Universidad de Valladolid

Máster universitario

Ingeniería Informática

Trabajo Fin de Máster

Predicción de trayectorias aéreas con técnicas de *Deep Learning*

Realizado por **Paula Mielgo Martín**

✠✠✠

Universidad de Valladolid

17 de julio de 2023

Tutores: Aníbal Bregón Bregón
Jorge Silvestre Vilches

Resumen

En la actualidad el avión es el medio de transporte más utilizado para realizar desplazamientos de media y larga distancia. Durante los últimos años se ha presenciado un repunte del número de pasajeros, que había disminuido bruscamente tras la pandemia. Además, las estimaciones para periodos futuros auguran un aumento del número de vuelos que serán necesarios para satisfacer las necesidades de los usuarios, lo cual aumentará aún más la concentración y complejidad en la gestión del tráfico aéreo.

En este contexto se hace necesario el uso de herramientas que permitan anticipar el estado del espacio aéreo y facilitar la labor de los controladores. Debido al gran volumen de datos recogidos durante la travesía, las técnicas basadas en *machine learning* se presentan como la mejor opción para realizar este tipo de estimaciones. En particular, en el presente documento se propone el uso de técnicas de *deep learning* para la predicción de trayectorias aéreas. Para ello, se toman datos ADS-B de vuelos con destino el aeropuerto Adolfo Suárez Madrid-Barajas completados entre enero y septiembre de 2022, y se construyen modelos basados en redes neuronales recurrentes y *transformers*. En concreto, se utiliza la red Long Short-Term Memory y la arquitectura Temporal Fusion Transformer. Finalmente, se analizan y comparan los resultados obtenidos para concluir que los *transformers* mejoran el desempeño de las redes recurrentes.

Descriptores

Predicción de trayectorias, gestión del tráfico aéreo, Long Short-Term Memory, *transformer*.

Abstract

Nowadays, air travel is the most widely used means of transport for medium- and long-distance travel. In recent years, there has been a recovery in passenger numbers, which had fallen sharply during the pandemic. In addition, estimations predict an increase in the number of flights required to meet the needs of users, which will further rise the concentration and complexity of air traffic management.

In this context, it is necessary to use tools to anticipate the state of the airspace and to facilitate the work of air traffic controllers. Due to the large amount of data collected during a flight, techniques based on machine learning appear to be the best option to carry out this type of estimation. In particular, this work proposes the use of deep learning techniques for predicting flight trajectories. For this purpose, ADS-B data from flights arriving at Adolfo Suárez Madrid-Barajas between January and September 2022 are used to build models based on recurrent neural networks and transformers. Specifically, the Long Short-Term Memory network and the Temporal Fusion Transformer architecture are used. Finally, the results obtained are analysed and compared to conclude that transformers improve the performance of recurrent networks.

Keywords

Trajectory forecasting, air traffic management, Long Short-Term Memory, transformer.

Índice general

Índice de figuras

Índice de tablas

1: Introducción

El avión es una de las opciones preferidas hoy en día para realizar viajes de media y larga distancia debido a su velocidad, fiabilidad y comodidad. Sus características hacen que cada vez más usuarios se decanten por este medio. A pesar de que la tendencia creciente en el número anual de pasajeros se vio frenada y disminuida por la pandemia y por el actual conflicto bélico en Ucrania, las previsiones para los siguientes años auguran un repunte en este valor. En concreto, Eurocontrol (Organización Europea para la Seguridad de la Navegación Aérea) vaticina que, en el caso más probable, la recuperación del tráfico aéreo a los niveles obtenidos en 2019 se alcanzará en 2024 [23] (ver Figura 1.1).

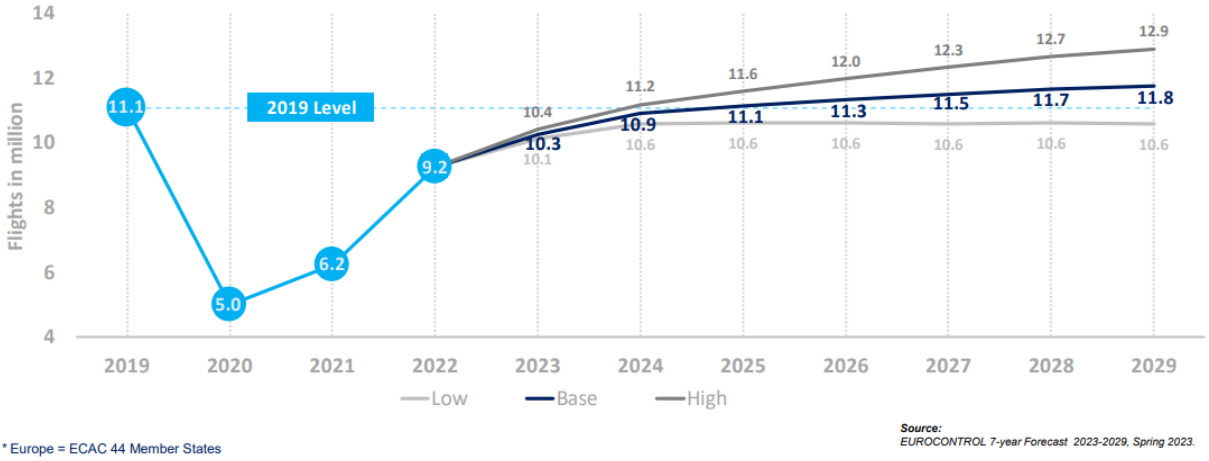

Figura 1.1: Previsión del número de vuelos anuales en Europa para los próximos años

Las previsiones para España son similares. Esto puede verse en la Figura 1.2, que ha sido construida con la información extraída del informe anual de tráfico en los aeropuertos españoles de 2021 [30] (último disponible a fecha de consulta) junto con el Documento de Regulación Aeroportuaria publicado en ese mismo año [21].

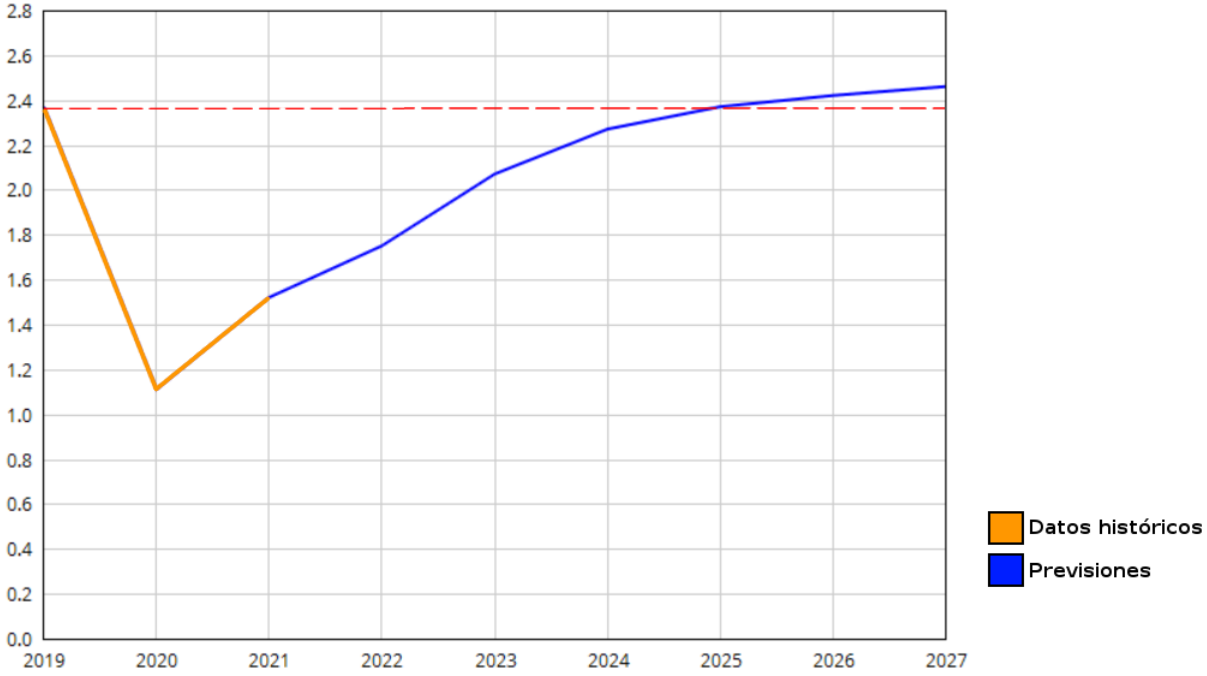

Figura 1.2: Previsión del número de vuelos en España para los años 2022 a 2027

Este incremento previsto en la demanda de vuelos diarios derivará en una mayor concentración y complejidad en la gestión del tráfico aéreo. Es por ello que, para garantizar la seguridad de las operaciones de las aeronaves en circulación, será necesario contar con sistemas basados en el uso de técnicas de Inteligencia Artificial que complementen la labor de los controladores aéreos. Estas pueden estar fundamentadas en modelos físicos que definan el movimiento de los aviones o en el uso de otro tipo de tecnologías basadas en datos. Dentro de esta última categoría destaca el uso de algoritmos de *machine learning* [36, 48], y más concretamente, de arquitecturas de *deep learning* [39].

Alineado con lo expuesto, en el presente Trabajo Fin de Máster nos centraremos en la construcción de un sistema de predicción de trayectorias de vuelo utilizando redes recurrentes y *transformers*, que podría ser útil para estimar el estado de espacio aéreo futuro y así organizar de manera más eficiente las operaciones aeroportuarias, como puede ser la configuración de pistas. Además, también ayudaría a reducir los riesgos de conflictos entre aeronaves, a mejorar la predicción de tiempos de llegada y a disminuir la huella de carbono.

1.1. *Problem statement*

En el contexto mencionado sería ideal conocer los próximos puntos que recorrerá la aeronave a lo largo del vuelo, incluyendo posibles desviaciones causadas por condiciones climáticas adversas, problemas técnicos o restricciones impuestas por autoridades de aviación, de forma que los controladores de tráfico aéreo pudiesen predecir el estado futuro del espacio aéreo e informar a pilotos con vuelos en circulación próxima o actual, optimizando así el tráfico. Se alcanzaría así un estado en el que se podrían planificar los recursos y maniobras necesarias para garantizar la seguridad y eficiencia de las operaciones.

Sin embargo, actualmente las decisiones se basan en la información proporcionada por los radares (el primario [12] y el secundario [14]), que presentan una serie de limitaciones que pueden incidir sobre su precisión. Además, también se cuenta con las comunicaciones establecidas con los controladores aéreos de la torre de control asociada, pero esto puede impactar negativamente sobre la coordinación entre aeronaves debido al error humano causado por la presión a la que se ven sometidos por la enorme responsabilidad del trabajo o al cansancio ocasionado por las largas jornadas. De esta manera, los factores mencionados presentan algunas limitaciones en términos de seguridad, uso eficiente de los recursos como el tiempo o el combustible y un aumento de la huella de carbono.

Con la idea de sustituir el radar secundario surge la tecnología ADS-B (Automatic Dependent Surveillance – Broadcast), que es un sistema que envía de manera periódica y automática información relativa a la situación actual de la aeronave. Los datos de posición se obtienen mediante la comunicación con satélites, por lo que no es necesario que la nave se encuentre próxima a antenas terrestres. Este concepto se desarrollará con más detalle en el Capítulo 3.

La propuesta planteada en esta memoria pretende predecir los próximos puntos en la trayectoria de una aeronave, de forma que, tanto pilotos como controladores, puedan estimar el estado del espacio aéreo en instantes futuros y tomar decisiones en consecuencia. Para ello, se utilizarán datos ADS-B de vuelos con destino el aeropuerto Adolfo Suárez-Madrid Barajas entre enero y septiembre de 2022, así como otros *datasets* con información relativa al aeropuerto y aerolíneas que operan en los diferentes trayectos. Las arquitecturas elegidas para procesar estos conjuntos serán una red neuronal recurrente (en concreto Long Short Term Memory, LSTM [32]) y un *transformer* (Temporal Fusion Transformer, TFT [34]).

1.2. Objetivos del trabajo

El objetivo principal del proyecto será el siguiente:

OBJ-1: Construir un modelo basado en técnicas *deep learning* para la predicción de trayectorias de una aeronave.

Dentro de este objetivo global pueden identificarse algunos objetivos específicos:

- **OBJ-1.1**: Realizar una revisión de las soluciones propuestas hasta la fecha para resolver el problema de predicción de trayectorias aéreas.
- **OBJ-1.2**: Proponer un modelo LSTM para la predicción de trayectorias.
- **OBJ-1.3**: Proponer un modelo TFT para la predicción de trayectorias.
- **OBJ-1.4**: Evaluar y comparar los modelos implementados.

Restricciones

Por otro lado, las restricciones identificadas son:

- **REST-1**: Volumen de datos disponibles. Como se ha mencionado anteriormente, los datos a utilizar han sido tomados en el aeropuerto Adolfo Suárez-Madrid Barajas entre enero y septiembre de 2022.
- **REST-2**: El alcance y duración del proyecto deben limitarse a la carga de trabajo establecida por la guía docente de la asignatura Trabajo Fin de Máster (150 horas) junto con la asociada a la beca de colaboración con departamentos del Consejo Social (210 horas).

1.3. Estructura de la memoria

Los capítulos en que se divide la memoria son los siguientes:

- **Capítulo 1. Introducción.** En el que se introduce el proyecto en el contexto actual y se definen los objetivos y restricciones.
- **Capítulo 2. Planificación.** Se enumeran las tareas que se pretenden desarrollar y se planifican en el espacio temporal. Además, se estiman los costes derivados de su consecución.
- **Capítulo 3. Contexto de trabajo.** En el que se introducen conceptos básicos del entorno específico de la gestión del tráfico aéreo, así como términos básicos dentro de la Inteligencia Artificial.
- **Capítulo 4. Estado del arte.** Se presentan las principales propuestas existentes hasta la fecha para la resolución de problemas de características similares.
- **Capítulo 5. Diseño experimental.** Presenta el conjunto de datos y las arquitecturas con las que se pretende trabajar. También detalla la métrica elegida para la experimentación.

