá en replicar los resultados de [3]. Se intentará seguir, en la medida de lo posible, las indicaciones que en dicha referencia se dan acerca del entorno de ejecución de los experimentos, parámetros utilizados para el entrenamiento y

evaluación de modelos, criterios de parada del entrenamiento… entre otros. Si no se logra replicar los resultados, se intentará averiguar las posibles causas. Una vez obtenidos los resultados, se llevará a cabo un análisis de los mismos.

El presente capítulo muestra los experimentos que se han llevado a cabo como parte de este Trabajo de Fin de Grado. En primer lugar, se describe con detalle el conjunto de datos que se ha utilizado. En las siguientes subsecciones se describe el *software* y *hardware* empleados. A continuación, se muestran los detalles de la implementación de cada uno de los predictores con los que se ha trabajado, así como la división del *dataset* en los diferentes subconjuntos de entrenamiento, validación y test. Por último, se muestran todos los resultados y las discusiones correspondientes.

3.2. Conjunto de datos utilizado

Se ha optado por utilizar el conjunto de datos de [3], pues es un conjunto de datos disponible en un repositorio público que, además de en [3], se ha utilizado en otros artículos, como en [10], y es con el que mejores resultados se han reportado. Además, es un conjunto de datos multivariable, pues han registrado otras características aparte de la capacidad, lo que, según los artículos, permite mejorar la calidad de las predicciones.

Se pondrá el foco de atención en escenarios de movilidad de ruta fija [3]. Las rutas que serán analizadas son rutas diarias en transporte público. A lo largo del análisis se determinará cuáles son las características que más relevancia tienen en la determinación de la capacidad futura.

La propuesta de [3] es reutilizar *software* disponible y el *hardware* de los propios usuarios, es decir, sus propios teléfonos móviles. En la práctica, estas medidas se pueden tomar gracias a equipos de optimización de red, a las contribuciones de un gran grupo de personas o incluso pidiéndoselo directamente a las compañías de transporte. Los modelos de predicción *offline* se pueden ejecutar en tiempo real en local en los teléfonos móviles para llevar a cabo la predicción (no para la construcción del modelo).

Fijado el objetivo, en [3] realizan un estudio de medición en el sistema de transporte público de Nueva York. A diferencia de otros conjuntos de datos de la capacidad de la red LTE, han generado un **conjunto de datos multivariable**[12] y se han centrado en rutas de transporte público fijas. Han tomado cinco rutas de autobús o metro de NYC, tal y como se ilustra en la Figura 3.1.

Figura 3.1. Rutas del transporte público de la ciudad de Nueva York donde se han realizado mediciones de la capacidad LTE. Figura tomada de [3].

[12] Disponible en https://github.com/NYU-METS/Main

Las medidas presentes en el dataset se tomaron desde noviembre de 2019 hasta finales de enero de 2020. Para cada ruta se tomaron largas e ininterrumpidas trazas haciendo en torno a unos ocho viajes de un extremo a otro, en ambos sentidos (ida y vuelta), con una duración de viaje superior a 30 minutos. En su conjunto, el conjunto de datos coleccionado abarca aproximadamente 30 horas. Es interesante resaltar que la información disponible en el conjunto de datos no permite determinar qué datos corresponden a cada uno de los viajes, ya que todos los viajes están concatenados, por lo que no podemos saber cuándo termina el viaje de ida y comienza el de vuelta para poder razonar qué partes de cada viaje son las más «difíciles» de predecir.

Aparte de medir la capacidad, objetivo principal, se midió información relativa de canal y contexto utilizando *NetMonitorPro*[13], una aplicación de monitorización de redes diseñada para dispositivos con sistema operativo Android. En concreto, para la obtención del conjunto de datos, se empleó un dispositivo móvil Google Pixel 1 con un plan de datos 4G/LTE ilimitado. Para registrar los datos relativos a la capacidad, se utilizó *iPerf*[14], una herramienta muy utilizada para llevar a cabo diversas pruebas en redes. Concretamente, sirvió para descargar datos desde el servidor de laboratorio situado en el campus NYC, y registrar la tasa de transferencia efectiva TCP cada 1000 milisegundos. Nótese que la capacidad se almacenaba con marcas de tiempo, útiles para relacionar la información de canal y contexto obtenida con *NetMonitorPro*.

Se registraron, además de la capacidad, otras 52 características del canal y de contexto [3]. De estas características solo seleccionaron algunas, teniendo en cuenta la correlación cruzada entre cada característica y la capacidad. Como resultado de esta determinación, proponen usar solo ocho características como entrada a los modelos de predicción. Dichas características se resumen en la tabla Tabla 3.1.

Como era de esperar, la capacidad en el instante anterior tiene la mayor importancia en la predicción del instante actual: el valor del rendimiento precedente tiene el mayor peso en la predicción del valor de rendimiento siguiente [3]. Le sigue la Banda de la señal, ya que normalmente las transmisiones de datos de alta velocidad sobre LTE se suelen realizar en bandas de frecuencia como la 1900 y 2100, mientras que las transmisiones de baja velocidad se suelen dar en otras como la banda 700. La capacidad tiende a ser alta cuando se transmite en una banda «ideal», y se convierte en baja cuando se pasa a transmitir en una banda «no ideal». La siguiente característica más importante es el RSSI que, como su nombre indica, se trata de un indicador de la fortaleza de la potencia de señal recibida desde la estación base. Nótese que a la característica RSRQ (*Reference Signal Received Quality*) no se le ha asignado tanta importancia en la predicción [3] ya que se deriva del RSSI, pero eso no quiere decir que sea menos importante, simplemente que para la predicción van a dominar las otras características (Capacidad, Banda y RSSI) cuando se usen todas en su conjunto. Así, la característica *Echng* indica si ha ocurrido traspaso (*handoff*) del *Evolved Node B* (ENodeB, entidad de red que se comunica directamente con los dispositivos móviles y gestiona la comunicación de los usuarios con la red central [17]). Cuando ocurre *handoff* o traspaso, hay un pequeño período de tiempo en el que el dispositivo no recibe ningún servicio de ningún ENodeB y la capacidad cae

[13] *NetMonitorPro* está disponible en la tienda de aplicaciones Play Store de Google, en Android, en el siguiente enlace: https://play.google.com/store/apps/details?id=ru.v_a_v.netmonitorpro&hl=es&gl=US

[14] Web oficial de *iPerf*: https://iperf.fr/

a cero. Para detectarlo, puede resultar útil el identificador del ENodeB y, si cambia, es porque ha ocurrido *handoff*. También es importante la característica de velocidad, pues la calidad de la señal y la frecuencia de *handoff* dependen fuertemente de la velocidad a la que se mueve el dispositivo.

Característica	**Información capturada**
Capacidad – *Bandwidth* (BW)	Capacidad de descarga, expresada en Mbps.
Vecinos LTE	Número de celdas LTE a las que el dispositivo puede conmutar.
RSSI	Nivel de potencia de la señal recibida.
RSRQ	Indicador de calidad de la señal recibida.
Echng (Ech)	Indica si el ENodeB ha cambiado en el segundo previo.
TA	Indica el avance de tiempo necesario para alcanzar el ENodeB.
Velocidad	Velocidad de movimiento del dispositivo.
Banda	Banda de frecuencia de la señal.

Tabla 3.1. Características seleccionadas.

En la Tabla 3.2 se muestran las estadísticas de las trazas que se van a utilizar. Se puede observar la diferente variabilidad que existe entre las diferentes líneas. La media de capacidad disponible es aproximadamente el doble en el *Bus M15* que en el *7 Train*. Llama la atención el hecho de que la desviación estándar de la línea *7 Train* es muy próxima a la media, lo que sugiere que las variaciones en esa línea van a ser muy acusadas.

	7 Train	**Bus B16**	**Bus B61**	**Bus B62**	**Bus M15**
Media	8.68	13.71	17.80	18.35	20.63
Mediana	6.84	12.8	16.0	15.8	19.1
Máx.	37.4	45.0	47.4	50.4	44.3
Mín.	0.0	0.0	0.0	0.0	0.0
Desv. Std.	8.29	9.55	11.69	12.16	9.53
Long. (s)	15122	22282	21179	22005	23108

Tabla 3.2. Estadísticas de las trazas del sistema de transporte público de NYC (Mbps).

A continuación se muestra una tabla comparativa de nuestras estadísticas con las calculadas en [3], donde se han remarcado en negrita aquellos valores que difieren.

	7 Train	Bus B16	Bus B61	Bus B62	Bus M15
Media	**8.68**	13.71	17.80	**18.35**	20.63
Media [3]	**8.67**	13.71	17.80	**18.34**	20.63
Mediana	**6.84**	12.8	16.0	15.8	19.1
Mediana [3]	**6.85**	12.80	16.00	15.80	19.10
Máx.	37.4	45.0	47.4	50.4	44.3
Máx. [3]	37.40	45.00	47.40	50.40	44.30
Mín.	0.0	0.0	0.0	0.0	0.0
Mín. [3]	0	0	0	0	0
Desv. Std.	8.29	9.55	11.69	12.16	9.53
Desv. Std. [3]	8.29	9.55	11.69	12.16	9.53
Long. (s)	**15122**	**22282**	**21179**	**22005**	**23108**
Long. (s) [3]	**15116**	**22277**	**21174**	**22000**	**23103**

Tabla 3.3. Comparativa de nuestras estadísticas con las estadísticas de [3], estadísticas de las trazas del sistema de transporte público de NYC (Mbps).

Como se puede ver en la anterior Tabla 3.3, los valores de media, mediana, valores máximos y mínimos, y desviación estándar son prácticamente los mismos. La diferencia quizá algo más significativa se encuentra en la longitud de las trazas. Teniendo en cuenta que se toma una medida cada 1000 milisegundos, la longitud de las trazas coincide con la duración de la traza en segundos. La duración de la traza de [3] es menor en 5 segundos (6 segundos en el caso de la traza de *7 Train*). Puede deberse a que los autores han restado el tamaño de la ventana. Dadas las escasas diferencias, hemos confiado que se trata del mismo conjunto de datos.

A continuación, se muestran las trazas de capacidad del conjunto de datos previamente descrito y que se van a utilizar. Es llamativo el hecho de que las caídas de la capacidad disponible a un valor nulo son mucho más frecuentes en la línea del *7 Train* (véase la Figura 3.4). El resto también tiene caídas a cero, pero, a simple vista, no parece que lo hagan con tanta frecuencia.

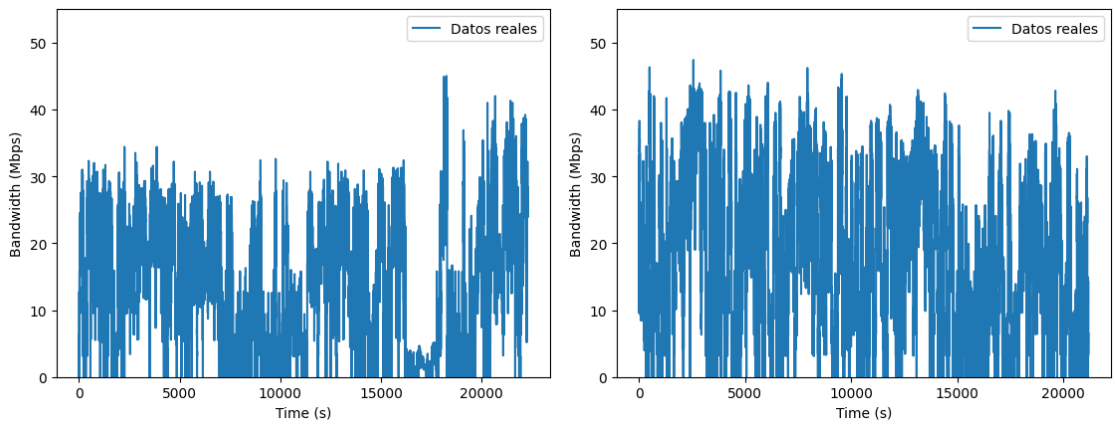

Figura 3.2. Traza del Bus B16 (izquierda) y del Bus B61 (derecha).

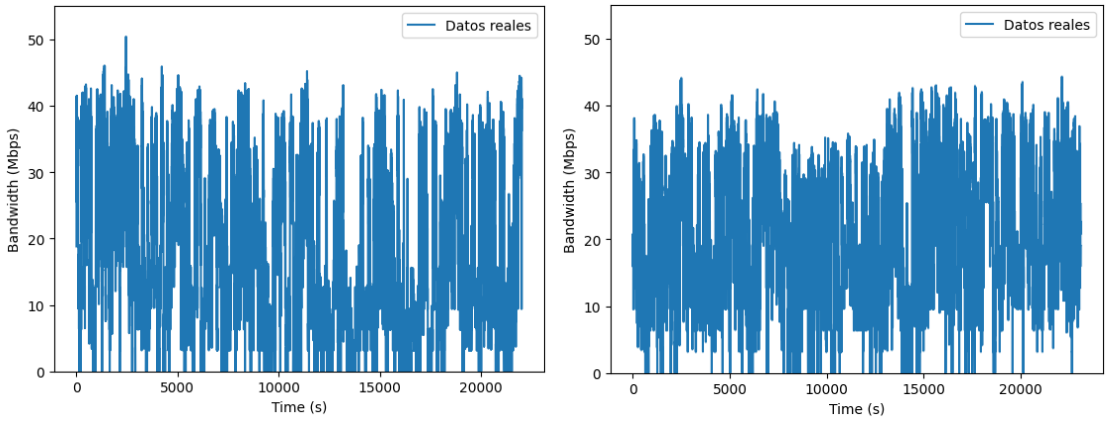

Figura 3.3. Traza del Bus B62 (izquierda) y Bus M15 (derecha).

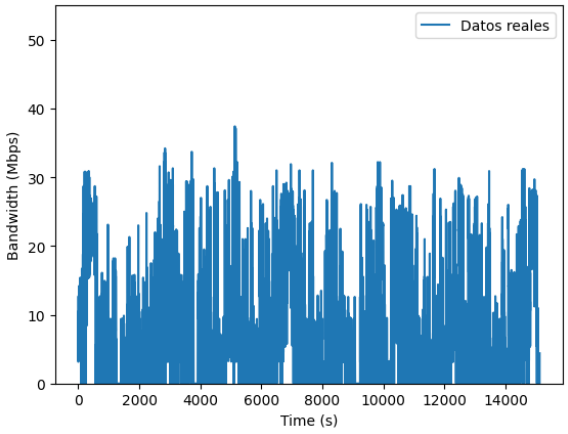

Figura 3.4. Traza del tren Subway 7 Train.

Con estas trazas, se llevará a cabo todo el proceso de entrenamiento y evaluación de los modelos de predicción propuestos.

3.3. Variante de LSTM utilizada

Se ha optado por implementar un modelo LSTM (y no otras variantes como EA-LSTM) por ser la variante más sencilla y necesaria para poder entender el resto de sus variantes, además de ser el modelo de predicción que se toma como base para dar los primeros resultados del rendimiento de predicción en casi todos los artículos, concretamente en [3], [9] y [10] se usa.

Dado que los detalles de implementación del modelo LSTM utilizado en [3] no estaban completos (y no dejaban claro si algunos parámetros mencionados se referían al modelo LSTM base o a alguna de sus variantes), ni se proporcionaba la implementación, se contactó con sus autores para resolver las incógnitas, pero no recibimos respuesta. Esto

nos ha llevado a desarrollar el modelo de la manera más ajustada posible a lo que estaba descrito en los artículos.

3.4. Entorno de desarrollo

Se desarrollarán modelos de predicción de la capacidad en tiempo real basados en aprendizaje profundo (DL, *Deep Learning*), intentando predecir la capacidad de unos pocos próximos instantes de tiempo, basándose en medidas de la capacidad en instantes anteriores, así como información de canal y de contexto. En concreto, se implementará la versión de LSTM que describen en [3].

En resumen, el objetivo será construir modelos LSTM para la predicción de la capacidad disponible a 1-3 segundos vista, utilizando trazas disponibles en repositorios públicos, reproduciendo los resultados de [3], y comparar estos modelos con otros de menor coste computacional.

Para implementar LSTM, se ha utilizado el lenguaje de programación Python y sus paquetes especializados para aprendizaje automático y profundo haciendo uso de la GPU. A continuación, se comentarán algunos detalles del entorno de desarrollo empleado, desde el lenguaje de programación hasta el entorno de desarrollo utilizado.

3.4.1. Python

Se ha utilizado el lenguaje de programación Python, tanto para el procesamiento de los datos como para el uso de técnicas de aprendizaje automático. Python es un lenguaje de programación interpretado e interactivo que sigue el paradigma de orientación a objetos. Incorpora un gran número de módulos, excepciones, clases, datos dinámicos de alto nivel, etc. Es un lenguaje ampliamente utilizado en librerías de aprendizaje profundo como *PyTorch*[15] (usado en profundidad para la construcción del modelo LSTM).

Pese a ser por naturaleza un lenguaje de programación orientado a objetos, también soporta otros tipos de paradigmas de programación como la programación imperativa o funcional. En resumidas cuentas, Python se caracteriza por combinar una notable potencia con una sintaxis muy clara. Tiene una licencia de código abierto, y se puede ejecutar en variantes de *Unix* (como *Linux*) y en *Windows*.

Otra de las grandes virtudes de este lenguaje es la infinidad de bibliotecas de las que consta, como *NumPy*, *SciPy*, *Pandas*, o la ya mencionada *PyTorch* algunas de las cuales facilitan enormemente el trabajo en el ámbito de la computación científica y numérica.

[15] https://pytorch.org/

3.4.2. Anaconda

Anaconda[16] es una plataforma de distribución de Python de código abierto ampliamente utilizada en el trabajo con datos o aprendizaje automático. En su repositorio en la nube se encuentran más de 7000 paquetes Python, los cuales se pueden instalar mediante el comando `conda-install`.

Se basa en los entornos (*environments*) que pueden ser mantenidos, actualizados y ejecutados de forma separada sin interferencia entre ellos [45]. También permite duplicar entornos de forma cómoda. Se puede utilizar mediante su interfaz gráfica o mediante línea de comandos.

3.4.3. Jupyter Notebook

Integrado en *Anaconda*, se encuentra *Jupyter Notebook*, que, como bien define en [46], es una «herramienta imprescindible para la ciencia de datos».

Jupyter Notebook es una herramienta de código abierto que permite desarrollar y compartir código en diferentes lenguajes de programación (Python, R, Julia…). Integra el código, texto, gráficos y demás elementos multimedia en un documento interactivo. El código se escribe y ejecuta en celdas individuales, lo que permite explorar y analizar los datos de manera interactiva. Además del propio código, se puede agregar texto explicativo, ecuaciones matemáticas, imágenes y visualizaciones. En definitiva, *Jupyter Notebook* es una herramienta muy flexible que se adapta a diversos entornos desde el aprendizaje automático (caso que nos ocupa) hasta la enseñanza y la investigación.

Numerosas son las ventajas de utilizar *Jupyter Notebook*: una interfaz fácil de usar, flexibilidad (al ser posible utilizar distintos lenguajes de programación), colaboración entre varios usuarios que trabajan en el mismo proyecto (facilitándoles colaborar y compartir resultados), y la visualización, dada la capacidad de integración de gráficos y visualizaciones interactivas.

Se trata de una aplicación web que permite la creación y el intercambio de documentos que incluyen, como bien se ha comentado antes, código, ecuaciones, visualizaciones y texto explicativo. Los documentos (llamados *notebooks*) se organizan en celdas que pueden ser de diferente tipo (código, texto, *markdown*, ecuaciones…). Cuando se ejecuta una celda de código, se muestra el resultado de la operación directamente debajo de la celda. Esto permite una interacción rápida y dinámica con el código y los datos, lo que es especialmente útil en la ciencia de datos. También cuenta con una amplia variedad de herramientas y funciones que facilitan el trabajo con datos y código: es posible importar bibliotecas de Python, por ejemplo, para trabajar con datos o visualizarlos en gráficos interactivos.

[16] https://www.anaconda.com/

3.4.4. PyCharm

PyCharm[17] es un IDE de Python pensado para la ciencia de datos y el desarrollo web. Permite empezar a programar directamente sin tener que instalar y configurar complementos: tiene todas las herramientas de Python necesarias para trabajar con datos. Permite organizar todos los proyectos de una forma efectiva, y facilita la tarea de ejecución remota, simplemente especificando en la correspondiente opción del proyecto que se desea utilizar un servidor remoto mediante conexión SSH así como el entorno Conda remoto con el que se desea trabajar. Permite escribir y depurar código fácilmente, y muestra sugerencias inteligentes que ayudan a evitar errores de sintaxis. Es compatible con diversos sistemas operativos como Windows, Linux y MacOS.

Para la construcción del modelo LSTM se ha preferido utilizar Jupyter Notebook por la capacidad de ejecución celda a celda y preferencia personal de tener todo el código dentro de un fichero de cuaderno Python. Sin embargo, para la ejecución del modelo de EA-LSTM ya implementado[18] se ha optado por utilizar un entorno integrado de desarrollo clásico como es PyCharm, pues en el repositorio se encontraba un proyecto completo de IDE.

3.5. Hardware empleado

El *software* previamente mencionado no sería nada sin un *hardware* en el que ejecutarse. Para las pruebas de predicción LSTM, se ha utilizado un servidor que dispone el grupo de investigación GSIC/EMIC[19] de la Universidad de Valladolid. En la Figura 3.5 se puede consultar concretamente cuál es el *hardware* que posee dicho ordenador.

HPE PROLIANT DL380 GEN 10	1,00
HPe Intel Xeon Silver 4210 caché de 13,75 M 2,20Gz	1,00
HPE 32GB 2RX4 PC4-2933Y-R SMART KIT	3,00
HPE 800W Flex Slot Platinum Hot Plug	1,00
HPE 1.8TB SAS 2.5" 10K SFF SC 512e DS HDD	2,00
HP HPE DL38X GEN10 HIGH PERF FAN	1,00
HPE DL380 GEN10 HIGH PERF HEATSINK	1,00
GRAFICA NVIDIA SERVER RTX 5000 16GB DDR6	2,00
(2) Intel Xeon S4210 10 core	
128 GB (4 x 32GB) PC4-2933Y DDR4 RDIMM	
Smart Array P408i-a SR	
(2)HPE 1.8TB 12G 10k rpm HPL SAS SFF (2.5in)	
(2) GPU NVIDIA RTX5000	
HPE DL38x Gen10 High Performance Fan Kit	
Doble fuente 800W Flex Slot Platinum Hot Plug Low	
HPE Ethernet 10Gb 2P 530T Adapter	

Figura 3.5. Detalle del hardware empleado para los entrenamientos de técnicas LSTM.