- **Capítulo 6. Experimentación.** En el que se exponen detalladamente los diferentes entrenamientos ejecutados.
- **Capítulo 7. Evaluación.** Presenta la evaluación final sobre el conjunto de *test* de los modelos seleccionados durante la experimentación.
- **Capítulo 8. Conclusiones y líneas de trabajo futuras.** Reflexiones finales sobre el trabajo realizado y propuesta de algunas líneas de mejora futura.
- **Apéndice A. Manual de instalación.** Breve manual de instalación de un entorno de Anaconda.
- **Apéndice B. Contenido adjunto.** Material que se entrega junto con la memoria.

2: Planificación

2.1. Metodología

La metodología de trabajo elegida para el desarrollo del proyecto es ASAP (*Agile Student Academic Projects*) [37] que es una propuesta metodológica que surge como proyecto de innovación docente en la Universidad de Valladolid en 2018. El objetivo de esta propuesta es la aplicación de los principios del Manifiesto Ágil [25] y de las diversas prácticas que de ellos se derivan en el contexto académico universitario. En particular, ASAP organiza los procesos de enseñanza-aprendizaje que se llevan a cabo durante el desarrollo de los Trabajos Fin de Estudios (tanto de Grado, como de Máster), en los que el producto a desarrollar consiste, principalmente, en una memoria técnica de proyecto, un producto software que satisfaga los requisitos del proyecto, y una defensa final del proyecto ante el tribunal de evaluación. Para ello, se definen una serie de roles, eventos y artefactos que permiten que los participantes puedan mantener interacción entre sí de forma regular, que el ritmo de trabajo sea constante y que se genere *feedback* de forma frecuente. Así, el proceso de aprendizaje por parte del alumno se conforma como un proyecto construido iterativa e incrementalmente a lo largo de varios *sprints* de aprendizaje.

Aunque no se pretende hacer una descripción detallada de la metodología ASAP (que puede consultarse en [37]), se mencionarán brevemente algunos conceptos básicos que se pondrán en práctica durante la realización del presente Trabajo Fin de Máster.

Roles

Dentro de ASAP se consideran roles para todas las personas que, de una forma u otra, intervienen en el desarrollo del proyecto.

- Estudiante. Es el rol principal puesto que es el encargado de construir un producto de calidad. Para ello, debe acudir a todos los eventos establecidos en la metodología, así como definir y planificar las tareas necesarias para satisfacer los objetivos planteados en cada *sprint*. Por supuesto, también debe completar dichas tareas según los criterios establecidos y comunicar en el espacio de trabajo compartido el estado actualizado del proyecto.

- Tutor. Desempeñado por uno o más profesores que tutorizan el Trabajo Fin de Máster. El tutor es responsable, en primer lugar, de proponer un proyecto al estudiante, acorde con los requerimientos definidos en la guía docente de la asignatura para, posteriormente, definir qué objetivos del mismo se abordarán al inicio de cada *sprint*. Por otro lado, debe asegurarse de que la realización de los diferentes eventos se completa de manera satisfactoria, así como proporcionar *feedback* de manera regular, resolviendo también si fuese necesario, posibles dudas o bloqueos que experimente el estudiante.

- Comunidad. Formada por cualquier persona capaz de aportar valor al producto desarrollado. La comunidad debe asistir a los eventos de comunicación de progresos y a la retrospectiva, en los que puede aportar *feedback* complementario orientado a la mejora del proyecto tanto en términos de producto como de proceso.

- Tribunal. Formado por los profesores designados a la evaluación del Trabajo Fin de Máster. El tribunal es responsable de evaluar objetivamente el producto entregado, así como el acto de defensa de acuerdo a los criterios establecidos en el reglamento asociado a la asignatura.

Eventos

ASAP define un total de 5 eventos que se describen a continuación.

- *Sprint*. Son divisiones temporales del proyecto. En cada *sprint*, se aborda un subconjunto de los objetivos del proyecto, eligiendo aquellos que presenten una mayor prioridad de acuerdo al estado del mismo. La división en *sprints* permite planificar a corto plazo, reduciendo la incertidumbre y facilitando la revisión frecuente del producto construido. Deben ser de duración inferior a un mes.

- Reunión de inicio. Es el primer evento de cada *sprint* y en él participan el estudiante y el tutor. Durante este encuentro debe definirse el objetivo del *sprint* teniendo en cuenta el *feedback* recibido en el *sprint* anterior. También deben definirse y planificarse en este momento las tareas necesarias para su consecución.

- Reunión de sincronización. Es una reunión semanal en la que, nuevamente, participan estudiante y tutor. Durante la misma, el estudiante expone los avances obtenidos durante la semana, así como los bloqueos o dificultades experimentadas. Finalmente, informa de qué tareas pretende desarrollar en el futuro cercano. De esta forma, se consigue mantener una comunicación constante a lo largo del *sprint*.

- Comunicación de progresos. Es una reunión que se realiza al final del *sprint*, en la que participan el estudiante, el tutor y la comunidad. En ella, cada estudiante expone, de manera similar a como sería el acto de defensa, el trabajo realizado hasta ese momento. Tras esto, el resto de participantes aporta comentarios y sugerencias de mejora con respecto al producto durante un turno de debate.

- Retrospectiva. Es la última reunión del *sprint*, realizada tras la comunicación de progresos. De nuevo se cuenta con los mismos participantes para reflexionar sobre la dinámica de trabajo seguida durante el *sprint*. De esta forma, mediante una puesta en común de las opiniones individuales, se generan valoraciones positivas y negativas sobre el mismo y se aportan sugerencias de mejora.

Artefactos

Concluiremos la descripción de la metodología introduciendo los dos artefactos propuestos por ASAP.

- Incremento. Se define como el estado del producto en el momento actual.
- Retroalimentación. También denominado *feedback*, consiste en la valoración que recibe el estudiante al finalizar cada *sprint*, tanto por parte de la comunidad en el evento de comunicación de progresos, como la emitida por parte del tutor una vez revisado el incremento entregado al final de la iteración.

2.2. Planificación temporal

El presente proyecto se ha realizado, en parte, como trabajo de investigación asociada a las becas de colaboración con departamentos convocadas por el Consejo Social de la Universidad de Valladolid. Es por ello que, a la hora de definir el alcance se tuvo en cuenta, no solo la carga definida para el Trabajo Fin de Master (150 horas), sino también la exigida por el mencionado contrato (210 horas). Por otro lado, a diferencia del resto de compañeros de curso en el Máster, las prácticas curriculares realizadas durante el mes de junio no guardaron relación alguna con este trabajo.

De esta forma, estableciendo el inicio del proyecto el día 9 de enero de 2023 y el cierre el 10 de julio, la carga horaria que debería realizarse de forma diaria (de lunes a viernes) para ajustarse a los requerimientos temporales sería de, aproximadamente 3 horas. Partiremos de este hecho para realizar la planificación del proyecto.

Puesto que el tiempo de realización del proyecto se extiende durante seis meses y teniendo en cuenta los principios de la metodología elegida, que recomiendan iteraciones cortas, se planificarán seis *sprints* de un mes de duración. Se ha establecido la siguiente asignación de tareas (relacionadas con las historias de proyecto identificadas) a los diferentes *sprints* de aprendizaje planificados:

Sprint #1

- **H1. Entorno de negocio**
  - T1.1. Introducción al entorno de negocio

- **H2. Contexto científico-técnico**
  - T2.1. Conceptos generales de Data Science
  - T2.2. Redes recurrentes
  - T2.3. Transformers
  - T2.4. Herramientas y tecnologías
- **H3. Memoria**
  - T3.1. Redacción del apartado de Planificación
  - T3.2. Redacción del apartado de Introducción y objetivos
  - T3.3. Bibliografía
- **H4. Presentación Sprint**
  - T4.1. Presentación del *sprint* #1

Sprint #2

- **H5. Análisis**
  - T5.1. Descripción de los datos
  - T5.2. Requisitos
- **H6. Estado del arte**
  - T6.1. Búsqueda de referencias
  - T6.2. Lectura y descripción de propuestas
- **H3. Memoria**
  - T3.4. Corrección del *sprint* #1
  - T3.5. Redacción del apartado de Análisis
  - T3.6. Redacción del estado del arte
  - T3.7. Bibliografía
- **H4. Presentación Sprint**
  - T4.2. Modificación parte *sprint* #1
  - T4.3. Presentación parte *sprint* #2

Sprint #3

- **H7. Implementación de arquitecturas**
  - T7.1. Implementación de las arquitecturas LSTM
- **H8. Experimentación con LSTM**
  - T8.1. Exploración inicial
  - T8.2. Ajuste de hiperparámetros
  - T8.3. Inclusión de características
- **H3. Memoria**
  - T3.8. Corrección *sprint* #2
  - T3.9. Redacción inicial del apartado de Experimentación
  - T3.10. Bibliografía
- **H4. Presentación Sprint**
  - T4.4. Modificación parte *sprint* #2
  - T4.5. Presentación parte *sprint* #3

Sprint #4

- **H8. Experimentación con LSTM**
  - T8.4. LSTM con capas densas
  - T8.5. Análisis de resultados
- **H3. Memoria**
  - T3.11. Corrección *sprint* #3
  - T3.12. Avance en la redacción del apartado de Experimentación
  - T3.13. Bibliografía
- **H4. Presentación Sprint**
  - T4.6. Modificación parte *sprint* #3
  - T4.7. Presentación parte *sprint* #4

Sprint #5

- **H7. Implementación de arquitecturas**
  - T7.2. Implementación de la arquitectura TFT
- **H9. Experimentación con TFT**
  - T9.1. Exploración inicial
- **H3. Memoria**
  - T3.14. Corrección *sprint* #5
  - T3.15. Avance en la redacción del apartado de Experimentación
  - T3.16. Bibliografía
- **H4. Presentación Sprint**
  - T4.8. Modificación parte *sprint* #4
  - T4.9. Presentación parte *sprint* #5

Sprint #6

- **H9. Experimentación con TFT**
  - T9.2. Ajuste de hiperparámetros
  - T9.3. Inclusión de características
  - T9.4. Análisis de resultados
- **H3. Memoria**
  - T3.17. Corrección *sprint* #5
  - T3.18. Completar el apartado de Experimentación
  - T3.19. Redacción de Resultados y Conclusiones
  - T3.20. Bibliografía
  - T3.21. Correcciones finales
- **H4. Presentación Sprint**
  - T4.10. Modificación parte *sprint* #5
  - T4.11. Presentación parte *sprint* #6

Diagrama de Gantt

Para concluir con la planificación temporal, se muestra en las Figuras 2.3 y 2.4 el diagrama de Gantt asociado a las tareas identificadas. La semana correspondiente a Semana Santa (semana 13) no se incluye en la planificación por no ser lectiva. Debido a la duración prolongada y bloqueante que presentan algunas tareas, como los entrenamientos de modelos, será necesario paralelizar tareas en función de la disponibilidad de recursos. Como esta dinámica no es fácilmente predecible, se plantea una planificación cuya unidad básica es la semana, indicando qué tareas se abordarán en cada una, sea de forma secuencial, concurrente o alternativa.

Tarea	Story Points	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Sprint #1	**29**													
Introducción al entorno de negocio	3													
Conceptos generales de Data Science	5													
Descripción de redes recurrentes	4													
Descripción de transformers	7													
Herramientas y tecnologías	1													
Redacción apartado Planificación	3													
Redacción del apartado de Introducción y objetivos	3													
Bibliografía	1													
Presentación del sprint #1	2													
Sprint #2	**26**													
Correcciones sprint #1	2													
Descripción de los datos	2													
Requisitos	4													
Búsqueda de referencias	1													
Lectura y descripción de propuestas	7													
Redacción del apartado de Análisis	2													
Redacción del Estado del arte	4													
Bibliografía	1													
Modificación presentación sprint #1	1													
Presentación parte sprint #2	2													
Sprint #3	**26**													
Correcciones sprint #2	1													
Implementación de arquitecturas LSTM	2													
Exploración inicial LSTM	3													
Ajuste de hiperparámetros LSTM	6													
Inclusión de características LSTM	7													
Redacción inicial del apartado de Experimentación	3													
Bibliografía	1													
Modificación presentación sprint #2	1													
Presentación parte sprint #3	2													

Figura 2.3: Diagrama de Gantt: *sprints* 1, 2 y 3

Tarea	Story Points	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26
Sprint #4	**28**													
Correcciones sprint #3	2													
LSTM con capas densas	11													
Análisis de resultados LSTM	4													
Avance en la redacción del apartado Experimentación	6													
Bibliografía	1													
Modificación presentación sprint #3	2													
Presentación parte sprint #4	2													
Sprint #5	**26**													
Correcciones sprint #4	1													
Implementación de la arquitectura TFT	9													
Exploración inicial TFT	7													
Avance en la redacción del apartado Experimentación	4													
Bibliografía	1													
Modificación presentación sprint #4	2													
Presentación parte sprint #5	2													
Sprint #6	**32**													
Correcciones sprint #5	2													
Ajuste de hiperparámetros TFT	7													
Inclusión de características TFT	7													
Análisis de resultados TFT	5													
Completar apartado de Experimentación	3													
Redacción de Resultados y Conclusiones	4													
Bibliografía	1													
Modificación presentación sprint #5	1													
Presentación parte sprint #6	2													

Figura 2.4: Diagrama de Gantt: *sprints* 4, 5 y 6

2.3. Presupuesto

El presupuesto asociado al proyecto se divide en dos apartados: recursos técnicos y recursos humanos. Por un lado, en los recursos técnicos se considerarán aquellos sistemas, herramientas y objetos necesarios para la realización del proyecto. Por otro lado, dentro de los recursos humanos se considerará el coste asociado a los trabajadores.