[17] https://www.jetbrains.com/es-es/pycharm/
[18] Disponible en https://github.com/bzantium/EA-LSTM
[19] https://www.gsic.uva.es/

Como aspectos a destacar, el ordenador posee dos tarjetas gráficas *Quadro RTX 5000*[20], las cuales permiten conseguir una elevada eficiencia en tiempo, en comparación con usar la CPU, gracias al empleo de software especializado para la paralelización en el entrenamiento de técnicas de *Deep Learning*. Para el almacenamiento de resultados, se hizo uso de su capacidad de casi 600 GB. El procesador es un *Intel Xeon Silver 4210*[21], que forma parte de una serie de procesadores que ofrecen un desempeño esencial, una memoria de mayor velocidad, baja latencia y eficiencia energética, pensados para satisfacer exigentes necesidades informáticas relacionadas con inteligencia artificial, computación de alto rendimiento, redes, etc. Un aspecto a tener en cuenta es que, normalmente, el *hardware* de los teléfonos móviles (donde se pretendería instalar el predictor basado en técnicas LSTM) no es tan potente como el de este ordenador. Es más: a corto o medio plazo no se espera que los teléfonos móviles dispongan de una potencia similar a la de este ordenador. Una posible solución sería que los móviles fuesen meros recopiladores de información, siendo esta información transmitida a un ordenador central con la suficiente potencia como para entrenar un modelo y proporcionar las correspondientes predicciones. Es decir, el entrenamiento se llevaría a cabo en máquinas que exceden con mucho la capacidad de procesamiento del móvil, mientras que el uso del modelo ya sí que se podría hacer en el móvil.

3.6. Detalles de implementación de los algoritmos de predicción

3.6.1. Media Móvil Exponencialmente Ponderada

Concretamente, la implementación de la Media Móvil Exponencialmente Ponderada (EWMA) con la que se ha trabajado en este proyecto se expresa en la siguiente formulación, siendo *n* la longitud de la serie temporal:

$$x_{EWMA}(t) = \alpha * x(t-1) + (1-\alpha) * x_{EWMA}(t-1) \text{ con } t = 2, \dots, n-1$$

Se ha optado por no definir predicción alguna para el instante inicial (t=0), y para el primer instante (*t=1*) tomar como predicción el dato de t=0: $x_{EWMA}(t=1) = x(t=0)$.

3.6.2. Media Armónica

La implementación concreta de la Media Armónica (*Harmonic Mean*) con la que se ha trabajado en este proyecto se expresa en la siguiente formulación:

$$x_{Harmonic\ Mean}(i) = \frac{h}{\sum_{m=i-h}^{i-1} x(m)} \text{ con } i = h, h+1, \dots, n-1$$

[20] Para más información, consultar https://www.nvidia.com/es-es/design-visualization/previous-quadro-desktop-gpus/

[21] Para más información, consultar la página web oficial de Intel en el siguiente URL: https://www.intel.la/content/www/xl/es/products/details/processors/xeon/scalable/silver.html

Siendo *n* el número total de valores del conjunto de datos. Nótese que el conjunto de datos contiene valores para *x(0)* hasta *x(n-1)*, y que los primeros *h* valores carecen de predicción alguna, se comienza a predecir a partir del instante *h*.

Se ha optado por tomar una ventana de tamaño *h*, y predecir $x_{Harmonic\ Mean}$ a partir del instante *h* utilizando, para ese primer instante, los valores reales de *x* entre los instantes 0 y *h-1* (incluido este último).

3.6.3. Detalles de la implementación de LSTM

En el artículo [3], los autores especifican que han utilizado una arquitectura LSTM de 3 capas, utilizando el optimizador Adam, y tamaño de bloque (*batch* size) de 32. En cada capa: 32 neuronas, pues afirman que no obtienen mejora significativa en el rendimiento cuando se usan más. Nuestro LSTM implementado[22] se trata de un modelo al que se han definido N capas de M neuronas cada una. La implementación LSTM 3x256, por ejemplo, donde N=3 y M=256, es una implementación con 3 capas de 256 neuronas cada una. Hemos probado con la arquitectura que los autores decían utilizar: 3 capas de 32 neuronas y, al no conseguir replicar los resultados, hemos probado otras como la de 3 capas de 256 neuronas. El último procedimiento que se hace tras pasar la entrada por cada capa oculta es el relativo a la linearización para que la salida tenga el tamaño esperado (en la implementación, el número de *features* o características a la entrada se indica en la variable `input_size`, mientras que el número de *features* o características en la salida se indica en la variable `output_size`. Con `hidden_sizei` se indica el número de neuronas en la capa i-ésima. Se ha establecido la probabilidad de descarte (`dropout_probability`) a 0.

Utilizando los métodos ya implementados en la librería Pandas, se lleva a cabo la lectura del fichero y reestructuración de acuerdo con el número de *features* deseado. En concreto, se ha hecho uso de la función `read_csv` de Pandas. En los experimentos de única característica (donde solo se toma como entrada el dato relativo a la capacidad en cada instante), se toma, a partir del vector de salida de dicha función, un vector de una columna y tantas filas como sea necesario (tantas filas como instantes en los que se ha registrado la capacidad). En cambio, en los experimentos de múltiples características, se redimensiona a un vector de 8 columnas y tantas filas como sea necesario, pues se toman 8 características para cada instante de registro de la capacidad (es decir, la capacidad, y otras 7).

3.7. Entrenamiento y test del modelo LSTM implementado

Para generar el *dataset* para las predicciones, se toma una ventana de puntos conocidos de tamaño `lookback`, y se intenta predecir (objetivo o *taget*) una ventana de puntos «futura» de tamaño `lookforward`. Todas estas manipulaciones las lleva a cabo la función `create_dataset`, que toma como entradas un vector, y los parámetros de

[22] El código utilizado para implementar tanto los métodos clásicos como los LSTM se encuentra disponible en https://github.com/gsic-emic/bwforecast

`lookback` y `lookforward`. La función proporciona una salida formada por dos tensores de *Torch*, uno con la información que se toma como «conocida», y otro con el objetivo a predecir. El tensor de información conocida consta de tres dimensiones: la primera indica el número de instantes en los que se toma la medida, la segunda indica el valor de `lookback` o `lookforward` (según corresponda), y la tercera indica el número de características. El tensor que contiene la información objetivo consta solamente de dos dimensiones, pues el objetivo es predecir solamente la capacidad en el futuro (una sola «característica»). Una vez creado el *dataset*, se procede a la división en subconjuntos de entrenamiento, validación y test.

Para la generación del conjunto de datos a utilizar en el modelo LSTM, se toma el conjunto completo de datos proporcionados en cada fichero `.csv` y se divide en subconjuntos de **entrenamiento**, **validación** y **test**. Un 60% del conjunto total se utiliza para entrenamiento, un 10% para validación, y el restante 30% para test, tal y como especifican los autores en [3].

A continuación, para cada subconjunto, se lleva a cabo la normalización de los datos de cada característica con respecto al máximo y mínimo registrados en el subconjunto de entrenamiento. En otras palabras, para cada característica, se normaliza cada dato de los tres subconjuntos con respecto al máximo y mínimo de dicha característica en el subconjunto de entrenamiento.

Se lleva a cabo un entrenamiento de 300 épocas. Hemos comprobado que el óptimo del error sobre las predicciones hechas en el subconjunto de validación se suele dar dentro de las primeras 50 épocas (ver Figura 3.7), por lo que consideramos que un entrenamiento con 300 es más que suficiente. CUDA[23] [47] (*Compute Unified Device Architecture*, Arquitectura de Dispositivo Unificado de Computación) es una plataforma de computación paralela y modelo de programación de NVIDIA que permite a los desarrolladores utilizar las unidades de procesamiento gráfico (GPU) de NVIDIA para ejecutar cálculos y tareas de elevada complejidad de forma más eficiente que en una CPU convencional. Si estuviese disponible CUDA, se mueven los subconjuntos de datos previamente generados a la GPU. Nosotros sí hemos ejecutado el modelo en GPU (véase el apartado 3.5).

La función de pérdidas es una basada en error cuadrático medio (MSE), concretamente la función `MSELoss()` de la librería *Torch-nn*. El optimizador utilizado es Adam, perteneciente a la librería *Torch Optim*.

Para el entrenamiento por lotes, obtenemos un «cargador de datos» con ayuda de la función `DataLoader()` de la librería *Torch Utils Data*. Optamos por desactivar la aleatoriedad (`shuffle=False`) y un tamaño de lote de 32 (`batch_size=32`).

Para cada una de las épocas, se ordena el entrenamiento del modelo (mediante la llamada a la función `train()`). Para cada lote, se preestablece a cero el gradiente del optimizador, se obtiene la predicción, se calculan las pérdidas entre el lote original y la predicción, se realiza la propagación hacia atrás (mediante la función `backward()` de `nn.MSELoss()`) y se avanza un paso del optimizador. Una vez hecho esto con todos

[23] https://developer.nvidia.com/cuda-toolkit

los lotes, se evalúa el modelo (función `eval()`). Deshabilitando el cálculo del gradiente, se calculan las predicciones para cada subconjunto de entrenamiento, validación y test, y se calculan las pérdidas para almacenarse en un vector para su posterior representación gráfica. Si las pérdidas en el subconjunto de validación son las menores, se interpreta que el modelo está ajustado óptimamente, por lo que se almacenan los resultados de predicción de dicho modelo. Como condición de parada del entrenamiento, se establece que si los errores en el subconjunto del entrenamiento son los menores y si el número de época es superior a una determinada paciencia (`patience`), entonces se calcula la mejora (la diferencia del error en el subconjunto de entrenamiento con el error en el subconjunto de entrenamiento hace `patience` épocas, normalizado con respecto dicho error). Si dicha mejora es inferior a un factor (`delta`), entonces se para el proceso de entrenamiento, entendiéndose que no hay mejora significativa si se sigue entrenando.

Para la evaluación del rendimiento, se toman las predicciones que dieron el menor error en el subconjunto de validación, se desnormalizan, y se calcula, para cada subconjunto y para cada punto de la ventana futura (`lookforward`) las métricas de RMSE y MAE. Para generar los histogramas, se hace uso de la función `hist()` de la librería *Matplotlib Axes*.

3.8. Resultados

Se dan los resultados para el subconjunto de test en las siguientes tablas. Para el autobús *Bus M15*, los resultados se muestran en la siguiente Tabla 3.4. Como podemos ver, nuestras implementaciones de LSTM (LSTM MultiV 3x256, LSTM MultiV 3x32 y LSTM MultiV 2x32) no logran obtener un RMSE tan bajo como la implementación LSTM de los autores (LSTM Autores [3]).

	RMSE	**MAE**
1 steps ahead		
Naive	4.6259	3.2885
EWMA	4.3416	**3.1045**
Harmonic Mean	4.8915	3.3033
LSTM MultiV 3x256	5.5863	4.1706
LSTM MultiV 3x32	5.5281	4.1130
LSTM Univ 2x32	5.5517	4.0304
LSTM Autores [3]	**4.1899**	3.1052
2 steps ahead		
Naive	5.1040	3.4543
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.5817	4.1882
LSTM MultiV 3x32	5.5399	4.1343
LSTM Univ 2x32	5.5509	4.0264
LSTM Autores [3]	**4.6514**	**3.2973**
3 steps ahead		
Naive	5.6675	3.8070
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.5934	4.2068
LSTM MultiV 3x32	5.5475	4.1476
LSTM Univ 2x32	5.5497	4.0247
LSTM Autores [3]	**5.1524**	**3.6317**
4 steps ahead		
Naive	6.0874	4.0859
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.6063	4.2156
LSTM MultiV 3x32	5.5549	4.1585
LSTM Univ 2x32	**5.5495**	**4.0242**
LSTM Autores [3]	-	-

Tabla 3.4. Resultados para la traza del Bus M15 – 1, 2, 3 y 4 steps ahead.