Recursos técnicos

Para la realización del proyecto se cuenta tres máquinas distintas, que se utilizarán según las necesidades concretas de cada etapa.

- Ordenador personal con un procesador Intel Core i7-7700HQ y 16GB de RAM.
- Máquina virtual del departamento con características similares.
- Máquina virtual del departamento con GPU Nvidia RTX A40.

El coste del ordenador personal es de, aproximadamente, 1.100€, con una vida útil de unos 5 años. Por otro lado, dado que no se conocen las características particulares de la primera máquina virtual, su precio se estimará también en 1.100€. A esto debe sumarse el precio de la máquina con GPU, unos 11.200€, con una vida útil de 5 años.

Será imprescindible contar con conexión a Internet para consultar la diferente bibliografía seleccionada, así como para poder llevar a cabo las reuniones de sincronización

establecidas en la metodología seleccionada. Para ello, se contratarán 300Mb de fibra simétrica por un precio de 30€ mensuales.

En cuanto a las herramientas seleccionadas, se utilizarán aplicaciones de licencia gratuita, como Weights & Biases, Microsoft Teams, Overleaf, Anaconda o Jupyter Notebook. Los sistemas operativos tampoco han supuesto un incremento del presupuesto puesto que venían incluidos con el equipo (Windows 10) o son de distribución libre (Ubuntu).

Los recursos técnicos mencionados junto con su coste asociado pueden consultarse en la Figura 2.5.

	Coste	Porcentaje de uso mensual	Meses	Total
Ordenador personal	1.100,00 €	0,54%	7	41,58 €
Máquina virtual 1	1.100,00 €	0,54%	7	41,58 €
Máquina virtual 2	1.100,00 €	0,54%	7	41,58 €
Nvidia RTX A40	11.200,00 €	0,54%	7	423,36 €
Internet	30,00 €	100%	7	210,00 €
Weights & Biases	0,00 €			0,00 €
Microsoft Teams	0,00 €			0,00 €
Overleaf	0,00 €			0,00 €
Anaconda	0,00 €			0,00 €
Jupyter Notebook	0,00 €			0,00 €
Windows 10	0,00 €			0,00 €
Ubuntu	0,00 €			0,00 €
				758,10 €

Figura 2.5: Recursos técnicos

Recursos humanos

El primer perfil necesario para el desarrollo proyecto es un analista, cuyo salario medio en España ronda los 31.650€ anuales [15]. Esto se traduce, suponiendo 40 horas semanales y prorrateándolo en 12 pagas, a unos 16,48€ por hora. Por otro lado, también se requerirá la contratación de un *data scientist*, con un salario aproximado de 39.250€ anuales [16], 20,44€ por hora. Finalmente, será necesario también contar con un programador, cuyo salario medio es de 28.150€ anuales [17], que equivale a unos 14,66€ por hora. Además, también deben considerarse los gastos que deben abonarse a la Seguridad Social debido a la contratación de los trabajadores, que supone aproximadamente el 30 % del salario bruto [19].

La estimación del número de horas requeridas para cada uno de los perfiles junto con el correspondiente coste económico asociado puede consultarse en la Figura 2.6.

	Salario por hora	Horas totales	Total sin SS	Coste SS	Total
Analista	16,48 €	120	1.977,60 €	593,28 €	2.570,88 €
Data scientist	20,44 €	140	2.861,60 €	858,48 €	3.720,08 €
Desarrollador	14,66 €	100	1.466,00 €	439,80 €	1.905,80 €
					8.196,76 €

Figura 2.6: Recursos humanos

Presupuesto final

En vista a lo indicado en las Figuras 2.5 y 2.6, el presupuesto final obtenido es de 8.954,86€.

3: Contexto del Trabajo

3.1. Entorno de negocio

En esta sección se presenta una introducción a la gestión del tráfico aéreo, desarrollando algunos conceptos como el de trayectoria 4D, la tecnología ADS-B o una descripción del aeropuerto en que se centra el proyecto, el aeropuerto Adolfo Suárez Madrid-Barajas. El primer concepto a tener en cuenta es el de gestión del tráfico aéreo o *Air Traffic Management* (ATM), que se define, según el Reglamento 549/2004 del marco SES [20], como el conjunto de las funciones, tanto a bordo como en tierra, necesarias para garantizar el movimiento seguro y eficiente de las aeronaves durante todas las fases de las operaciones. Dentro de éstas, destaca el control del tráfico aéreo o *Air Traffic Control* (ATC), que es el servicio que ejercen los controladores aéreos emitiendo autorizaciones o restricciones sobre las aeronaves en función del tránsito y de las condiciones del entorno [4]. Para ello, el espacio aéreo se divide en las denominadas zonas de gestión ATC, que son regiones asociadas a un grupo de controladores y que permiten acotar y limitar espacialmente su franja de actuación.

Es de vital importancia para el correcto control del tráfico aéreo la recopilación y procesamiento de los datos de posición de las aeronaves que circulan por cada una de las zonas ATC. Típicamente estos son altitud, longitud y latitud. No obstante, es necesario añadir también una dimensión temporal para poder medir posibles retrasos con respecto a la planificación inicial. Surge así el concepto de trayectoria 4D.

De esta forma, podemos definir una trayectoria 4D como una sucesión de puntos recorridos por una aeronave, incluyendo los correspondientes al despegue y aterrizaje, compuestos por datos relativos a la altitud, longitud, latitud y tiempo.

ADS-B

Tradicionalmente, la información, tanto de posición como otros datos complementarios, se han transmitido con el uso de los radares primario y secundario, que se basan en la emision y recepción de ondas electromagnéticas. Sin embargo, su funcionamiento presenta

algunos problemas en zonas amplias en las que no existen antenas receptoras. De esta forma, cuando la aeronave sobrevuela mares y océanos, los radares pierden su utilidad. Y no solo experimenta dificultades en estas zonas geográficas, sino que en zonas montañosas también se presentan problemas similares. Además, estos sistemas poseen un tiempo de retardo de hasta 12 segundos, algo altamente considerable teniendo en cuenta la velocidad a la que se desplazan las aeronaves. Es por ello, que surge la necesidad de utilizar sistemas alternativos que solventen los inconvenientes descritos. En concreto, se introducirá el concepto de ADS-B, que soluciona algunos de los problemas planteados, reemplazando a los radares secundarios con la finalidad de mejorar tanto eficiencia como seguridad en un entorno con mayor capacidad de tráfico aéreo.

La tecnología ADS-B, *Automatic Dependant Surveillance Broadcasting*, consiste en un sistema que permite a las aeronaves comunicar, de forma automática y periódica, información asociada a la trayectoria que está realizando, de forma que otras aeronaves o centros de control equipados con receptores compatibles con esta tecnología puedan recogerlos. Lo reseñable de ADS-B es que determina la posición de una aeronave con el uso de satélites (ver Figura 3.7), ya que obtiene esta información directamente de los sistemas de posicionamiento GPS del avión, de forma que los mensajes se transmiten de forma más fiable y rápida. Por otro lado, aunque ADS-B sigue presentando problemas de comunicación con tierra al atravesar zonas oceánicas, en las que aún no es posible desplegar receptores de esta tecnología, proporciona una aplicación denominada ITP [1], que permite a las aeronaves realizar cambios de nivel en esos tramos donde la comunicación es muy complicada para, por ejemplo, evitar zonas de vuelo turbulentas.

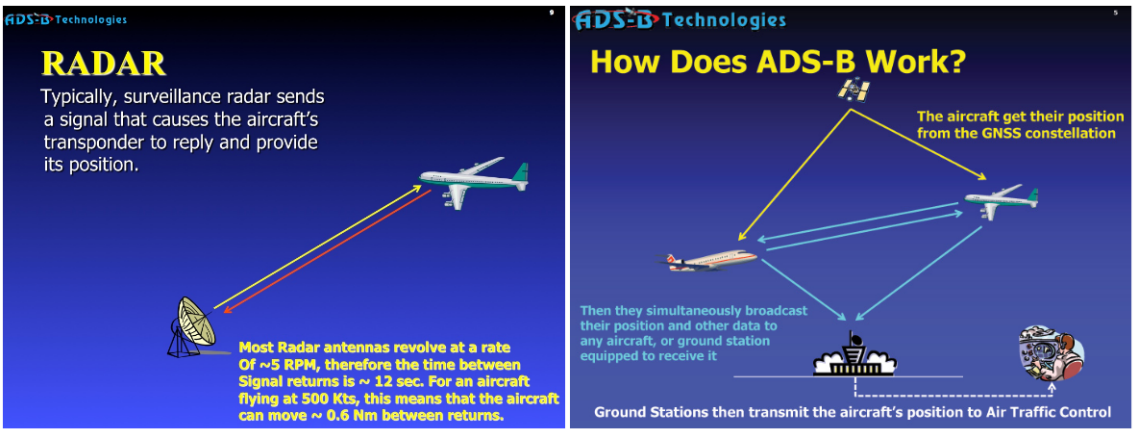

Figura 3.7: Comparativa de radar y ADS-B, extraída de [2]

Entre las ventajas que plantea el uso de este tipo de sistemas destaca un aumento en la precisión de los datos y el hecho de que estos se obtengan con una mayor tasa de actualización (retardo de 1 o 2 segundos). De esta forma, se puede mejorar la eficiencia en la gestión del tráfico aéreo pudiéndose aumentar la capacidad de tráfico del mismo. Por otro lado, los receptores utilizados para la recopilación de mensajes ADS-B son más económicos

que los empleados en el caso de los radares y, además, mejoran el funcionamiento en zonas montañosas por su amplio rango de cobertura.

Como es de esperar, la recopilación y procesamiento de estos datos podría complementar la información que poseen en la actualidad los controladores aéreos para mejorar la gestión del tráfico. Es por ello que surgen redes de receptores destinadas a la recopilación y almacenamiento de mensajes ADS-B, entre las que destacan OpenSky Network [1], que es la fuente de datos que utilizaremos, o ADSBHub.

Aeropuerto Adolfo Suárez Madrid-Barajas

Como se ha mencionado al inicio del capítulo, los datos que se utilizarán durante todo el proyecto se corresponden con mensajes ADS-B asociados a vuelos con destino en Adolfo Suárez Madrid-Barajas, que es el aeropuerto más transitado de España. La base aérea, que cuenta con casi 2000 hectáreas de superficie, dispone (como indica su web [3]) de cuatro terminales (denominadas T1, T2, T3 y T4), una Terminal Ejecutiva, un centro de carga aérea y dos zonas de hangares. Por otro lado, cuenta con cuatro pistas paralelas dos a dos donde se realizan despegues y aterrizajes. El sentido en que se usan las pistas viene condicionado por factores climatológicos y de seguridad, pero por lo general se opta por una configuración norte (de sur a norte), como puede visualizarse en la Figura 3.8, para minimizar la contaminación acústica.

Figura 3.8: Esquema de las posibles configuraciones de pistas, extraído de [22]

[1] https://opensky-network.org

3.2. Contexto científico-técnico

Machine Learning

Se conoce como *machine learning* o aprendizaje automático [28] al campo de la Inteligencia Artificial que se centra en el diseño y desarrollo de modelos que permitan a los computadores, en base a la experiencia, mejorar su desempeño a la hora de realizar una tarea concreta. Para ello, deben proporcionarse datos de ejemplo, a partir de los cuales, los algoritmos de aprendizaje consigan extraer conocimiento para, posteriormente, poder inferir nuevas respuestas. Ese proceso de aprendizaje se conoce como entrenamiento del modelo.

De esta forma y, a diferencia de la programación convencional, no es necesario definir las reglas del algoritmo, sino que es capaz de "programarse a sí mismo" mediante la observación de ejemplos (muestras de datos) relacionados con la tarea objetivo. Sin embargo, es importante tener en cuenta, que el uso de este tipo de soluciones no son siempre la mejor opción ya que pueden derivar en modelos con menor eficiencia y eficacia en comparación con soluciones tradicionales. Por ello, es muy importante conocer en profundidad el problema que se pretende abordar, para poder determinar correctamente el tipo de solución que más se adecúe a los requerimientos del mismo.

Los algoritmos basados en *machine learning* pueden clasificarse en dos categorías principales:

- Aprendizaje supervisado, en el que los modelos se entrenan con ejemplos etiquetados, es decir, datos de los cuales se conoce el resultado o solución. De esta forma, no solo se cuenta con la entrada o *input* del algoritmo, sino que también se posee la salida correcta que debería devolver. Es por ello, que el objetivo principal de esta categoría de Aprendizaje Automático es la predicción a partir de ejemplos desconocidos para el modelo. Entre las técnicas de aprendizaje supervisado utilizadas a día de hoy destacan las regresiones lineales, las máquinas de vectores de soporte, los árboles de decisión o las redes neuronales.
- Aprendizaje no supervisado, donde los modelos se entrenan con datos de ejemplo sin etiquetar. En este caso, los algoritmos se centran en el reconocimiento de patrones o asociaciones entre elementos en base a sus similitudes o diferencias. Dentro de esta categoría podemos discernir dos tipos de técnicas diferentes. Por un lado, se encuentran las orientadas a *clustering*, centradas en buscar agrupaciones en los datos, entre las que destacan algoritmos como k-medias, k-medianas o mapas autoorganizados. Por otro lado, existen otras técnicas orientadas a la reducción de dimensionalidad, como puede ser el análisis de componentes principales.

Redes neuronales

Con el objetivo de imitar los procesos biológicos del cerebro humano, que permiten resolver infinidad de problemas de manera casi instantánea (por ejemplo reconocer que un modelo nuevo de coche, que no se haya visto antes, es, efectivamente, un coche y no otra cosa), surgen las redes neuronales, que se enmarcan dentro del *machine learning*. De esta manera, se busca simular la capacidad de reconocer y extraer patrones sobre información presente en el entorno para, posteriormente, generalizar y poder realizar nuevas predicciones.

Se define una red neuronal artificial [46] como un modelo matemático que busca imitar el comportamiento y estructura de las neuronas biológicas en el cerebro humano. Para entender el funcionamiento de este tipo de modelos es necesario definir primero la unidad básica del sistema: la neurona artificial. Como es de esperar, estos elementos también guardan una cierta similitud con las diferentes partes que conforman las células cerebrales (ver Figura 3.9):

- Dendritas. Son las ramificaciones que reciben la información procedente de otra neurona.
- Cuerpo celular o soma. Es la parte central de la célula. Contiene el núcleo y se encarga de procesar la información recibida.
- Axón. Es la prolongación del cuerpo celular y su función es transportar la información procesada a otras neuronas.

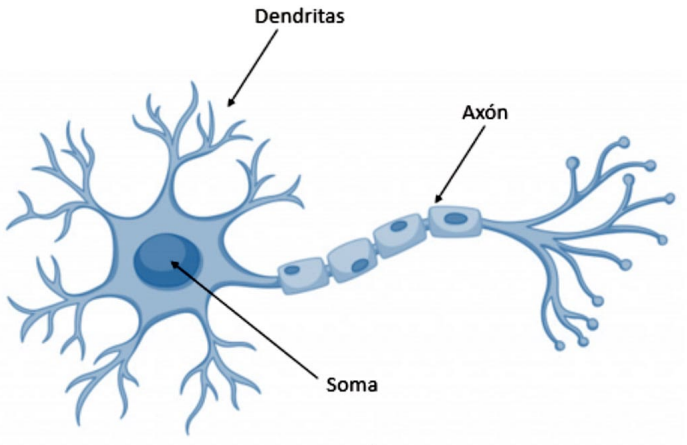

Figura 3.9: Partes de la neurona biológica, extraída de [13]

Aunque existen varios tipos de neuronas artificiales, se describirá en esta sección la versión clásica, el Perceptrón (ver Figura 3.10), que se compone de las siguientes partes:

- Un conjunto de elementos de entrada $x_1, x_2, ..., x_m$ procedentes de otra neurona o del exterior.
- Un conjunto de pesos $w_1, w_2, ..., w_m$ destinados a ponderar los elementos de entrada: $w_1 \cdot x_1$, $w_2 \cdot x_2$, ..., $w_m \cdot x_m$.
- Un sesgo o *bias* b, que es un valor negativo que regula la predisposición de la neurona para devolver como salida un valor u otro. Es decir, para poder contrarrestar un sesgo bajo se requerirán entradas y pesos altos de forma que, en las operaciones posteriores realizadas en la neurona, la suma ponderada de las entradas junto con el sesgo devuelvan un valor positivo.
- Regla de propagación $\sum_{i=1}^{m}(x_i \cdot w_i) + b$, encargada de calcular la diferencia entre el sesgo y la suma de los valores de entrada ponderados.
- Función de activación escalón, utilizada para limitar la amplitud de la salida de una neurona (a 0 o 1) en función de si la regla de propagación devuelve un valor positivo o negativo.
- Salida y. Es el valor devuelto por la función de activación, que se podrá ser utilizado como entrada de otras neuronas.

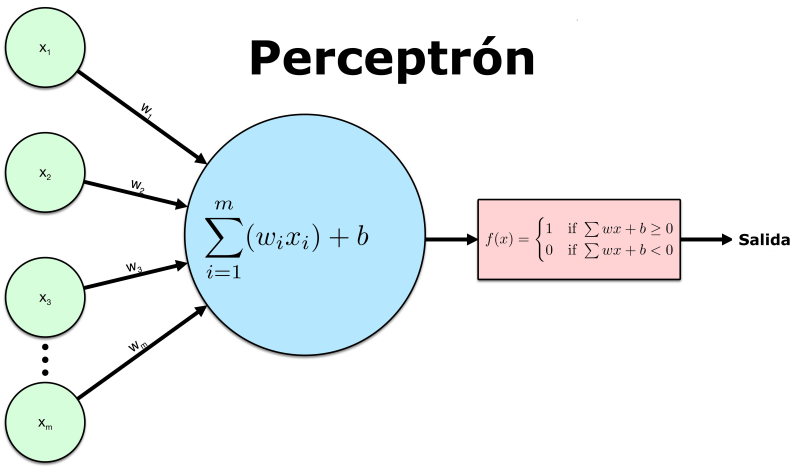

Figura 3.10: Partes de la neurona artificial, extraída de [7]

De esta forma se puede observar una cierta similitud entre algunos elementos de las neuronas artificiales y biológicas, como por ejemplo entre las dendritas y los elementos de entrada, entre el soma y la función de activación o entre la salida y el axón.

Visto esto, es importante mencionar que el perceptrón se puede modificar para obtener otros tipos de neuronas artificiales con el fin de abordar tareas diferentes. En particular, esta adaptación consistiría en cambiar la regla de propagación o la función de activación a utilizar.

Una vez definida la unidad básica de procesamiento de estas arquitecturas, se puede definir una red neuronal como un conjunto de neuronas artificiales distribuidas en diferentes capas e interconectadas entre sí. Se pueden diferenciar tres tipos de capas (ver Figura 3.11) según la procedencia/destino de las entradas/salidas que procesan las neuronas contenidas en la misma.

- Capa de entrada, que recibe sus entradas desde el exterior del sistema.
- Capa oculta, que recibe sus entradas desde las neuronas de una capa y envía sus salidas a otra capa.
- Capa de salida, que envía sus salidas al exterior del sistema.

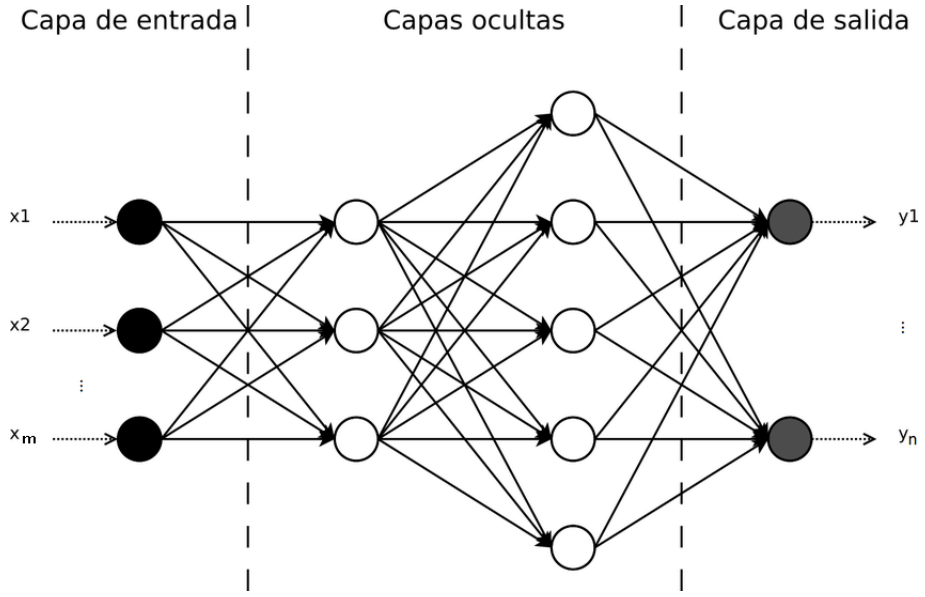

Figura 3.11: Tipos de capa, extraída de [38]

De esta forma, una red neuronal puede verse como una función $f(\mathbf{x}) = \mathbf{y}$ que transforma un conjunto de entradas $\mathbf{x} = (x_1, x_2, ..., x_m)$ en un conjunto de salidas $\mathbf{y} = (y_1, y_2, ..., y_n)$, que dependerán de los pesos asociados a las conexiones entre neuronas y del sesgo. El entrenamiento es, entonces, el proceso de ajuste de los pesos $w_1, w_2, ..., w_N$ y del sesgo b de cada una de las neuronas para que las salidas generadas se aproximen lo máximo posible a las etiquetas de los datos utilizados como entrada.

El entrenamiento comienza inicializando los pesos (de manera aleatoria o con unos valores preestablecidos) y procesando los datos de ejemplo. Las salidas calculadas por

la red se comparan con los valores esperados (etiquetas) y se calcula el error cometido con la función que se desee. Tras esto, entra en juego un algoritmo optimizador, como el de Descenso del gradiente [43], para actualizar los valores de los $b, w_1, ..., w_N$ y, con ello, intentar minimizar la función de error. Esto se repite iterativamente hasta alcanzar una condición de parada establecida de forma previa al inicio. Es habitual, por ejemplo, alcanzar un número de iteraciones máximo, un tiempo de ejecución determinado o incluso un valor umbral para el error. El proceso descrito se conoce como algoritmo de retropropagación y cada una de las iteraciones se denomina época.

Deep Learning

Dentro del *machine learning* existe un campo que se centra en el aprendizaje de patrones complejos sobre datos utilizando redes neuronales profundas, que no son más que redes neuronales con un elevado volumen de capas y neuronas. Es lo que se conoce como *deep learning*.

La idea básica de los algoritmos enmarcados en esta categoría es la existencia de una jerarquía de capas de forma que la información de los datos se extrae de forma progresiva (partiendo de información sencilla que se combina hasta obtener información más compleja).

El *deep learning* está teniendo en la actualidad una gran popularidad en la resolución de problemas de visión computacional, reconocimiento del habla, procesamiento del lenguaje natural, pero también en muchas más áreas. En concreto, las arquitecturas que están teniendo un mayor impacto son las redes convolucionales, las redes recurrentes y los *transformers*. En esta memoria no se describirán las redes convolucionales puesto que se utilizan principalmente para el procesamiento de imágenes y vídeos y, por tanto, excede el alcance del proyecto. No obstante, sí que se mencionarán los principios básicos de las redes recurrentes y de los *transformers*.

Redes recurrentes

Para la resolución de problemas basados en secuencias, como puede ser el caso de series temporales, surge la necesidad de emplear arquitecturas capaces de explotar esa componente de orden. Por ello surgen las redes recurrentes (RNN) [29], que presentan la capacidad de tener algo parecido al concepto usual de memoria y mantener así información sobre los elementos que ya han sido procesados por la red. Esto se consigue estableciendo conexiones entre neuronas de la misma capa o incluso dentro de la propia neurona (utilizando la salida de una neurona como entrada de esa misma neurona en la siguiente iteración) como se puede ver en la Figura 3.12.

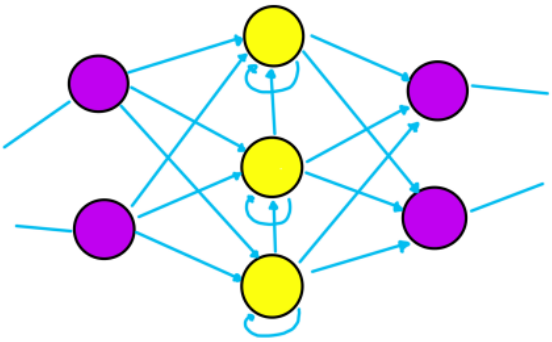

Figura 3.12: Ejemplo de red neuronal recurrente

Este tipo de redes pueden representarse de manera alternativa como un conjunto de redes neuronales no recurrentes conectadas entre sí que representan a la red original en diferentes instantes de tiempo (ver Figura 3.13).

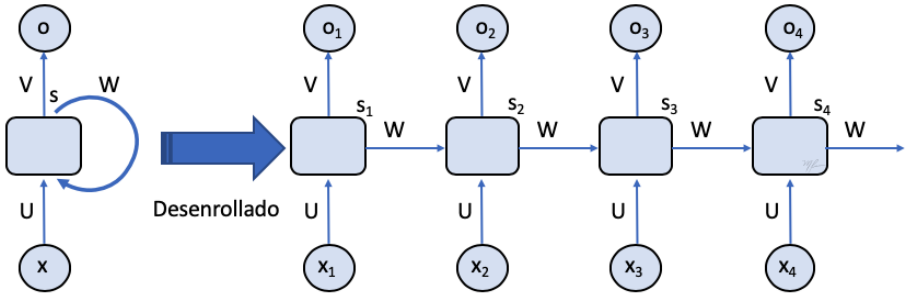

Figura 3.13: Forma desenrrollada de una red neuronal recurrente, extraída de [9]

Por otro lado, hay que tener en cuenta que el hecho de que las neuronas se realimenten entre sí dificulta el proceso de entrenamiento, puesto que el error cometido por una unidad de procesamiento ya no solo dependerá de los valores de sus pesos y sesgo, sino que habrá que tener también en cuenta el error acumulado por esa misma neurona en iteraciones pasadas. De esta forma, la manera en que deben actualizarse de los pesos para minimizar la función de error es algo más complicada. Es por ello que, para el entrenamiento de redes recurrentes, se utiliza una modificación del algoritmo de retropropagación visto anteriormente, conocido como algoritmo de retropropagación en el tiempo [49] y basado en la idea vista en la Figura 3.13: trabajar con la sucesión de redes no recurrentes en vez de con la red que realimenta sus neuronas.

No obstante, hay que tener en cuenta las limitaciones que presentan algunas de estas redes ya que, por su estructura, la influencia de las entradas se desvanece a lo largo del tiempo (problema del *vanishing gradient*). Esto deriva en que los modelos no son capaces de mantener una cierta memoria sobre datos a medio y largo plazo, y es el motivo principal por el que surgen las denominadas redes recurrentes Long Short-Term Memory (LSTM).