Para ilustrar la forma que tiene la señal de predicción, se muestran las siguientes gráficas:

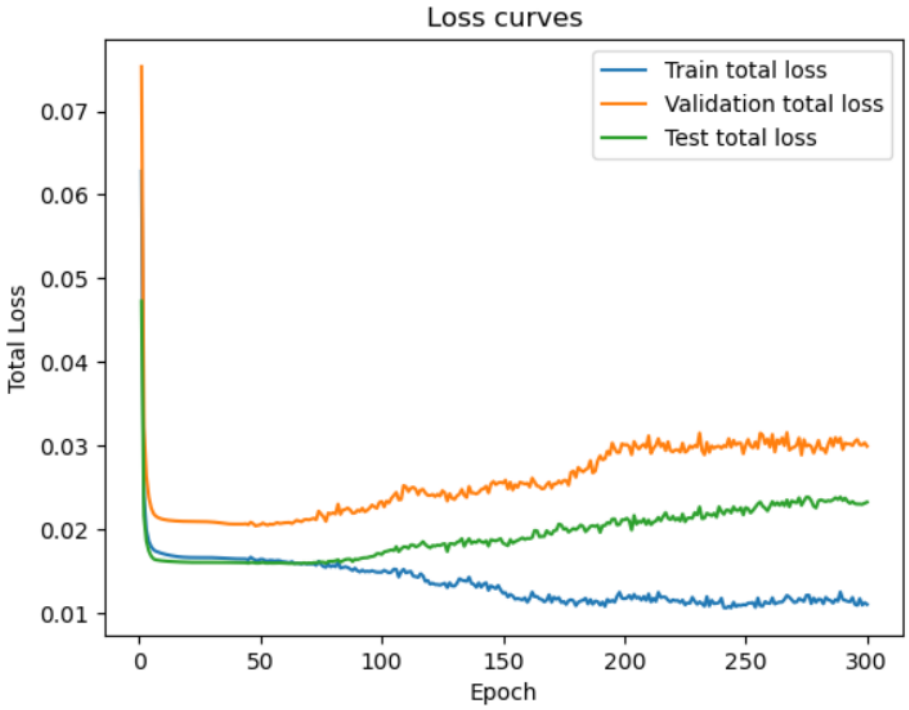

Figura 3.6. Curva de pérdidas Bus M15 con LSTM multivariable de 3x32 neuronas.

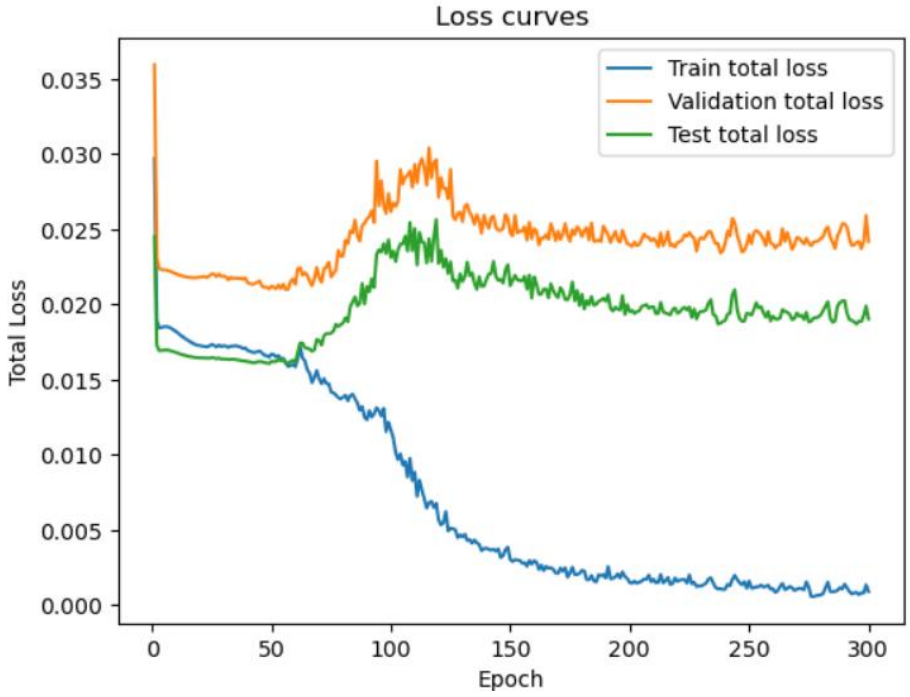

Figura 3.7. Curva de pérdidas Bus M15 con LSTM multivariable de 3x256 neuronas.

Inicialmente, hicimos pruebas entrenando con hasta 2000 épocas, y en todos los casos el codo del mínimo se veía siempre antes. Incluso antes de las 100 épocas. Tal y como se ha implementado, el proceso de entrenamiento continúa hasta las 300 épocas (o hasta que no haya mejora significativa), pero la predicción que se almacena es la correspondiente a la época de entrenamiento en la que se ha dado el menor error de validación (véase el «codo» de las gráficas del error de validación de la Figura 3.6 y de la Figura 3.7). Se puede apreciar que el codo estará en torno a las 50 épocas.

Analíticamente, el codo de la Figura 3.6 está en 48 épocas, y el de la Figura 3.7 está en 56 épocas.

En la época en la que se produjo el óptimo error de validación, se obtienen la siguiente gráfica de predicción (véase la Figura 3.8). Se puede ver que la amplitud de las predicciones es, aparentemente, menor que la de la señal original, tanto en el subconjunto de entrenamiento como en el de validación y test.

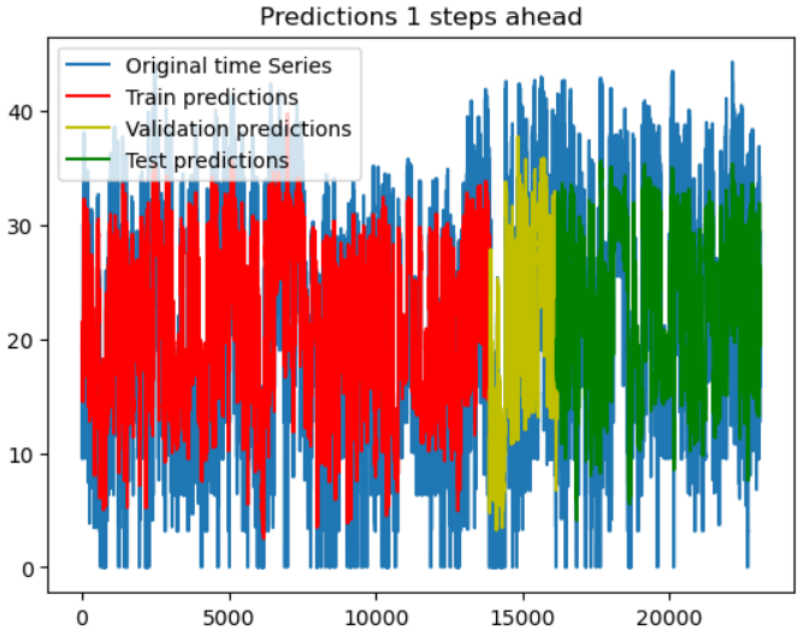

Figura 3.8. Señal original y predicciones en la línea Bus M15 con LSTM multivariable de 3x32 neuronas. Eje X: tiempo (s). Eje Y: capacidad (Mbps).

Para ilustrar la distribución de los errores de test, se muestran los histogramas de los errores de test, en las siguientes figuras:

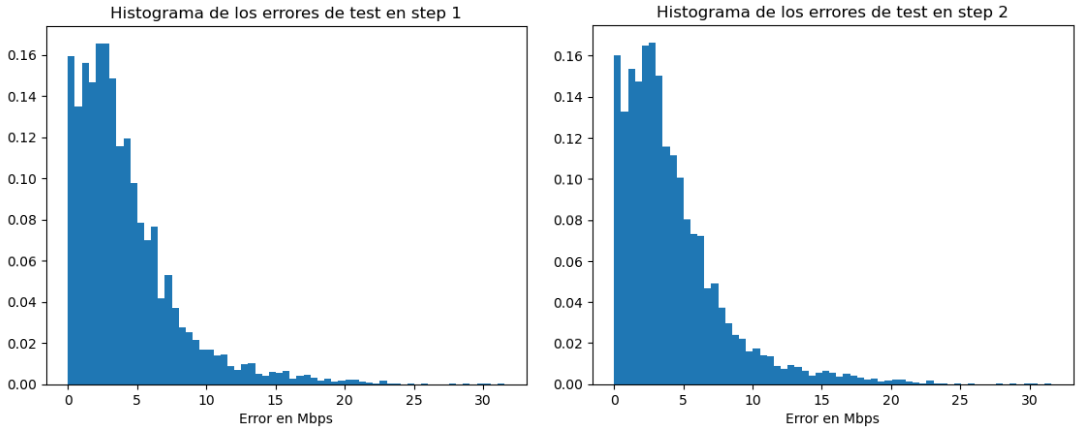

Figura 3.9. Histograma de los errores de predicción en el subconjunto de test en la línea Bus M15, utilizando la arquitectura LSTM de 3 capas de 32 neuronas cada una, steps 1 y 2. Anchura de bin = 0.5.

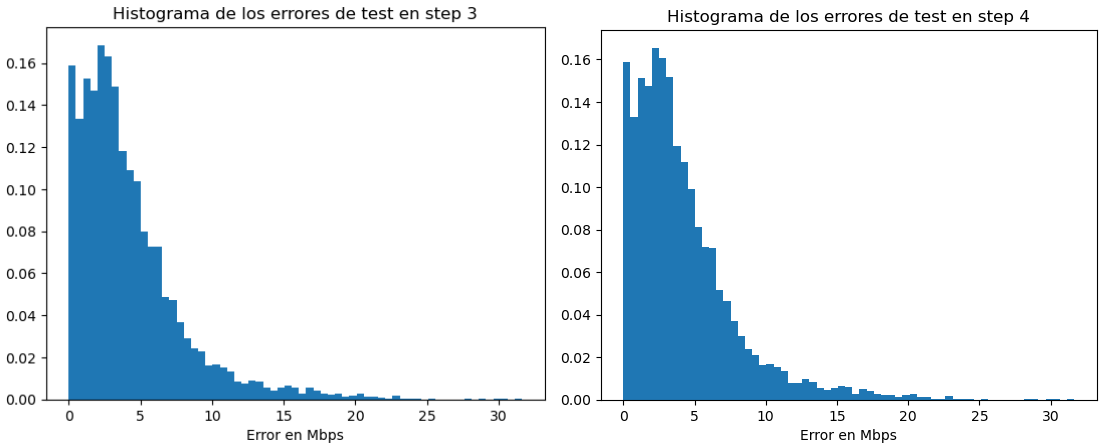

Figura 3.10. Histograma de los errores de predicción en el subconjunto de test en la línea Bus M15, utilizando la arquitectura LSTM de 3 capas de 32 neuronas cada una, steps 3 y 4. Anchura de bin = 0.5.

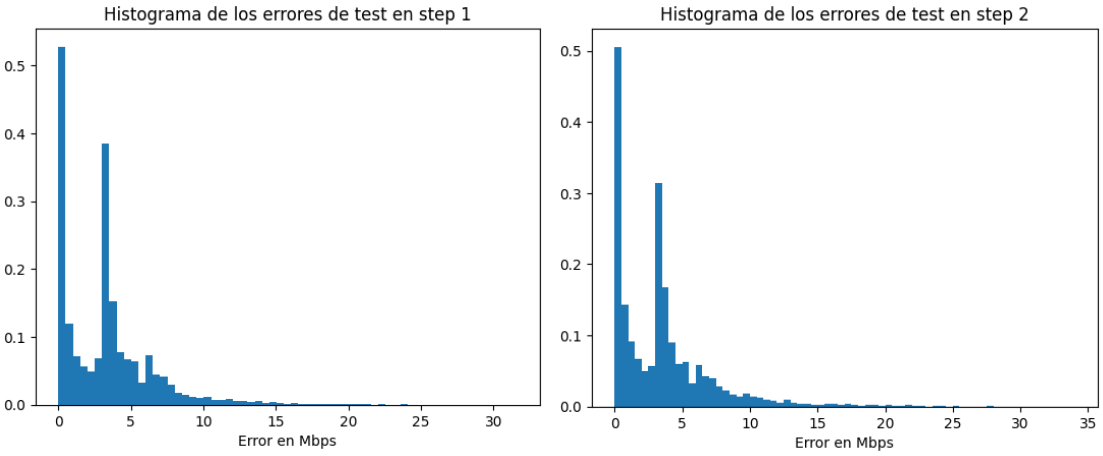

Figura 3.11. Histograma de los errores de predicción en el subconjunto de test en la línea Bus M15, steps 1 y 2, utilizando el predictor Naive. Anchura de bin = 0.5.

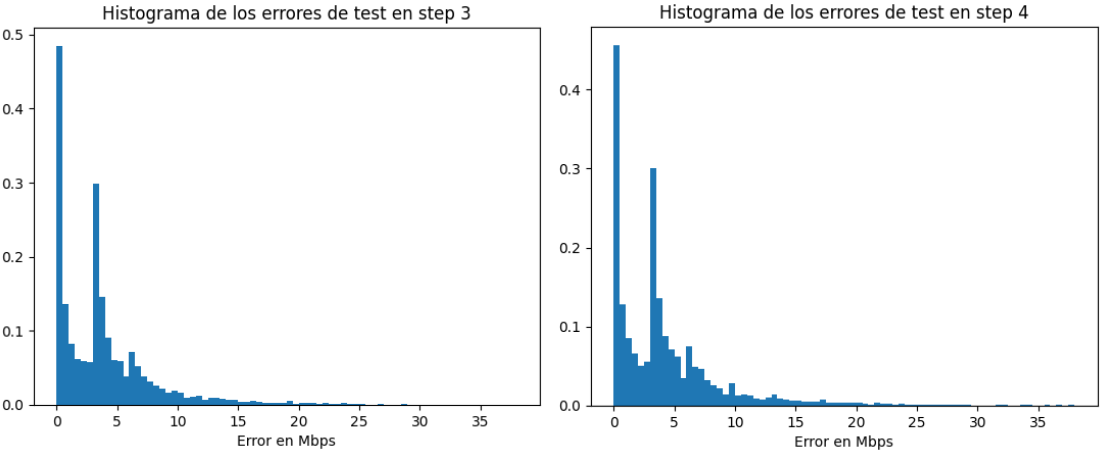

Figura 3.12. Histograma de los errores de predicción en el subconjunto de test en la línea Bus M15, steps 3 y 4, utilizando el predictor Naive. Anchura de bin = 0.5.

En el resto de líneas de autobús encontramos los resultados que se muestran en las siguientes tablas. Para el autobús *Bus B61*, los resultados se muestran en la siguiente Tabla 3.5.

	RMSE	**MAE**
1 steps ahead		
Naive	4.4995	3.1111
EWMA	**4.2035**	**2.9720**
Harmonic Mean	4.9796	3.3559
LSTM MultiV 3x256	5.5685	4.1679
LSTM MultiV 3x32	5.4747	4.0554
LSTM Univ 2x32	5.1758	3.6660
LSTM Autores [3]	4.2137	-
2 steps ahead		
Naive	4.8906	**3.2693**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.5723	4.1741
LSTM MultiV 3x32	5.4686	4.0431
LSTM Univ 2x32	5.1869	3.6614
LSTM Autores [3]	**4.5162**	-
3 steps ahead		
Naive	5.3546	**3.5882**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.5712	4.1807
LSTM MultiV 3x32	5.4666	4.0364
LSTM Univ 2x32	5.1908	3.6645
LSTM Autores [3]	**4.9645**	-
4 steps ahead		
Naive	5.6655	3.8090
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.5790	4.1954
LSTM MultiV 3x32	5.4682	4.0355
LSTM Univ 2x32	**5.1878**	**3.6635**
LSTM Autores [3]	-	-

Tabla 3.5. Resultados para la traza del Bus B61 – 1, 2, 3 y 4 steps ahead.

Para el autobús *Bus B62*, los resultados se muestran en la siguiente Tabla 3.6.

	RMSE	**MAE**
1 step ahead		
Naive	4.7765	3.1958
EWMA	4.4948	**3.0483**
Harmonic Mean	5.3271	3.4785
LSTM MultiV 3x256	5.9098	4.3061
LSTM MultiV 3x32	5.8758	4.2657
LSTM Univ 2x32	5.7754	4.0694
LSTM Autores [3]	**4.4797**	-
2 steps ahead		
Naive	5.3323	**3.3871**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.9106	4.3052
LSTM MultiV 3x32	5.8876	4.2760
LSTM Univ 2x32	5.7694	4.0539
LSTM Autores [3]	**4.8975**	-
3 steps ahead		
Naive	5.8991	**3.7796**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.9109	4.3046
LSTM MultiV 3x32	5.9000	4.2977
LSTM Univ 2x32	5.7671	4.0489
LSTM Autores [3]	**5.4020**	-
4 steps ahead		
Naive	6.3432	**4.0495**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.9110	4.3042
LSTM MultiV 3x32	5.9179	4.3224
LSTM Univ 2x32	**5.7662**	4.0496
LSTM Autores [3]	-	-

Tabla 3.6. Resultados para la traza del Bus B62 – 1, 2, 3 y 4 steps ahead.

Para el autobús *Bus B16*, los resultados se muestran en la siguiente Tabla 3.7.

	RMSE	**MAE**
1 step ahead		
Naive	4.0101	2.5626
EWMA	3.6965	2.4155
Harmonic Mean	4.0530	2.5444
LSTM MultiV 3x256	4.7386	3.6838
LSTM MultiV 3x32	5.0915	3.6904
LSTM Univ 2x32	4.8628	3.3702
LSTM Autores [3]	3.6762	-
EA-LSTM	3.7082	2.6274
TG-LSTM [10]	**2.7940**	**2.3250**
2 steps ahead		
Naive	4.1378	**2.4618**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	4.7303	3.6747
LSTM MultiV 3x32	5.0825	3.6649
LSTM Univ 2x32	4.7500	3.2760
LSTM Autores [3]	**3.8984**	-
3 steps ahead		
Naive	4.3773	**2.6433**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	4.7249	3.6686
LSTM MultiV 3x32	5.0760	3.6465
LSTM Univ 2x32	4.7476	3.2712
LSTM Autores [3]	**4.0465**	-
4 steps ahead		
Naive	**4.6977**	**2.8040**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	4.7215	3.6647
LSTM MultiV 3x32	5.0718	3.6345
LSTM Univ 2x32	4.8192	3.3227
LSTM Autores [3]	-	-

Tabla 3.7. Resultados para la traza del Bus B16 – 1, 2, 3 y 4 steps ahead.

En la siguiente Tabla 3.8 se muestran los resultados para el metro *Subway 7 Train*.

	RMSE	**MAE**
1 step ahead		
Naive	5.3442	3.5494
EWMA	4.9008	3.3721
Harmonic Mean	5.7330	3.5895
LSTM MultiV 3x256	5.1417	3.7739
LSTM MultiV 3x32	5.0536	3.7254
LSTM Univ 2x32	5.0668	3.7526
LSTM Autores [3]	**4.3964**	**3.2936**
EA-LSTM	4.8980	3.4755
2 steps ahead		
Naive	5.3606	**3.3321**
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.1403	3.7643
LSTM MultiV 3x32	5.0553	3.7186
LSTM Univ 2x32	5.0634	3.7291
LSTM Autores [3]	**4.6065**	3.3622
3 steps ahead		
Naive	5.8267	3.6947
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.1386	3.7573
LSTM MultiV 3x32	5.0507	3.7040
LSTM Univ 2x32	5.0610	3.7108
LSTM Autores [3]	**4.8869**	**3.5845**
4 steps ahead		
Naive	5.9534	3.7876
EWMA	-	-
Harmonic Mean	-	-
LSTM MultiV 3x256	5.1398	3.7554
LSTM MultiV 3x32	**5.0469**	**3.6935**
LSTM Univ 2x32	5.0613	3.6996
LSTM Autores [3]	-	-

Tabla 3.8. Resultados para la traza del Subway 7 Train – 1, 2, 3 y 4 steps ahead.

En la Figura 3.13 se muestran los resultados pictóricamente.