LSTM

Gracias a la capacidad de mantener dependencias a largo plazo, las redes LSTM [32] se han convertido en las redes neuronales recurrentes más utilizadas en la actualidad. Esto se consigue gracias a la estructura de sus neuronas que, como puede observarse en la Figura 3.14, es más compleja que en el caso tradicional.

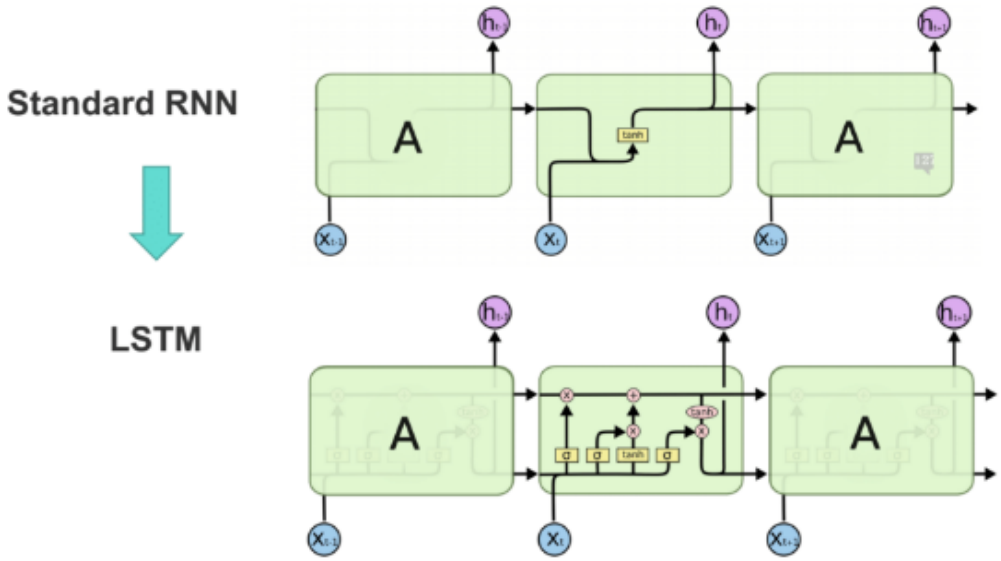

Figura 3.14: Neurona básica de una RNN frente a una neurona de LSTM, extraída de [10]

La innovación principal introducida en estas neuronas es el uso de una celda de estado encargada de almacenar la información. Por otro lado, cada neurona dispone también de otras estructuras denominadas puertas. En concreto poseen tres:

- Puerta de olvido. Encargada de decidir en qué medida el estado de la neurona anterior pasará a formar parte del nuevo estado. Para ello, considerando como entradas tanto la salida de la neurona anterior como la entrada de la neurona actual, utiliza la función de activación sigmoide para devolver un valor entre 0 y 1, donde 0 quiere decir que no se conserva nada de información previa y 1 que debe tenerse en cuenta todo el estado anterior.
- Puerta de entrada, cuya función es decidir qué valores de la entrada (la salida de la neurona anterior y la entrada de la neurona actual) deberían usarse para calcular el nuevo estado. En este caso se usa una función de activación sigmoide y, por otro lado, una función de activación tangente hiperbólica. El estado se actualiza en base a una combinación de los valores devueltos por ambas funciones.

- Puerta de salida. Se encarga de calcular la salida de la neurona en base a la salida de la neurona anterior, la entrada de la neurona actual y una transformación (mediante tangente hiperbólica) del estado calculado en la puerta anterior.

Las neuronas LSTM descritas pueden agruparse para conformar capas LSTM que, a su vez, conforman las llamadas redes LSTM.

Mecanismo de atención

El uso de los conocidos como mecanismos de atención comenzó hace algunos años con la finalidad de resolver problemas de traducción en combinación con redes neuronales recurrentes [24]. Sin embargo, este concepto alcanzó una mayor popularidad con la aparición de los *transformers*, que prescinden de la recurrencia y de las convoluciones, y se basan totalmente en las capacidades de esta tecnología. Antes de describir los *transformers*, se presentará primero la idea básica detrás de los mecanismos de atención. Para ello, nos basaremos en el ejemplo original aplicado a la traducción entre idiomas, aunque pueden extenderse a otro tipo de problemas.

La función principal de los mecanismos de atención es, a partir de un conjunto de palabras introducido como entrada, determinar y resaltar las posibles relaciones entre las mismas. Para ello, construye una matriz en la que relaciona todas las palabras y les asigna un valor entre -1 y 1 en función de la importancia de la relación semántica que posean. De esta forma, con esta información adicional el modelo será capaz de realizar la traducción de forma más sencilla que según la forma tradicional. Por ejemplo, si partimos de la oración "El sofá azul del salón es bonito pero está un poco roto", el mecanismo de atención debería identificar con un valor cercano a 1 la relación existente entra las palabras "sofá" y "roto", puesto que ambas se refieren al mismo objeto, pero generará valores más bajos para otras relaciones como "salón" y "un".

Este valor se calcula en base a una serie de operaciones matemáticas con matrices (que no se describirán en la memoria) para las cuales hay que tener en cuenta que las oraciones que se toman como entrada del modelo se deben representar de forma numérica como elementos de un espacio vectorial para que puedan ser procesadas. Esto se hace utilizando una técnica denominada *embedding* [41] que transforma las palabras en vectores denominados *tokens*.

Sin embargo, los *transformers* van un paso más allá y utilizan las denominadas *Multi-head Attention* que, aunque globalmente realizan la misma tarea descrita anteriormente, dividen la dimensión del espacio vectorial en que se encuentran los *tokens*, de forma que cada uno de los mecanismos de atención (denominados cabezas de atención) procesan uno de los subespacios. De esta manera, en lugar de calcular el valor final en una única operación de matrices, cada mecanismo calcula un valor "individual", que será promediado con el resto para obtener un valor final. Por ejemplo, si el espacio vectorial en que se codifican las palabras tiene dimensión 256 y contamos con 4 cabezas de atención, entonces cada una de ellas calculará el valor de la relación teniendo en cuenta solo 64 dimensiones. Así se consigue reducir el tiempo de cómputo y paralelizar algunas operaciones.

Transformers

Utilizando como base los mecanismos de atención descritos en el apartado anterior, en 2017 se presentan formalmente los *transformers* en un artículo científico titulado "Attention is all you need" [47]. En el artículo se menciona que esta novedosa arquitectura reporta mejores resultados que las redes LSTM de aquel entonces para la tarea de traducción. Además, también se comenta una importante ventaja de la arquitectura con respecto al proceso de entrenamiento, y es que, por cómo se definen las operaciones de los diferentes módulos, el procesamiento es fácilmente paralelizable, pudiéndose explotar así todo el potencial de las GPUs actuales y reduciendo drásticamente el tiempo de ejecución requerido.

La estructura de los *transformers* se basa en el uso de componentes que ya existían en ese momento, pero que nunca se habían combinado de esa manera. A grandes rasgos, se divide en dos bloques (ver Figura 3.15): un encoder (el bloque izquierdo) y un decoder (el bloque de la derecha).

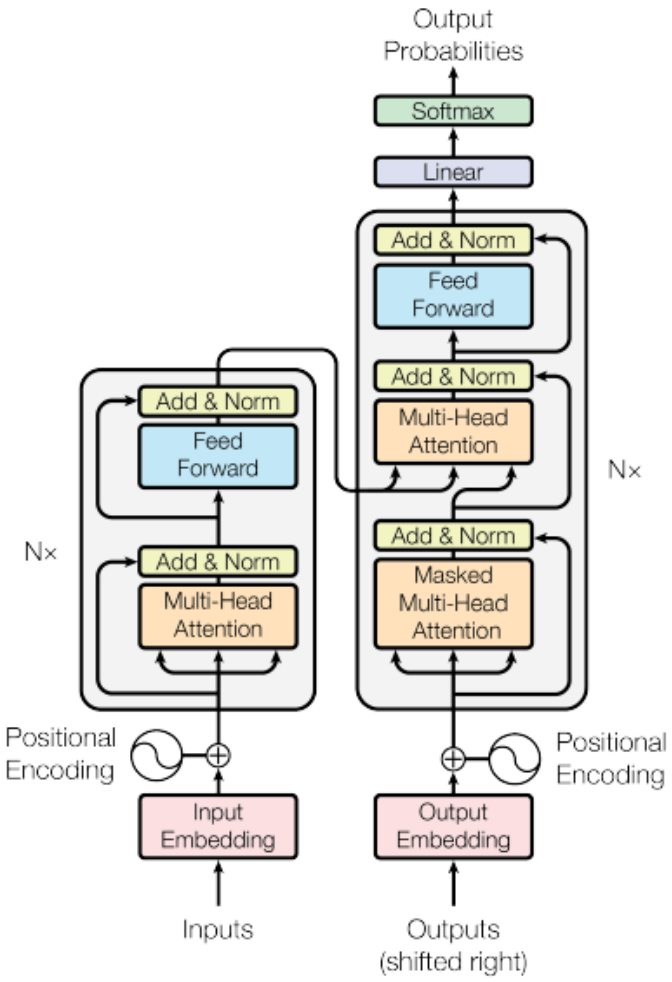

Figura 3.15: Estructura de la arquitectura Transformer, extraída de [47]

La misión principal del encoder es la extracción de las características de los datos de entrada, reduciendo la redundancia y generando una representación más sencilla. Por otro lado, el decoder actúa de forma inversa transformando la información extraída por el encoder para generar una salida.

Como se puede observar en la Figura 3.15, tanto el encoder como el decoder están formados por módulos que pueden apilarse varias veces (de ahí el Nx que aparece en el lateral de la figura). Estos módulos se componen, principalmente, de mecanismos de atención y de capas *feed forward* (que solo permiten el paso de información hacia delante). Por otro lado, se colocan módulos *embedding* para realizar la codificación al espacio vectorial de cada una de las entradas (tanto las entradas del encoder, como las etiquetas que se pasan al decoder).

Ya se mencionó anteriormente el hecho de que los *transformers* suprimen por completo el uso de la recurrencia, eliminando también por ello la capacidad de guardar una cierta memoria sobre el orden de entrada de los elementos, especialmente importante en el caso del Procesamiento del Lenguaje Natural o *Natural Language Processing* (NLP). Por ello, esta arquitectura añade el uso de un *Positional Encoding*, que permite identificar la posición de cada uno de los tokens y enviar de manera simultánea todos ellos al modelo. Para ello, añade un valor adicional al vector que representa cada una de las palabras, de forma que la red pueda interpretar su posición en la sentencia. De esta forma, en la frase "Ayer me puse una camiseta naranja después de desayunar una naranja" los tokens de cada una de las dos apariciones de la palabra "naranja" serán ligeramente distintos para que la red pueda inferir la posición de las mismas.

Por otro lado, el entrenamiento de este tipo de arquitecturas se lleva a cabo con un algoritmo de retropropagación similar a lo visto anteriormente, donde se actualizan los pesos de los diferentes módulos y capas, incluyendo los de los mecanismos de atención.

Finalmente mencionar que, aunque en este apartado solo se haya tratado el caso de *transformers* para el problema de traducción, que era el uso original de los creadores, con el tiempo este tipo de arquitecturas se han utilizado en otros contextos como en la generación de texto, la visión artificial o la predicción de series temporales. Dentro de esta última categoría, que es la de mayor interés para este Trabajo Fin de Máster, destacan los Temporal Fusion Transformers, que introduciremos a continuación.

Temporal Fusion Transformers (TFT)

En 2020 Google presenta Temporal Fusion Transformer [34], una arquitectura basada en *transformers* capaz de utilizar el mecanismo de atención para extraer de manera eficiente las relaciones existentes en series temporales. Es muy importante remarcar el término "basada" de la oración anterior, puesto que los TFTs incluyen capas LSTM y, como hemos visto en la sección anterior, los *transformers* prescinden de todo tipo de recurrencia. De esta forma, este tipo de modelos combina el uso de neuronas LSTM con una estructura de encoder-decoder con mecanismos de atención similares a los que vimos en la Figura 3.15.

Una de las características principales de los TFTs es la diferenciación en 3 tipos que define para las características de entrada:

- *Time-varying known*. Características que varían a lo largo del tiempo pero cuyo valor se conoce a futuro. Por ejemplo, días festivos a lo largo de un año o periodos de rebajas.
- *Time-varying unknown*. En esta categoría se enmarcan aquellas variables que varían a lo largo del tiempo pero cuyos valores futuros no son conocidos. Esto no quiere decir que sean variables que se deseen predecir, solamente se hace referencia a su naturaleza. Como ejemplo de este tipo de atributos se podría mencionar la temperatura de un local o el volumen de ventas de un determinado producto.
- *Static*. Características que no varían a lo largo del tiempo. Por ejemplo, el nombre de una empresa o el código identificador de un producto.

De esta forma, el *transformer* es capaz de realizar un tratamiento diferente y acorde a las necesidades de cada tipo de variable de forma que la complejidad de la red se pueda adaptar según la naturaleza de los datos. Además, la arquitectura posee una red de selección de variables para seleccionar las características más importantes y evitar que los modelos se ajusten a otras menos relevantes.

Por otro lado, otra de las fortalezas de esta arquitectura es la capacidad de realizar predicciones por cuantiles, de forma que se pueden representar rangos de predicciones (en lugar de un único valor) de mayor o menor amplitud según las necesidades del problema. De esta forma, se obtienen pronósticos más robustos y realistas que permiten realizar mejores evaluaciones de riesgos a la hora de tomar decisiones.