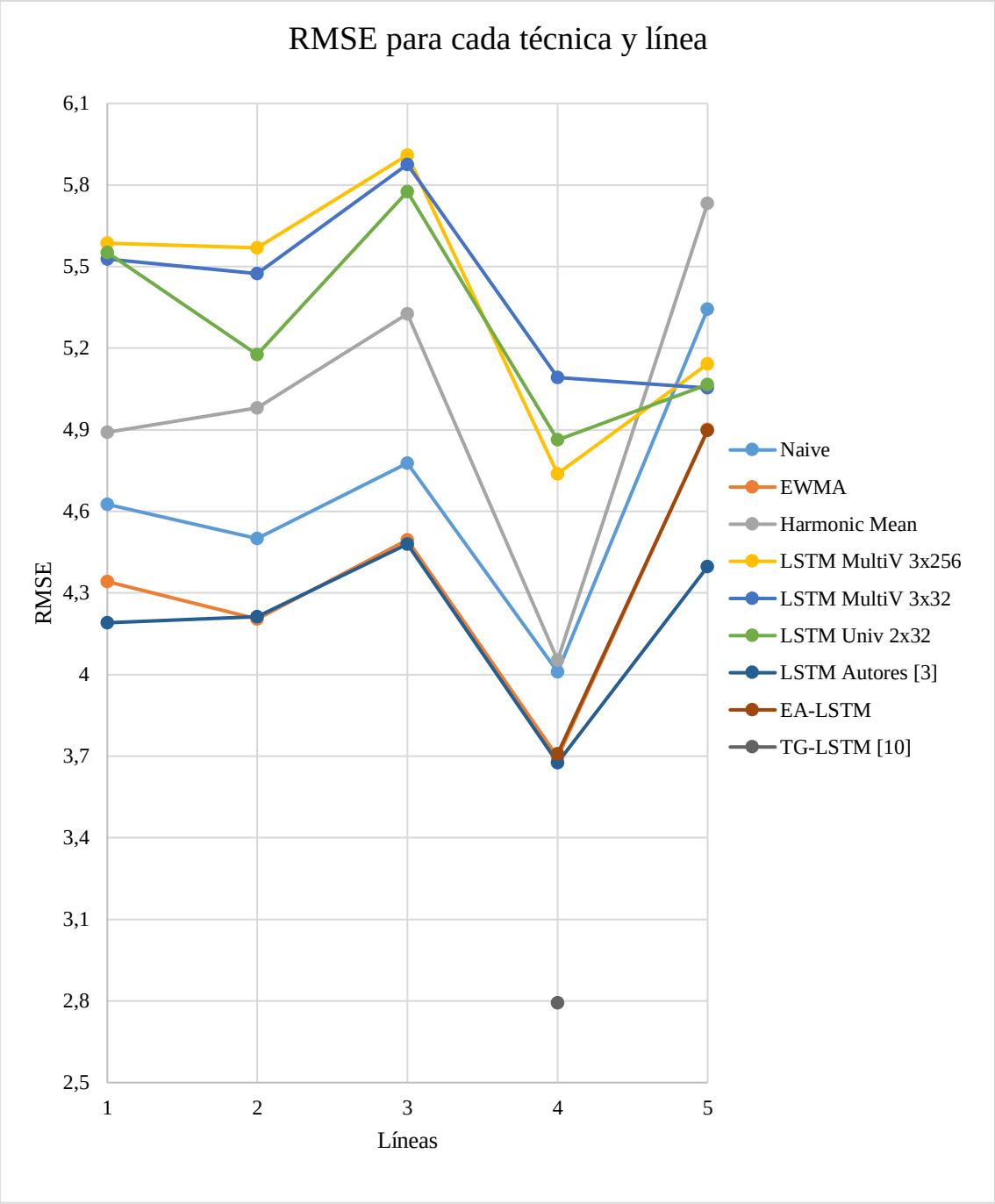

Figura 3.13. Gráfica ilustrativa del RMSE que ha dado cada técnica en cada línea.

Siendo:

Número de línea	Nombre de línea
1	Bus M15
2	Bus B61
3	Bus B62
4	Bus B16
5	Subway 7 Train

Tabla 3.9. Correspondencia número de línea – nombre de línea.

Como se puede ver en la gráfica ilustrativa de los resultados (véase la Figura 3.13), no hay mucha diferencia entre todas las líneas que se reportan en el conjunto de datos.

Si se pone la atención en el resultado del modelo LSTM autores (véase la Figura 3.13, traza «LSTM Autores»), se puede ver que el RMSE oscila entre 3,5 y 4,5 según la ruta o línea seleccionada. Se puede ver que las predicciones en la línea 4 – *Bus B16* son las que obtienen el menor RMSE, mientras que las predicciones en la línea 3 – *Bus B62* son las peores en términos de RMSE (mayor RMSE).

3.9. Discusiones

Como se puede ver en las tablas de resultados (véase las sección 3.8), con nuestras arquitecturas de LSTM (*LSTM MultiV 3x256*, *LSTM MultiV 3x32* y *LSTM UniV 2x32*) no hemos logrado reproducir los resultados de [3]. Los autores de dicho artículo afirman haber logrado unos valores de RMSE y MAE menores. Dichos valores se plasman en las tablas de resultados etiquetados como *LSTM Autores*. Hemos estado investigando cuáles pueden ser las posibles causas de estas diferencias. En primer lugar, no tenemos total certeza de que el conjunto de datos que nosotros estamos utilizando sea el mismo que el que utilizan en [3]. Si bien en el pie de la página 4 de [3] se hace referencia al directorio[24] de *GitHub* del *dataset* (directorio de donde hemos tomado el *dataset* con el que hemos hecho las pruebas), hemos calculado las estadísticas de las trazas (véase la Tabla 3.2 de este documento) para compararlas con las estadísticas proporcionadas por ellos en la tabla 1 de [3]. Así, hemos generado una tabla comparativa (véase la Tabla 3.3) y, dadas las escasas diferencias, hemos confiado que se trata del mismo conjunto de datos.

Por otro lado, hemos probado con la información de los parámetros del modelo LSTM que especificaban en [3], información que estaba incompleta, por lo que hemos tenido que explorar otros parámetros, así como otras distintas combinaciones de parámetros, pero tampoco hemos sido capaces de acercarnos a los resultados que reportan en [3]. Esto nos lleva a sacar una conclusión y una crítica: todos los autores deberían proporcionar información acerca de todos los hiperparámetros que han utilizado o, mejor todavía, proporcionar el código con el que han realizado las implementaciones y experimentos.

Observando también la Tabla 3.3, podemos ver que la línea *7 Train*, en comparación con su media, tiene un error muy grande (véase la desviación estándar). En general, podemos ver que la media es superior a la desviación estándar. Pero, en la línea *7 Train*, la desviación estándar tiene aproximadamente el mismo valor que la media de la capacidad de la línea.

En la Figura 3.8 se muestra la forma de la señal original junto a la señal de predicción, apreciándose que la predicción no llega a los extremos. Parece ser que es porque LSTM no es capaz de seguir las variaciones tan bruscas. En todas las arquitecturas de LSTM probadas se ha dado el mismo problema: con más neuronas por capa la predicción lograba ajustarse un poco mejor, pero sin llegar tampoco a los extremos.

[24] https://github.com/NYU-METS/Main

Hemos comentado que no hay mucha diferencia entre todas las líneas. Esto puede deberse al hecho de que haya muchas estaciones base, de manera que los dispositivos móviles tienen un nivel de cobertura más o menos constante a lo largo del recorrido de la ruta.

En todas las líneas (véase la Figura 3.13) podemos ver que el comportamiento de los métodos es consistente, es decir, los peores métodos son los peores en todas las líneas, y los mejores métodos son los mejores. Existe un caso particular: la línea *Subway 7 Train*, donde no se cumple la generalización que acabamos de mencionar. Como las predicciones con *Naive* y *Harmonic Mean* son peores, significa que en esa línea existen cambios bruscos en períodos cortos de tiempo. El primero (*Naive*) se basa en predecir que el siguiente valor de capacidad va a ser igual al anterior, y en el segundo (*Harmonic Mean*) se hace una media (armónica) de los valores anteriores: en ambos casos tienen relevancia los valores anteriores, de manera que, si en el siguiente instante la capacidad varía bruscamente, ambos no lo van a poder predecir con exactitud.

Analizando los *datasets*, vemos que las variaciones en la capacidad son lentas, por lo que se pueden seguir con facilidad. Tal y como detallan los autores del *dataset* [3], se trata de un *dataset* multivariable centrado en rutas de transporte público fijas, coleccionando **datos de ocho viajes de ida y vuelta** con una duración aproximada de 30 minutos y **concatenándolos** formando un *dataset* con una duración total de unas 30 horas, pues toman medidas de la capacidad cada 1000 milisegundos. Nos estamos planteando si realmente tiene sentido esta concatenación de varios días, pues en función de la hora del día o el sentido de la ruta, parece evidente que el patrón de variación de la capacidad va a cambiar. Nos parece más lógico tomar medidas y generar trazas de día en día, pues el patrón va a cambiar secuencialmente de una hora a la siguiente, creándose picos a determinadas horas del día. Es decir, habrá unas ciertas horas del día en las que haya un mayor número de usuarios conectados a la red, afectando a la capacidad disponible para cada uno. Lo mismo sucede entre distintos días de la semana, siendo diferente la evolución un lunes que un sábado, por ejemplo.

Los predictores basados en LSTM son buenos cuando hay periodicidades a largo plazo (pues destacan en que son capaces de capturar dichas periodicidades), y en el conjunto de datos que hemos utilizado las trazas no son de un día entero (véase la Tabla 3.2. Estadísticas de las trazas del sistema de transporte público de NYC (Mbps)). En dicha tabla se puede ver que la traza que más dura es la del Bus M15, con una duración de 23108 segundos o 6.42 horas aproximadamente (1 milisegundo entre cada medida) por lo que los ritmos del día no se están capturando. Dentro de 30 minutos, por ejemplo, es probable que no haya dependencias a largo plazo, que es donde realmente destaca LSTM. Aunque también, seguramente existan otras dependencias, como alguna dependencia de la posición del móvil en el recorrido de la línea, y esas dependencias también las puede aprender y aprovechar el predictor LSTM.

En la sección 3.8 se han mostrado los resultados de la ejecución de los predictores utilizando, entre otros, la traza de la línea de autobús *Bus M15*, en concreto, los histogramas de los errores de predicción en el subconjunto de test. En la Figura 3.14 se muestran los histogramas de los errores de predicción en el subconjunto de test, utilizando la arquitectura LSTM de 3 capas de 32 neuronas cada una, y anchura de *bin* 0.5 (a la

izquierda) y utilizando el predictor *Naive*, también con anchura de *bin* 0.5 (a la derecha), ambos de la línea de *Bus M15*. Si prestamos atención al histograma de los errores de test de las predicciones de LSTM (Figura 3.14 izquierda), podemos observar un decaimiento típico de los errores de predicciones proporcionadas por LSTM. Entre 0 y 5 Mbps tenemos aproximadamente una zona plana (un valor más o menos constante de errores de valor entre 0 y 5 Mbps). A continuación, decrece. Es decir, hay menos errores de valor superior. Por otro lado, observemos el histograma de los errores de test de las predicciones de *Naive* (Figura 3.14 derecha). Podemos apreciar dos grandes picos en errores de valor aproximadamente 0 Mbps (sin error) y 4 Mbps. Es decir, vamos a tener muchos más errores con esos valores si utilizamos el predictor *Naive* que si utilizamos LSTM. Si la capacidad apenas varía ligeramente, parece razonable predecir que la capacidad en el siguiente instante va a ser la misma que en el instante actual, y es en ese caso donde el predictor *Naive* está destacando (véase que hay bastantes más predicciones *Naive* que LSTM con error nulo). El predictor *Naive*, como se ha comentado en las secciones previas, apenas tiene carga computacional, nada comparable con la carga computacional que requiere LSTM (tanto es así, que hemos necesitado ejecutarlo en GPUs para obtener resultados en un tiempo razonable). Por tanto, si no hay mucha variación en la capacidad, no nos parece necesario decantarnos por un predictor LSTM, salvo que nos encontrásemos ante alguna aplicación particular en la que los errores superiores a 4 Mbps fuesen críticos (donde tenemos un pico de muchas predicciones *Naive* con ese error).

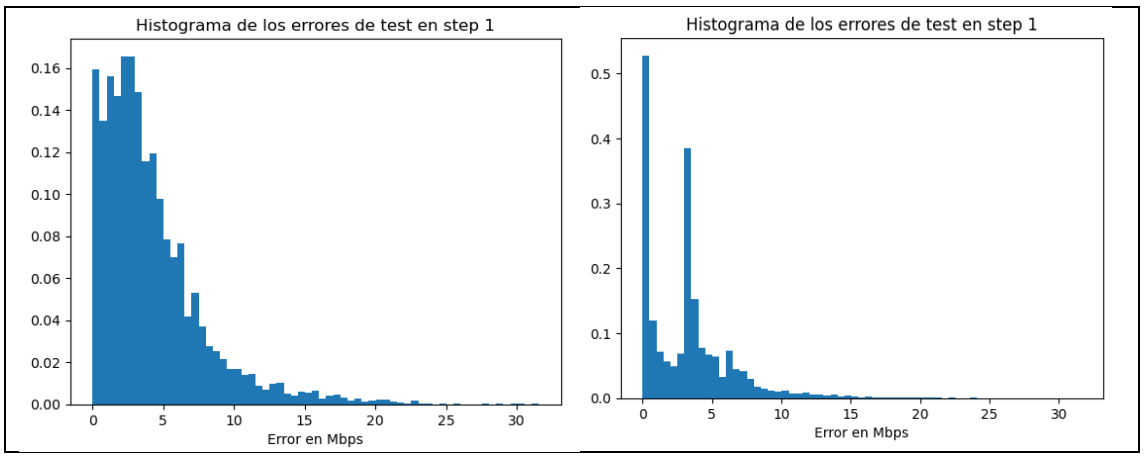

Figura 3.14. Histograma de los errores de predicción para la línea Bus M15 en el subconjunto de test en el primer step, utilizando la arquitectura LSTM de 3 capas de 32 neuronas cada una (izquierda) y utilizando el predictor Naive (derecha). En ambos casos, anchura de bin de 0.5.

Vamos a comprobar cuál es la probabilidad de error mayor que un determinado umbral con ambos predictores, para así demostrarlo cuantitativamente. Para LSTM, volvemos a ejecutar el modelo obteniendo el siguiente histograma (variante de 3 capas de 32 neuronas):

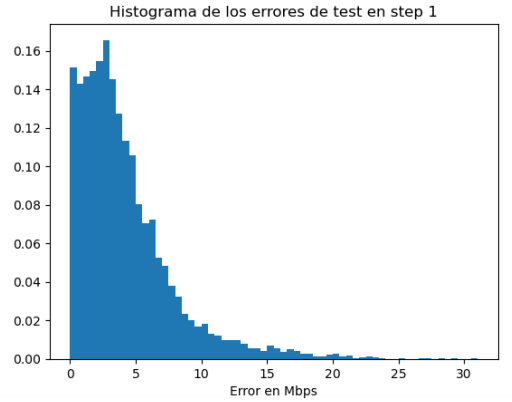

Figura 3.15. Histograma de los errores de predicción (nueva ejecución) para la línea Bus M15 en el subconjunto de test en el primer step, utilizando la arquitectura LSTM de 3 capas de 32 neuronas.

Para el primer *step*, obtenemos la siguiente tabla (Tabla 3.10), que muestra las probabilidades de errores por encima de varios umbrales (en megabits por segundo) para cada uno de los dos predictores considerados en la discusión.

Umbral (Mbps)	**Probabilidad de error con el predictor LSTM de 3 capas de 32 neuronas**	**Probabilidad de error con el predictor *Naive***
1 Mbps	0.8520	0.6733
2 Mbps	0.7040	0.6099
3 Mbps	0.5439	0.5512
4 Mbps	0.4078	0.2828
5 Mbps	0.2982	0.2101
6 Mbps	0.2229	0.1619
7 Mbps	0.1605	0.1032
8 Mbps	0.1173	0.0675
9 Mbps	0.0895	0.0516
10 Mbps	0.0709	0.0408

Tabla 3.10. Probabilidades de error mayor que determinados umbrales con predictores LSTM y Naive, en la línea de bus M15.

Habíamos comentado que un argumento con peso para elegir al predictor LSTM frente al *Naive* era que los errores que estuviesen por encima de un umbral fuesen críticos. Integrando el histograma como función de densidad de probabilidad, obtendríamos la probabilidad de error. Si se integra en todo el rango de valores de error, dicho área es la unidad. Para elaborar la Tabla 3.10, hemos integrado desde el umbral (en Mbps) en adelante, obteniendo la probabilidad de error superior a un umbral. Podemos apreciar en la Tabla 3.10 que, incluso con LSTM, las probabilidades de error superior a un umbral son superiores cuando se hace uso de LSTM. Por ejemplo, si fijamos el umbral en 5 Mbps, si se predice con el modelo LSTM, la probabilidad de error superior a 5 Mbps es de 0.2982, mientras que, si se predice con *Naive*, la probabilidad de error superior a 5 Mbps

es de 0.2101. Por tanto, para esta línea de autobús M15, no encontramos argumentos con peso suficiente que nos hagan inclinar la balanza hacia la elección de un predictor LSTM, dada su alta carga computacional y su precisión de predicción. Nos queda la duda de si con la arquitectura LSTM exacta que han utilizado los autores de [3] se obtendrían unos resultados tan buenos como dicen obtener.

Podríamos haber utilizado otros métodos de predicción, como los *Transformers* o los algoritmos de predicción en línea, para completar y fortalecer las discusiones previamente expuestas. Como última conclusión, sería necesario disponer de más conjuntos de datos y probar con ellos para llegar más lejos en las afirmaciones que hemos hecho.

3.10. Conclusiones

En este capítulo se ha presentado y descrito el conjunto de datos que se ha utilizado para la realización de los experimentos, así como los motivos de la elección de dicho conjunto de datos. Se ha detallado cuál ha sido la variante LSTM implementada y los motivos que nos ha llevado a seleccionar LSTM y no cualquiera de sus variantes. Se ha comentado el entorno de desarrollo empleado, así como las características del *hardware* en el que se han ejecutado los modelos LSTM, modelos que requieren una mayor carga computacional. Se ha descrito también cómo se ha llevado a cabo la implementación de cada uno de los predictores clásicos, así como la implementación de LSTM. Se han comentado los detalles del proceso de entrenamiento y test que han permitido llegar a las soluciones en términos de RMSE y MAE, presentados en tablas. Por último, se han hecho unas discusiones acerca de los resultados obtenidos.

Capítulo 4

Conclusiones y trabajo futuro

En el capítulo anterior se han realizado experimentos con predictores clásicos y un predictor LSTM que pretendía ajustarse lo máximo posible al predictor LSTM empleado en [3]. Durante las pruebas realizadas con las arquitecturas LSTM propuestas, no hemos conseguido lograr unos resultados tan buenos, en términos de RMSE y MAE, que los de los autores de [3].

Tras varios experimentos realizados, concluimos que es un problema de parámetros del modelo o del entrenamiento. Los autores no proporcionaban información completa acerca de los mismos, por lo que hemos tenido que implementar y entrenar el modelo con la escasa información de la que disponíamos, y razonando cuáles deberían ser aquellos parámetros desconocidos, o bien haciendo pruebas a ver si lográbamos alguna mejora significativa. No estamos seguros de que sea exactamente el mismo conjunto de datos que el utilizado por los autores, pero por las estadísticas analizadas, salvo ligeras diferencias, parece que sí. Existen ciertas diferencias entre líneas, pero no son significativas.

Los modelos LSTM dan buenos resultados de predicción, aunque hemos visto que no especialmente mucho mejores que los proporcionados por métodos con carga computacional mucho menor como *Naive* y EWMA (predictores clásicos). En nuestro caso, los predictores clásicos han proporcionado, en algunos casos, mejores valores de RMSE y MAE incluso que LSTM. El coste computacional de LSTM es mucho mayor, por lo que no compensa construir modelos LSTM en muchas aplicaciones reales. Sería justificable su uso en casos en los que penalizasen más los errores por encima de un determinado umbral si la probabilidad de error fuese menor (como sugiere la forma de su histograma), pero también hemos demostrado que, en nuestro caso y para el conjunto de datos utilizado, la probabilidad de error por encima de un umbral es superior con LSTM que con otros métodos clásicos.

El principal objetivo de este Trabajo de Fin de Grado era reproducir los resultados de predicción obtenidos con LSTM en el artículo [3] y entender su adecuación para el problema de predicción de la capacidad de enlace. Si bien no hemos logrado reproducir exactamente los resultados, estamos satisfechos por haber logrado entender los modelos LSTM y discutir su conveniencia. Por otro lado, teníamos por objetivos parciales explorar los conjuntos de datos disponibles para la predicción de la capacidad de enlace (lo cual hemos hecho y analizado de manera satisfactoria) y explorar las diferentes variantes de

LSTM utilizadas en la literatura. También hemos logrado programar y entrenar un modelo LSTM desde cero, lo cual nos ha permitido entenderlo en profundidad. Hemos comparado este modelo LSTM con los algoritmos básicos y hemos discutido si merecería la pena o no optar por un modelo LSTM.

En los próximos trabajos, se plantea explorar nuevos métodos de predicción. Sería buena idea empezar por las arquitecturas LSTM «enriquecidas», como pueden ser EA-LSTM (hemos trabajado con un predictor EA-LSTM ya programado[25] y, efectivamente, daba buenos resultados, véase la Figura 3.13). En este Trabajo de Fin de Grado, una de las tareas iniciales en lo que a aprendizaje automático se refiere ha sido la programación desde cero de un modelo LSTM. Como proceso formativo nos ha permitido entender con detalle el grano fino de este tipo de predictores. Para próximos experimentos en el futuro, se contempla utilizar algoritmos ya implementados, intentando entender muy bien el código de las implementaciones que se nos proporcionen y, partiendo de esa base, se experimentar y proponer mejoras.

Se propone explorar nuevos algoritmos alternativos, como los métodos de aprendizaje en línea (no en bloque), ya que presentan algunas ventajas entre las que destaca la menor carga computacional que requieren, o los *transformers*.

Se plantea experimentar también con otros conjuntos de datos públicos para poder robustecer mejor las conclusiones extraídas y poder llegar a alguna conclusión más. También se plantea la posibilidad de generar nuestro propio conjunto de datos.

Se propone comprender mejor cuál es la raíz del problema desde el punto de vista de las redes 4G: estudiar cada una de las características que se dan en los conjuntos de datos para entender y razonar el mayor o menor impacto que puedan tener en la predicción de la capacidad.

Se pretende evaluar también la «transferencia de modelos». Esto es: entrenar un modelo con los datos de una línea, y evaluar el rendimiento de predicción con los datos de otra. Esto es de gran interés, pues en aplicaciones prácticas no pretenderemos entrenar un modelo LSTM para todos y cada uno de los casos, sino reaprovecharemos un modelo pre-entrenado para predecir la capacidad en la vida real.

[25] Disponible en https://github.com/bzantium/EA-LSTM

Referencias

[1] Orange España, «Las ondas. ¿Cómo funciona una red móvil?,» Orange, [En línea]. Available: https://radio-waves.orange.com/es/como-funciona-una-red-movil/. [Último acceso: 10 05 2024].

[2] I. Poole, Newnes Guide To Radio and Communications Technology, Burlington: Newnes (An imprint of Elsevier), 2003.

[3] L. Mei, J. Gou, Y. Cai, H. Cao y Y. Liu, «Realtime mobile bandwidth and handoff predictions in 4G/5G networks,» *Computer Networks,* vol. 204, nº 1389-1286, p. 108736, 2022.

[4] Wifi Talents, «Internet Traffic Statistics: Latest Data & Summary,» [En línea]. Available: https://wifitalents.com/statistic/internet-traffic/. [Último acceso: 15 06 2024].

[5] J. F. Kurose y K. W. Ross, Redes de Computadoras. Un enfoque descendente. 7ª edición, Pearson, 2017.

[6] Jazztel Info, «Consejos y trucos: Por qué varía la velocidad de conexión a internet desde el móvil,» Jazztel, [En línea]. Available: https://www.jazztel.info/por-que-varia-velocidad-conexion-internet-desde-movil/. [Último acceso: 10 05 2024].

[7] D. Raca, D. Leahy, C. J. Sreenan y J. J. Quinlan, «Beyond throughput, the next generation: a 5G dataset with channel and context metrics,» *Proceedings of the 11th ACM Multimedia Systems Conference,* pp. 303-308, 2020.