Aunque en esta memoria no se analizará en detalle la estructura de este tipo de redes, los diferentes módulos que la componen pueden observarse en la Figura 3.16.

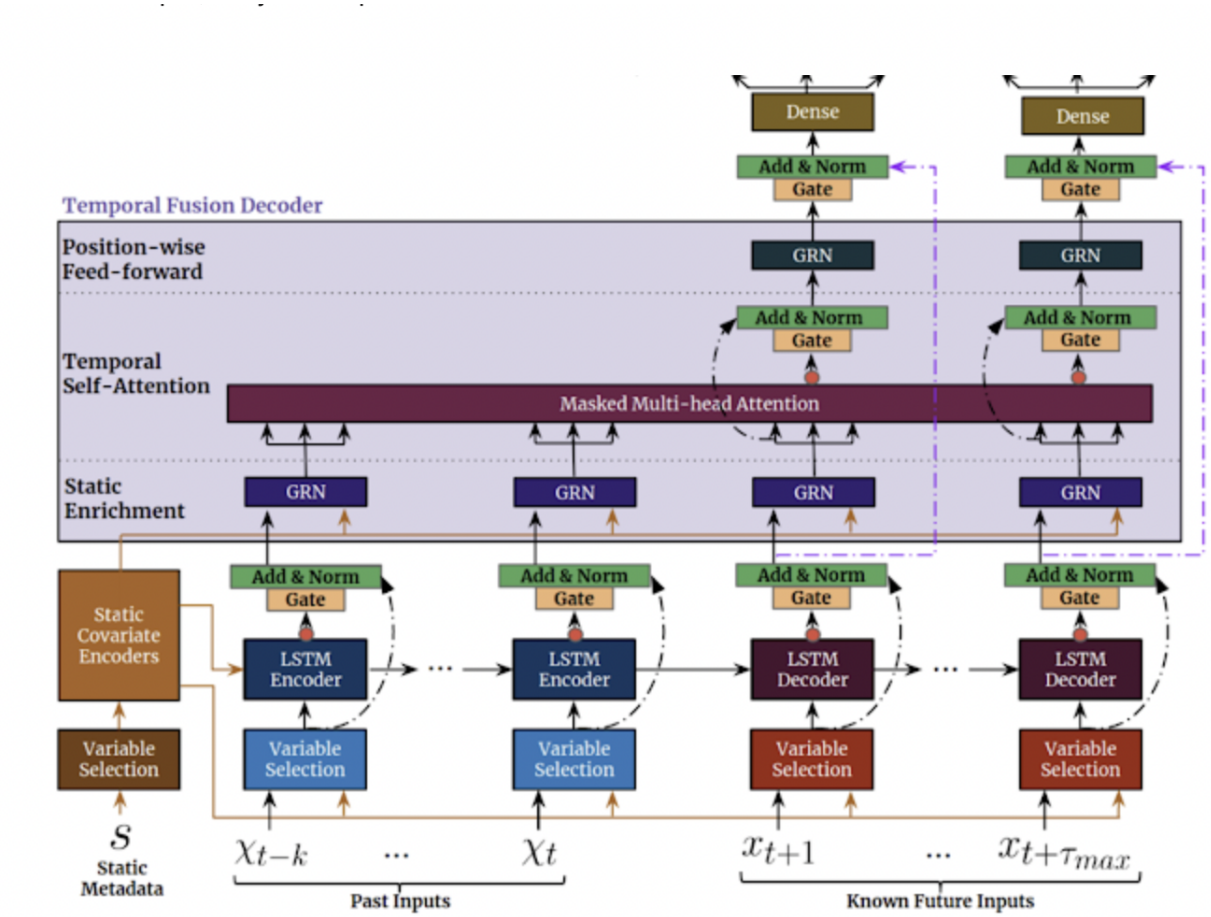

Figura 3.16: Estructura de la arquitectura TFT, extraída de [34]

Herramientas y librerías

Para concluir con el contexto científico-técnico se realizará una breve descripción de las diferentes tecnologías utilizadas.

En primer lugar, las ejecuciones realizadas con redes LSTM se han llevado a cabo sobre un ordenador portátil con Intel Core i7 y 16GB de RAM y, en paralelo, sobre una máquina virtual de características similares. Por otro lado, dada la alta complejidad de cómputo que presentaba la arquitectura TFT, fue necesario recurrir a una máquina virtual con GPU, en concreto una Nvidia RTX A40. Todo el código se desarrolló en cuadernos de Jupyter Notebook sobre una distribución de Anaconda (2.4.0).

Las redes LSTM se implementaron sobre la versión 2.9.1 de Tensorflow mientras que, para los TFTs, se utilizó Pytorch (versión 1.13.1 compatible con GPU). Además, en ambos casos se utilizaron otras librerías conocidas como scikit-learn, plotly, pandas o numpy. Por otro lado, el dashboard de visualización de resultados se desarrolló con el framework dash. Finalmente, la monitorización de los experimentos y la generación de curvas se llevó a cabo con la herramienta web Weights & Biases [18].

4: Estado del arte

En este capítulo se presenta una breve revisión de los avances más significativos en el contexto del problema planteado en esta memoria: la predicción de trayectorias de aeronaves.

4.1. Red SS-DLSTM

El primer proyecto analizado, titulado "A deep learning approach for aircraft trajectory prediction in terminal airspace" [50], se centra en el problema de predicción de trayectorias de aviones en los alrededores del aeropuerto, que es donde suelen producirse más variaciones con respecto al plan de vuelo establecido, debido a la densidad de aeronaves y a la complejidad estructural del mismo. El estudio se plantea como un problema de predicción de trayectorias 4D de tipo *sequence-to-sequence*, es decir, el modelo tomará como *input* una secuencia de n datos y generará como *output* una secuencia de m elementos. Para ello utiliza una red neuronal profunda de tipo LSTM (SS-DLSTM) formada por un *encoder*, capaz de realizar la extracción de características de los datos, y por un *decoder*, encargado de realizar las predicciones buscadas. El *dataset* utilizado se compone de datos extraídos de los sistemas ADS-B de aviones con origen o destino en *Baiyun International Airport (GBIA)* en China entre el 1 de agosto y el 30 de noviembre de 2018. Para entrenar el modelo, se utilizan los datos relativos a la posición del avión (longitud, latitud, altitud y *timestamp*), la dirección principal de control de la aeronave (el ángulo entre el eje vertical del avión y el plano horizontal), la velocidad direccional (*velocity*) y el tipo de aeronave. En algunos experimentos se utilizan también otros atributos derivados como la velocidad (*speed*) o la aceleración.

Por otro lado, esta propuesta aporta como una de sus contribuciones principales el uso de un modelo de regularización, consistente en minimizar una función objetivo basada en la función de Tikhonov para aproximar la trayectoria original, que presenta *outliers*, ausencia de algunos puntos y en la que los intervalos de tiempo son de diferente longitud, por una que no presenta ese tipo de irregularidades (ver proceso en Figura 4.17).

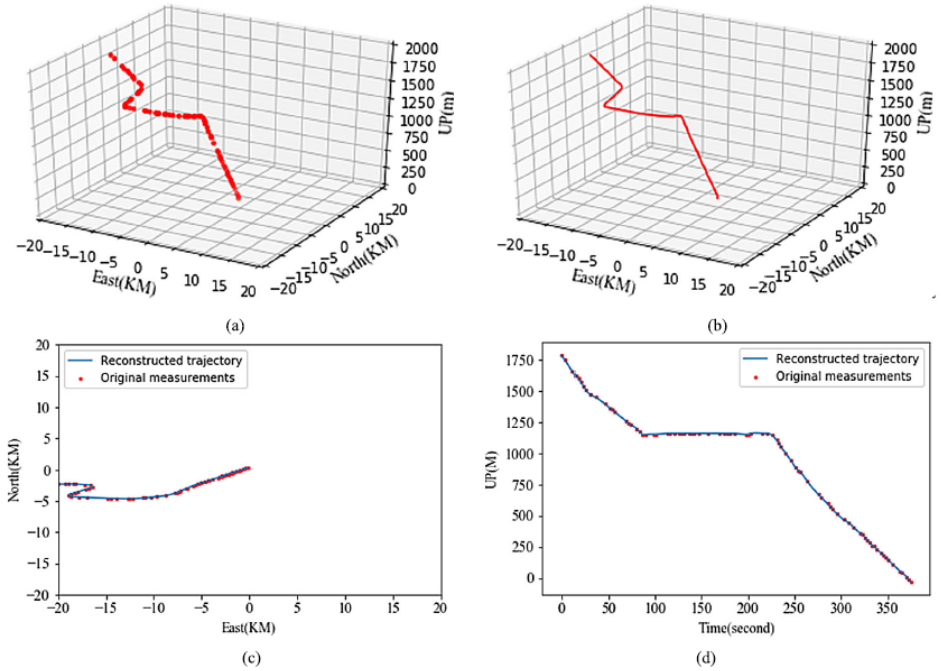

Figura 4.17: Ejemplo de aproximación de trayectoria usando el modelo de regularización

Las 98.620 trayectorias regularizadas son las que se emplean para entrenar el modelo SS-DLSTM mencionado al inicio. La longitud de la secuencia de predicción (el *output*) se fija en 90 segundos para las trayectorias correspondientes a una maniobra de despegue y en 150 segundos para una de aterrizaje. Por otro lado, la longitud de la secuencia de entrada, se optimiza como hiperparámetro, junto con el número de capas ocultas y el número de unidades de procesamiento por capa, en el proceso de entrenamiento.

El proceso de experimentación, para el que se utilizan diversas métricas, muestra que el modelo propuesto presenta mejores resultados que otros algoritmos básicos en todos los casos, tanto en la fase de despegue, como en la de aterrizaje.

4.2. Red híbrida: CNN-LSTM

La segunda de las propuestas analizadas, titulada "A hybrid CNN-LSTM model for aircraft 4D trajectory prediction" [35], utiliza una arquitectura híbrida compuesta por una red neuronal convolucional (CNN) y una red LSTM. La idea es tratar de optimizar la extracción de características: mientras que la convolución 1D empleada es capaz de obtener las características espaciales del conjunto de datos, las capas LSTM identificarán las temporales (ver estructura de la red en Figura 4.18).

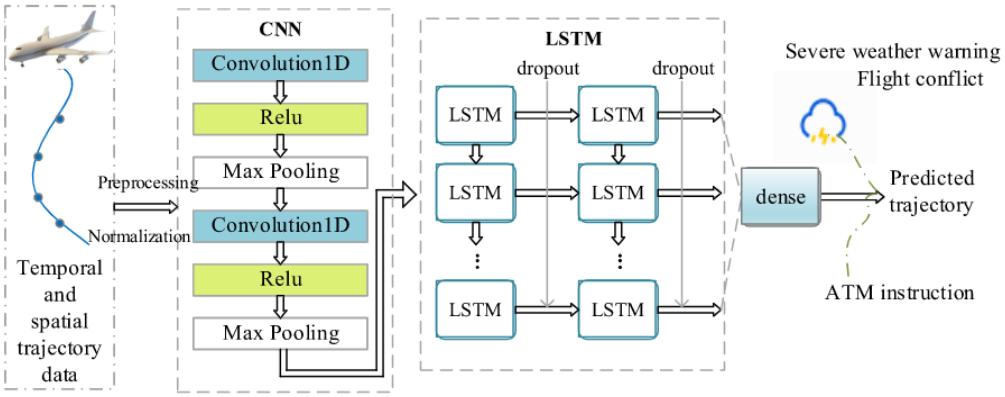

Figura 4.18: Estructura de la red híbrida

En este caso, la predicción de una trayectoria se define como el cálculo de un valor de tiempo, longitud, latitud y altitud en instantes futuros, donde se exploran dos variantes:

- *One step prediction*, es decir, realizando solo la predicción del siguiente valor en la trayectoria.
- *Sequence-to-sequence*, en el que se busca predecir la trayectoria para 3 y 5 instantes posteriores al actual.

En todos los casos, se utiliza un conjunto de datos ADS-B capturados en vuelos realizados desde Qingdao a Beijing entre febrero y mayo de 2017. Sobre el mismo se aplica una etapa de preprocesamiento y otra de normalización. En el preprocesamiento, entre otras cosas, se eliminan los puntos repetidos y se interpolan con un *spline* cúbico los valores ausentes (en el caso de que el porcentaje de los mismos no sea demasiado alto. Si hay demasiados valores ausentes, se eliminan esas trayectorias completas). Por otro lado, los atributos del conjunto de datos utilizados en los experimentos son la longitud, latitud, altitud, velocidad direccional, dirección del morro del avión y tiempo en que se capturan los datos ADS-B.

El estudio, realizado con Keras, muestra que el uso combinado de ambos modelos presenta mejores resultados que los obtenidos individualmente. Como principales desventajas de esta propuesta destaca el hecho de que las predicciones realizadas se han estudiado solo a corto plazo, y que el conjunto de datos no es lo suficientemente amplio, tanto porque solo explora una ruta concreta (Qingdao – Beijing), como porque no incluye otros factores que pueden influir en el rumbo de un vuelo como pueden ser datos temporales o datos relativos al plan de vuelo.

4.3. *Constrained* LSTM

La tercera propuesta analizada, titulada "4-D flight trajectory prediction with constrained LSTM network" [45], presenta una *constrained LSTM network* (cLSTM), es decir, una red LSTM sobre la que se consideran ciertas restricciones físicas. En concreto se consideran tres: *top* del ascenso o TOC (que influye principalmente en el ángulo de ascenso), puntos de la vía aérea o WPT (que aproxima los puntos de la trayectoria por una función lineal para reducir el coste computacional) y la dirección de la pista o RWD (que afecta al rumbo de los vuelos cuando se aproximan a una pista).

El *dataset* utilizado contiene nuevamente datos ADS-B, pero en este caso de junio a noviembre de 2017. Los atributos a utilizar son: *timestamp*, longitud, latitud, altitud y ángulo del morro del avión. En la fase de preprocesamiento se segmentan las trayectorias aplicando el algoritmo DBSCAN. De esta forma, se obtienen dos *clusters* de trayectorias: "en ascenso" y "no en ascenso". Este último grupo se divide posteriormente en otros dos: "*cruising*" (fase entre ascenso y descenso) y "en descenso" a partir de la distancia restante hasta el aeropuerto. El objetivo de este proceso de *clustering* es poder aplicar, de manera personalizada, unas u otras restricciones según el caso, lo que se traduce en utilizar una combinación lineal de estas restricciones en la definición de la función de pérdida de la red LSTM. En concreto:

- En la fase de ascenso, la función de pérdida para la red LSTM tendrá solo en cuenta la restricción de *top* del ascenso.
- Para los datos clasificados como *cruising* únicamente se considera la restricción WPT.
- Por último, para la fase de descenso se consideran las tres restricciones, que se ponderan con la media aritmética.

La idea es entonces concatenar las tres redes neuronales junto con sus correspondientes funciones de pérdida para construir la red cLSTM. Los datos ya clasificados en clusters se normalizan y agrupan en subconjuntos con una ventana deslizante, considerándose 10 elementos como entrada a la red y 1 como salida. La predicción será, por tanto, a un solo paso. Los resultados obtenidos muestran mejores valores frente a otros modelos como LSTM simple, *Support Vector Machine* (SVM) o *Majorization-Minimization Model* (MM).

4.4. Red generativa adversaria condicionada

El siguiente proyecto seleccionado, titulado "Conditional generative adversarial networks (CGAN) for aircraft trajectory prediction considering weather effects" [40], sugiere el uso de redes generativas adversarias condicionadas (CGAN). Esto quiere decir que se utilizan dos redes independientes, un generador y un discriminador, a los que se añaden más condiciones en las entradas. En concreto, en este modelo se utilizan como condiciones el último plan de vuelo registrado y los datos meteorológicos asociados al plan de vuelo en que se enmarca. El objetivo del estudio es la predicción de la trayectoria de la aeronave antes de partir utilizando las condiciones mencionadas.

La red generadora incluye una red neuronal convolucional con dos capas convolucionales y dos capas densas que se centran en extraer las características de los datos meteorológicos. Posteriormente, esta información se concatena con la información relativa a planes de vuelo y con un tensor formado por ruido aleatorio (distribución normal de media 0,5 y varianza 1) en una única capa LSTM para generar una predicción en la trayectoria como salida. Por otro lado, el discriminador presenta una estructura similar, aunque con una única capa convolucional, y se centra en diferenciar las trayectorias reales de las creadas por el generador. Todo ello puede observarse en la Figura 4.19.

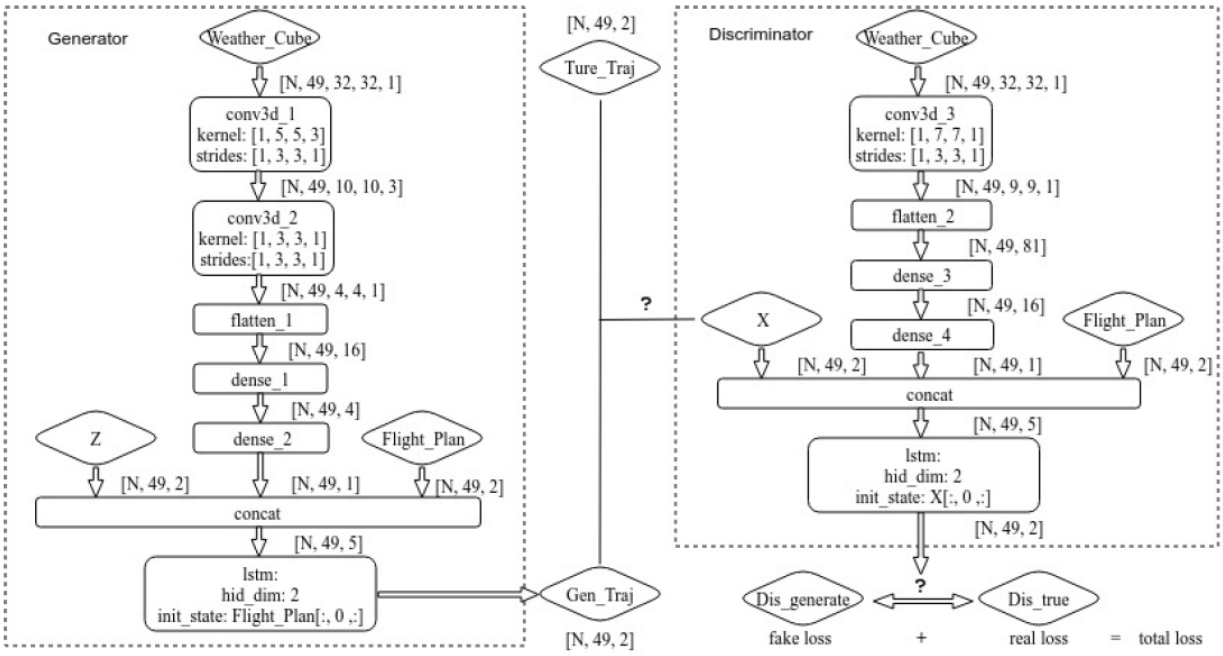

Figura 4.19: Estructura del generador y discriminador

La idea es entonces maximizar la posibilidad de reconocer los datos artificiales (los creados por el generador) como reales, lo que se traduce en minimizar la función de pérdida del discriminador.

En cuanto a los datos utilizados en este proyecto, se ha optado por utilizar datos de sectores de vuelo en lugar de trayectorias completas entre aeropuertos. Los sectores se definen como áreas bien definidas sobre el mapa sobre las que trabaja un determinado Centro de Control de Tráfico Aéreo (ARTCC). De esta forma, se utilizarán los datos de aquellos vuelos que han atravesado un sector aéreo fijado. Por supuesto, esto supondrá limitaciones en los resultados, ya que el modelo no estará entrenado para hacer predicciones fuera de ese sector. La fuente de datos elegida es Sherlock Data Warehouse y el sector seleccionado Indianapolis Air Route Traffic Center (ZID), de donde se extraen datos del día 24 junio 2019. Esta información se completa con datos relativos a planes de vuelo y a datos meteorológicos. Los datos pasan por una etapa de preprocesamiento en la que se realizan una serie de interpolaciones (para obtener intervalos de muestreo uniformes) y posteriormente se utilizan como entrada en las redes mencionadas.

Los resultados obtenidos en el estudio muestran una mayor precisión en la predicción con respecto a un modelo Conv-LSTM pero, según sus conclusiones, los resultados no satisfacen sus expectativas iniciales debido, posiblemente, al alto valor de desviación presente en los datos usados.

4.5. Red híbrida: *Machine Learning* con modelos físicos

El quinto proyecto analizado, titulado "Hybrid machine learning and estimation-based flight trajectory prediction in terminal airspace" [27], opta por una propuesta híbrida pero, en este caso, combinando modelos de *machine learning* con métodos de estimación físicos. Esta elección busca que, en una primera etapa, el modelo de aprendizaje automático entrenado con datos históricos realice una predicción del comportamiento futuro para un vuelo, mientras que, posteriormente, el modelo físico utilice ese valor predicho como variable a tener en cuenta junto con otras mediciones reales tomadas sobre la aeronave. El objetivo del estudio es la predicción en tiempo real de la trayectoria que seguirá el vuelo en los dos minutos siguientes al actual.

Nuevamente se utilizan datos ADS-B relativos a la longitud, latitud, altitud, velocidad horizontal, velocidad vertical y *timestamp*. En este caso, se consideran vuelos con origen o destino en el Incheon International Airport de Corea del Sur entre junio y agosto de 2019. Sobre este conjunto se aplica una fase de limpieza en la que se eliminan los puntos repetidos. En las trayectorias que presentan algunos puntos ausentes se interpolan esos valores, pero si son demasiado abundantes se eliminan las trayectorias completas. A esto le sigue una etapa de preprocesamiento en la que las trayectorias se agrupan usando un algoritmo de identificación de patrones desarrollado por los mismos investigadores (Deng, K. Kim, H. Choi [31]) con el objetivo de que el proceso de manejo de trayectorias sea más efectivo y eficiente (ver Figura 4.20).

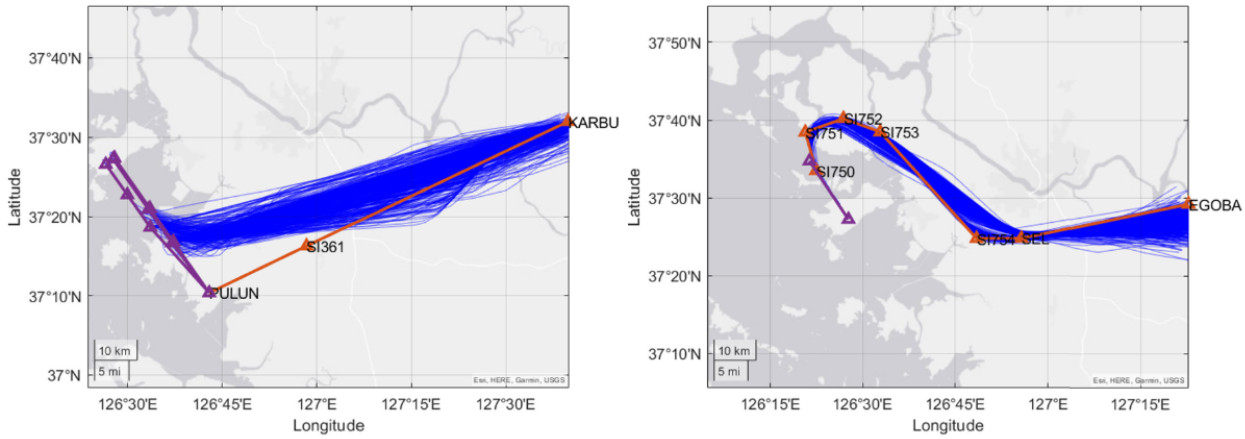

Figura 4.20: Agrupación de trayectorias utilizando identificación de patrones

Como he mencionado al inicio, el conjunto de datos preprocesado se usará como *input* de un modelo híbrido que presenta dos submodelos diferentes:

- Modelo basado en *machine learning*. En este caso se plantean inicialmente dos arquitecturas diferentes: Gaussian Mixture Model (GMM) y LSTM. Sin embargo, esta última se descarta debido al elevado tiempo de computación que imposibilita el procesamiento en tiempo real.
- Modelo de estimación físico. En concreto, el movimiento del avión se modela como un Stochastic Linear Hybrid System (SLHS). Esto se complementa con el algoritmo de estimación Residual-Mean Interacting Multiple Models (RM-IMM).

La intención de este modelo híbrido es aunar los beneficios de ambos métodos, lo cual se cumple según los resultados obtenidos, que muestran mejores resultados con respecto a modelos simples de aprendizaje automático como LSTM o GMM.

4.6. *Machine Learning* con uso previo de *clustering*

La última de las propuestas analizadas, titulada "Short-Term trajectory prediction using generative machine learning methods" [33], plantea el uso de un mecanismo de *clustering* junto con un algoritmo de *machine learning* para la predicción de trayectorias a corto plazo (una predicción) en un sector aéreo determinado. La idea es clasificar el histórico de trayectorias en K grupos (*clusters*) en base a sus características espaciales para, a continuación, entrenar K modelos predictivos, uno para cada una de las agrupaciones (ver Figura 4.21). Las arquitecturas utilizadas son Random Forest, one versus-all Support Vector Machine y Multi-layer Perceptron.

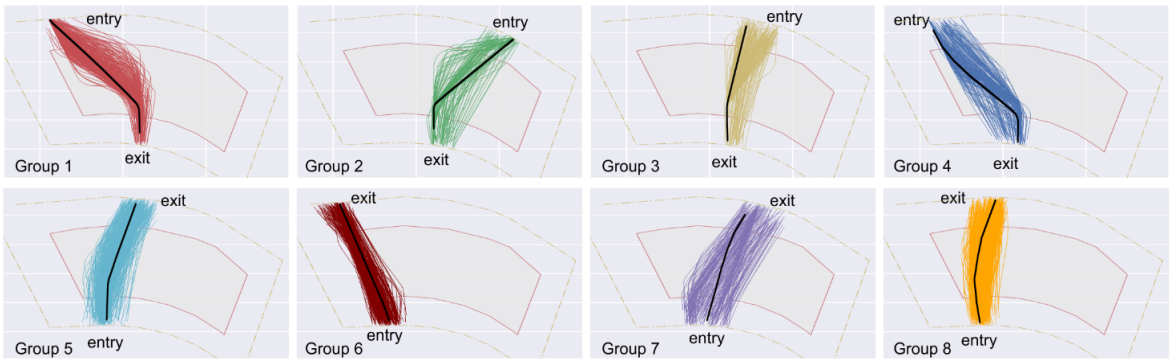

Figura 4.21: Valor óptimo de *clusters* identificados

Por otro lado, para poder llevar a cabo el proceso de predicción con los modelos entrenados, es necesario implementar otro mecanismo que sea capaz de clasificar la trayectoria en uno de los K *clusters*. Para ello se utiliza el algoritmo Random Forest y se construyen dos modelos diferentes: uno para clasificar las trayectorias que entran por el norte y otro para las del sur. Una vez clasificada la trayectoria, se utilizará el mejor de los modelos de predicción generados en la primera etapa.