[8] A. Narayanan, E. Ramadan, J. Carpenter, Q. Liu, Y. Liu, F. Qian y Z.-L. Zhang, «A first look at commercial 5G performance on smartphones,» *Proceedings of the Web Conference,* pp. 894-905, 2020.

[9] L. Mei, R. Hu, H. Cao, Y. Liu, Z. Han, F. Li y J. Li, «Realtime mobile bandwidth prediction using LSTM neural network and Bayesian fusion,» *Computer Networks,* vol. 182, nº 1389-1286, p. 107515, 2020.

[10] M. Zhang, X. Jiang, G. Jin, P. Li y H. Chen, «CapRadar: Real-time adaptative bandwidth prediction for dynamic wireless networks,» *Computer Networks,* vol. 233, p. 109865, 2023.

[11] I. Livieris y P. Pintelas, «A survey on algorithms for training artificial neural networks,» 09 2008.

[12] S. Hochreiter y J. Schmidhuber, «Long Short-term Memory,» *Neural computation,* vol. 9, pp. 1735-80, 1997.

[13] G. Y. Li, Z. Xu, C. Xiong, C. Lee, S. Zhang y H. Yang, «5G communications: Concepts and technologies,» *IEEE Network,* pp. 44-51, 2018.

[14] R. Jozefowicz, W. Zaremba y I. Sutskever, «An Empirical Exploration of Recurrent Network Architectures,» *Proceedings of the 32nd International Conference on Machine Learning,* vol. 37, pp. 2342-2350, 2015.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser y I. Polosukhin, «Attention is All you Need,» *Advances in Neural Information Processing Systems,* vol. 30, 2017.

[16] R. Lasser, «Engineering Method. Electrical and Computer Engineering Design Handbook,» [En línea]. Available: https://sites.tufts.edu/eeseniordesignhandbook/2013/engineering-method/. [Último acceso: 15 06 2024].

[17] B. Sainz de Abajo, Tecnologías Inalámbricas. Documentación de la asignatura Sistemas de Comunicación. ETSI Telecomunicación, Valladolid: Universidad de Valladolid, 2022.

[18] A. Ghosh, J. Zhang, J. G. Andrews y R. Muhamed, Fundamentals of LTE, Prentice Hall, 2005.

[19] T. S. Rappaport, Wireless Communications: Principles and Practice, Prentice Hall Communications Engineering and Emerging Technologies Series.

[20] J. Jiang, V. Sekar y H. Zhang, «Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE,» *Association for Computing Machinery,* p. 97–108, 2012.

[21] X. Yin, A. Jindal, V. Sekar y B. Sinopoli, «A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP,» *Association for Computing Machinery,* pp. 325-338, 2015.

[22] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu y J. Lyu, «Real-time bandwidth prediction and rate adaptation for video calls over cellular networks,» *Association for Computing Machinery,* nº 12, pp. 1-11, 2016.

[23] M. Mirza, J. Sommers, P. Barford y X. Zhu, «A machine learning approach to TCP throughput prediction,» *Association for Computing Machinery*, pp. 97-108, 2007.

[24] G. Tian y Y. Liu, « Towards agile and smooth video adaptation in dynamic HTTP streaming,» *Association for Computing Machinery*, pp. 109-120, 2012.

[25] K. Winstein, A. Sivaraman y H. Balakrishnan, «Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks,» *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). USENIX Association*, pp. 459-471, 2013.

[26] C. Yue, R. Jin, K. Suh, Q. Yanyuan y W. Wei, « LinkForecast: Cellular Link Bandwidth Prediction in LTE Networks,» *IEEE Transactions on Mobile Computing*, pp. 1-1, 2017.

[27] J. Lee, S. Lee, J. Lee, S. D. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee y S. Ha, «PERCEIVE: deep learning-based cellular uplink prediction using real-time scheduling patterns,» *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys '20). Association for Computing Machinery*, pp. 377-390, 2020.

[28] C. Mediavilla Pastor, «Trabajo de Fin de Máster. Predicción de calidad de enlace en redes comunitarias inalámbricas basadas en OLSR a partir de datos obtenidos con una frecuencia alta de muestreo,» Universidad de Valladolid, Valladolid, 2019.

[29] B. Wei, W. Kawakami, K. Kanai, J. Katto y S. Wang, «TRUST: A TCP throughput prediction method in mobile networks,» *IEEE Global Communications Conference. GLOBECOM.*, pp. 1-6, 2018.

[30] M. Xie, P. Castagliola, C. Zou y Z. Wang, «ScienceDirect - Exponentially Weighted Moving Average,» 2009. [En línea]. Available: https://www.sciencedirect.com/topics/mathematics/exponentially-weighted-moving-average. [Último acceso: 29 04 2024].

[31] Corporate Finance Institute, «Exponentially Weighted Moving Average,» [En línea]. Available: https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/exponentially-weighted-moving-average-ewma/. [Último acceso: 29 04 2024].

[32] Corporate Finance Institute, «Harmonic Mean,» [En línea]. Available: https://corporatefinanceinstitute.com/resources/data-science/harmonic-mean/. [Último acceso: 29 04 2024].

[33] A. L. Samuel, «Some Studies in Machine Learning Using the Game of Checkers,» *IBM Journal of Research and Development,* vol. 3, nº 3, p. 210, 1959.

[34] T. Mitchell, Machine Learning, McGraw-Hill, 1997.

[35] D. Peteiro-Barral y B. Guijarro-Berdiñas, «A survey of methods for distributed machine learning,» *Progress in Artificial Intelligence,* vol. 2, nº 1, pp. 1-11, 2013.

[36] P. Lyman y H. R. Varian, How Much Information?, Berkeley: School of Information Management and Systems at the University of California, 2003.

[37] «Health IT. How big is Internet and how do we measure it?,» [En línea]. Available: https://healthit.com.au/how-big-is-the-internet-and-how-do-we-measure-it/. [Último acceso: 16 06 2024].

[38] «Aprendizaje supervisado y no supervisado,» Universidad Europea, 07 07 2022. [En línea]. Available: https://universidadeuropea.com/blog/aprendizaje-supervisado-no-supervisado/. [Último acceso: 23 04 2024].

[39] «Amazon Web Services - Machine Learning e IA,» [En línea]. Available: https://aws.amazon.com/es/what-is/neural-network/. [Último acceso: 19 04 2024].

[40] Colah, «Blog de Colah,» 27 08 2015. [En línea]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Último acceso: 19 04 2024].

[41] Y. Li, Z. Zhu, D. Kong, H. Han y Y. Zhao, «EA-LSTM: Evolutionary attention-based LSTM for time series prediction,» *Knowledge-Based Systems,* vol. 181, p. 104785, 2019.

[42] J. Hu y W. Zheng, «Transformation-gated LSTM: efficient capture of short-term mutation dependencies for multivariate time series prediction tasks,» *2019 International Joint Conference on Neural Networks (IJCNN),* pp. 1-8, 2019.

[43] A. T. Bell, F. Wang, M. Ouyang y Y. Wu, «Science Direct - Root Mean Square Error,» [En línea]. Available: https://www.sciencedirect.com/topics/engineering/root-mean-square-error. [Último acceso: 25 04 2024].

[44] Z. Xu, A. Van Donkelaar, Y. Wu y H. Lu, «ScienceDirect - Mean Absolute Error,» [En línea]. Available: https://www.sciencedirect.com/topics/engineering/mean-absolute-error. [Último acceso: 25 04 2024].

[45] E. Díez Benito, «Trabajo de Fin de Grado - Aplicación de técnicas de Deep Learning para mejorar el enrutamiento en redes comunitarias inalámbricas basadas en OLSR,» Universidad de Valladolid, Valladolid, 2020.

[46] Ó. Fernández, «Descubre Jupyter Notebook: La herramienta imprescindible para la ciencia de datos,» 13 12 2023. [En línea]. Available: https://aprenderbigdata.com/jupyter-notebook/. [Último acceso: 24 04 2024].

[47] «¿Qué es CUDA? NVIDIA Asistencia,» 18 12 2023. [En línea]. Available: https://support.nvidia.eu/hc/es/articles/4850516229266--Qu%C3%A9-es-CUDA. [Último acceso: 20 06 2024].

Glosario y acrónimos

3GPP. *3rd Generation Partnership Project*
AMPS. *Advanced Mobile Phone Service*
ANOVA. *Multi-factor Analysis Of Variance*
CDMA. *Code Division Multiple Access*, Acceso Múltiple por División en el Código
DASH. *Dynamic Adaptative Streaming Over HTTP*
DL. *Deep Learning*
EDGE. *Enhaced Data rates for Global Evolution*
eNB. *evolved NodeB*
EPC. *Red troncal*
E-UTRAN. *Red de Acceso*
EWMA. *Exponential Weighted Moving Average*
FDD. *Frequency Division Duplex*
FDMA. *Frequency Division Multiple Access*, Acceso Múltiple por División en Frecuencia
GPRS. *General Packet Radio System*
GSM. *Global System for Mobile Communications*
HM. *Harmonic Mean, Harmonic Mean*
HSDPA. *High Speed Downlink Packet Access*
HSS. *Home Subscriber Server*
HSUPA. *High Speed Uplink Packet Access*
ISIM. *IP Multimedia Services Identity Module*
LSTM. *Long Short Term Memory*
LTE. *Long Term Evolution*
MAE. *Mean Absolute Error*
ME. *Mobile Equipment*
MIMO. *Multiple Input Multiple Output*
MME. *Mobile Management Entity*
MT. *Mobile Terminal*
NMT. *Nordic Mobile Telephone*
NN. *Neural Networks*
OCS. *Online Charging System*
OFCS. *Offline Charging System*
OFDM. *Orthogonal Frequency Division Multiplex*
OFDMA. *Orthogonal Frequency Division Multiple Access*
PAPR. *Peak to Average Power Ratio*
PCRF. *Policy and Charging Rules Function*
P-GW. *Packet Data Network Gateway*
QoE. *Quality of Experience*

QoS. *Quality of Service*
RF. *Random Forest*
RL. *Recursive Least Square adaptative algorithm*
RLS. *Recursive Least Squared*
RMSE. *Root Mean Square Error*
RNN. *Recurrent Neural Networks*
RSRP. *Reference Signal Received Power, Reference Signal Received Power*
RSRQ. *Reference Signal Receiving Quality, Reference Signal Receiving Quality*
RSSI. *Received Signal Strength Indicator*
SC-FDMA. *Single Carrier FDMA, Single Carrier – Frequency Division Multiple Access*
S-GW. *Serving Gateway*
SIM. *Suscriber Identity Module*
SMS. *Short Message Service*
SNR. *Signal to Noise Ratio, Signal to Noise Ratio*
SoC. *System on Chip*
StemGNN. *Spectral Temporal Graph Neural Network*
SVR. *Support Vector Regression*
TACS. *Total Access Communications System*
TDD. *Time Division Duplex*
TDMA. *Time Division Multiple Access*
TD-SCDMA. *Time Division Synchronous CDMA*
TE. *Terminal Equipment*
UE. *Equipo de Usuario*
UICC. *Universal Integrated Circuit Card*
UIT. *Unión Internacional de Telecomunicaciones*
UMTS. *Universal Movile Telecommunications System*
USIM. *User Services Identity Module*
WAP. *Wireless Application Protocol*
WiFi. *Wireless Fidelity*