Los modelos se entrenan utilizando datos ADS-B capturados entre el 1 y 15 de marzo de 2019 en el sector 2E de Singapur. Los atributos considerados son: longitud, latitud, altitud, velocidad, hora del día, callsign, aeropuerto de origen y aeropuerto de destino. Únicamente se eliminan aquellas trayectorias con menos de 6 puntos. El resto elementos de las trayectorias se aproximan para que su *timestamp* difiera en 30 segundos.

Los resultados muestran que el modelo que mejor funciona es Random Forest, que es capaz de predecir la trayectoria de los 5 minutos posteriores con un error de 1,06 millas náuticas y los 10 minutos siguientes con un error de 1,69 millas náuticas.

4.7. Discusión

Se presenta en esta sección una breve discusión sobre los trabajos analizados. Las dimensiones comparativas que se han escogido son: arquitectura utilizada, tipo de predicción a realizar, *datasets* seleccionados, tipo de preprocesamiento usado y métricas con que se evalúan los resultados.

Propuesta	Arquitectura	Tipo de predicción	Datasets	Preprocesamiento	Métricas
4.1	SS-DLSTM	Seq2seq	ADS-B	Regularización con función Tikhonov	EE, ATE, CTE, AE
4.2	CNN-LSTM	One step	ADS-B	Interpolación con spline cúbico	RMSE, MAE, MAPE
4.3	Constrained LSTM	Seq2seq y One step	ADS-B	Segmentación con DBSCAN	RMSE, MAE, MRE, DTW
4.4	CGAN	Seq2seq	Datos de vuelo por sector, planes de vuelo y climatológicos	Interpolación	RMSE
4.5	LSTM + RM-IMM	Seq2seq	ADS-B	Identificación de patrones	HE, ATE, CTE, VE
4.6	RF, one versus-all SVM y MLP	Seq2seq	ADS-B	Clustering	TWAE, MAE, Std. MAE

Tabla 4.1: Comparativa de proyectos

Como se puede observar, las propuestas analizadas utilizan mayoritariamente redes LSTM para la predicción de trayectorias, que suelen ser además de tipo *sequence-to-sequence*. Además, en todos los casos, a excepción del 4.4, se seleccionan datos ADS-B para entrenar y validar los modelos.

5: Diseño experimental

5.1. Descripción de los datos

El conjunto de datos del que se dispone para la realización de los experimentos se compone de mensajes ADS-B procedentes de *OpenSky*, junto con otros ficheros complementarios con información relativa a planes de vuelo o datos básicos de los aeropuertos. Sobre estos conjuntos se realizan algunas operaciones adicionales destinadas a la limpieza y reconstrucción de las trayectorias, que no se describirán en esta memoria puesto que fueron realizadas por los tutores del proyecto de forma previa a su inicio. Tampoco se presentará un análisis estadístico de los datos por el mismo motivo y todas las decisiones sobre la inclusión de atributos serán tomadas en base a la experiencia y recomendación de los profesores. Lo que sí se describirá es la estructura y características del *smart data* proporcionado, correspondiente a un conjunto de trayectorias con destino el aeropuerto Adolfo Suárez Madrid-Barajas desde una selección de 40 de los aeropuertos internacionales europeos con mayor volumen de tráfico anual, realizados entre el 1 de enero y el 30 de septiembre de 2022. El conjunto, almacenado en varios ficheros parquet [2], se compone de los vectores que conforman las citadas trayectorias y que se encuentran espaciados cada 15 segundos. Cada vector posee 46 atributos, por lo que solo serán descritos (en la Tabla 5.2) aquellos con mayor relevancia para el proyecto.

Por otro lado, las trayectorias del *dataset* aparecen divididas en tres subconjuntos:

- Entrenamiento o *train*, con 5.153 trayectorias, formadas por un total de 2.141.283 vectores.
- Validación o *val*, con 915 trayectorias, formadas por un total de 383.183 vectores.
- Evaluación o *test*, con 1.078 trayectorias, formadas por un total de 446.642 vectores.

De esta forma, el conjunto de entrenamiento posee aproximadamente el 72 % de las trayectorias, el de validación el 13 % y el de *test* el 15 %.

[2]Parquet: formato de almacenamiento con compresión orientado a columnas

Atributo	**Tipo**	**Descripción**
fpId	String	Identificador del vuelo
icao24	String	Identificador del transpondedor de la aeronave
callsign	String	Identificador (no único) del vuelo, asignado por la aerolínea
latitude	Float	Distancia entre el ecuador y el punto actual de la aeronave, medida en grados a lo largo del meridiano en el que se encuentra.
longitude	Float	Distancia entre el meridiano de Greenwich y el punto actual de la aeronave, medida en grados a lo largo del paralelo en el que se encuentra.
speed	Float	Velocidad horizontal de la aeronave, medida en nudos
vspeed	Float	Velocidad vertical de la aeronave, medida en pies por segundo
ground	Boolean	*Flag* que indica si la aeronave está o no en tierra
track	Float	Ángulo que forma el morro del avión con respecto al eje horizontal
aerodromeOfDeparture	String	Código del aeropuerto de origen
aerodromeOfDestination	String	Código del aeropuerto de destino
flightDate	String	Fecha del vuelo en formato YYY-MM-DD
vectorId	String	Identificador del vector
timestamp	Int	Instante de tiempo que representa el vector
altitude	Float	Altitud geométrica determinada por el sistema GPS, medida en pies
hav_distance	Float	Distancia a destino, medida en millas

Tabla 5.2: Atributos del *dataset* a utilizar

5.2. Arquitecturas

Como se ha mencionado en los objetivos del proyecto, se pretende utilizar dos tipos diferentes de arquitecturas: LSTM y TFT. Se describirán a continuación, de forma breve, las configuraciones que se pretenden probar.

Long short-term memory (LSTM)

En primer lugar, se presenta una red recurrente LSTM sencilla formada por capas distribuidas de forma secuencial. En concreto, se incluye una capa LSTM, junto con una capa densa y una capa para modificar la dimensionalidad del *array* de salida (ver Figura 5.22).

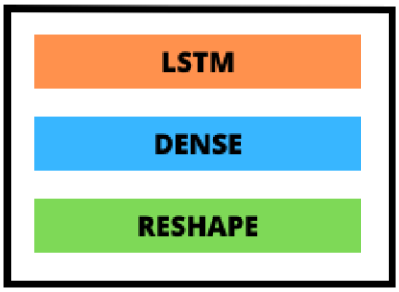

Figura 5.22: Estructura de la red LSTM sencilla

Por otro lado, en base a los resultados obtenidos con la red LSTM sencilla (ver Sección 7.3) y con la intención de mejorar las predicciones, se utilizará también otra versión de red LSTM que añada dos capas densas a la versión anterior (ver Figura 5.23).

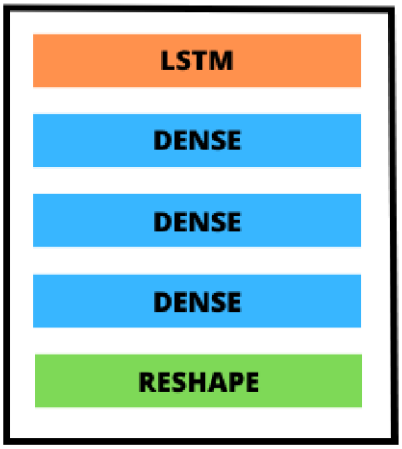

Figura 5.23: Estructura de la red LSTM que añade dos capas densas

Temporal Fusion Transformer (TFT)

Finalmente, para completar la experimentación y con el fin de incluir una arquitectura novedosa y que, a priori, debería presentar buenos resultados, se utilizará una implementación del TFT de Google cuya arquitectura puede consultarse en la Figura 3.16.

5.3. Métrica

En esta sección se describe cuál es la métrica elegida para evaluar y comparar los modelos entrenados, y cómo se calcula. Con el fin de obtener predicciones adecuadas al problema que se pretende abordar, se ha decidido escoger como métrica de evaluación el error cuadrático (MSE) [11], que se calcula de la siguiente manera:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

siendo n el número de elementos que se pretenden evaluar, y_i el valor real del elemento i ($i \in 1, 2, ..., n$) e $\hat{y}_i$ la predicción del modelo para ese mismo elemento. De esta forma, los valores obtenidos con esta métrica son siempre positivos y mejores cuanto más se aproximen a 0.

Así se consigue que los valores positivos y los negativos no se anulen, ya que cada uno de los errores individuales (los de cada uno de los n elementos) se eleva al cuadrado. Además, esta métrica penalizará los errores grandes de forma más agresiva que otras como el RMSE [3] , favoreciendo la obtención de curvas más regulares y suavizadas. Finalmente, la métrica es diferenciable a diferencia de otras como el MAE [4], por lo que se simplifican algunas operaciones matemáticas a la hora de ajustar los pesos de la red.

No obstante, la definición introducida para el error cuadrático medio se adecúa únicamente a problemas unidimensionales: predicción de un único valor para una sola característica. El problema que se pretende abordar, como hemos definido anteriormente, es la predicción a 10 puntos futuros de las tres características básicas: altitud, latitud y longitud. Sin embargo, la extensión del MSE a este caso es muy sencilla, se realiza la media aritmética del MSE obtenido para cada uno de los 10 puntos a predecir y para cada una de las 3 características. De esta forma, el MSE que se utilizará en la fase de experimentación se define de la siguiente manera:

$$MSE = \frac{1}{3}\sum_{i=1}^{3}\left(\frac{1}{10}\sum_{j=1}^{10} MSE_{ij}\right)$$

siendo MSE_{ij} el MSE para el punto de predicción j ($j \in 1, 2, ..., 10$) y para la característica i ($i \in 1, 2, 3$).

Una vez fijada la métrica es importante destacar algunas observaciones:

- Para realizar la evaluación de los modelos sobre los datos de *test* se mostrarán 4 resultados para una mejor comprensión de los mismos:
  - MSE para la altitud (calculado con los vectores desescalados)

[3] El RMSE o raiz del error cuadrático medio se define como $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} = \sqrt{MSE}$

[4] El MAE o error absoluto medio se define como $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

    - MSE para la latitud (calculado con los vectores desescalados)
    - MSE para la longitud (calculado con los vectores desescalados)
    - MSE global (calculado con los vectores escalados para que sea directamente comparable con los resultados de validación)

- Es importante definir las unidades en que se calcula cada característica para hacernos una idea de la magnitud del error cometido (cuando aparece desescalado). Por un lado, la latitud y la longitud se miden en grados, y por otro lado, la altitud se mide en pies (1 metro equivale a 3,29 pies).
- El modelo de TFT elegido calcula el MSE para cada una de las características por separado. Con el objetivo de que las curvas sean directamente comparables con las obtenidas en las redes LSTM se realizará un postprocesamiento de las mismas. Esto consiste en calcular la media aritmética de los MSE obtenidos de forma individual para cada una de las épocas.

6: Experimentación

En este capítulo se presenta cómo se ha llevado a cabo el proceso de experimentación y se analizan los resultados obtenidos. El problema que se pretende resolver es la predicción a futuro de 10 puntos (dos minutos y medio) pertenecientes a la trayectoria que recorre una aeronave. Para ello, se introducen previamente los hiperparámetros que se pretenden optimizar en el proceso. Además, se presenta también la métrica elegida para la posterior evaluación de modelos.

6.1. Hiperparámetros

En este apartado se definen los hiperparámetros que se optimizarán durante la experimentación.

Arquitecturas LSTM

Comenzamos definiendo aquellos hiperparámetros que se pretenden ajustar para las redes basadas en capas LSTM.

- **Número de épocas**. Las épocas de entrenamiento se definen como el número total de iteraciones realizadas sobre el *dataset* completo, que se encuentra agrupado en lotes o *batches*. De esta forma, puede describirse análogamente como el número de veces que se procesa cada dato. Se contempla un rango máximo de 80 épocas.
- **Función de activación**. Consiste en una función matemática que transforma la salida de una neurona artificial para limitar su rango de valores posibles y también para incentivar o disuadir ciertos comportamientos. Se considera la tangente hiperbólica, la sigmoide y la lineal.
- **Número de unidades de procesamiento de la capa LSTM**. Es decir, el número de neuronas que incluirá la capa recurrente. Se contempla un rango que va de 10 a 40 unidades.

- **Tamaño de la ventana deslizante o *lookback***. Número de registros pasados que se aportarán como entrada a la red neuronal para predecir los siguientes puntos de la trayectoria. Se establecen valores de entre 24 y 56 registros.

Temporal Fusion Transformer

Por otro lado, los parámetros a optimizar en la red TFT son:

- **Número de épocas.** Se establece un valor máximo de 120 épocas.
- ***Head attention size* o número de cabezas de atención**. Número de mecanismos de atención utilizados para calcular los pesos entre los *tokens* de entrada. Se consideran valores de 1, 4 y 8 cabezas.
- **Número de elementos de procesamiento de la capa oculta.** Se contempla un rango que va de 140 a 380 unidades.
- **Tamaño de la ventana deslizante.** Se establecen valores de entre 50 y 65 elementos.

6.2. Herramienta de visualización de predicciones

Para una mejor comprensión de la tendencia de las trayectorias predichas por los modelos, se toma una herramienta de visualización desarrollada de forma previa al inicio de este proyecto por Jorge Silvestre Vilches. Esta pieza software se implementa con el *framework* Dash [6], construido sobre Plotly. La principal ventaja que ofrece Dash, es la capacidad de actualizar sus diferentes componentes en tiempo real, como respuesta a la interacción del usuario con la interfaz del sistema. Para ello, se utilizan los denominados *callbacks*, que son funciones que se ejecutan como respuesta a la interacción con los elementos de la interfaz, y realizan cálculos o generan salidas en otros elementos.

Un ejemplo de representación de predicción de trayectoria con la herramienta descrita puede observarse en la Figura 6.24.

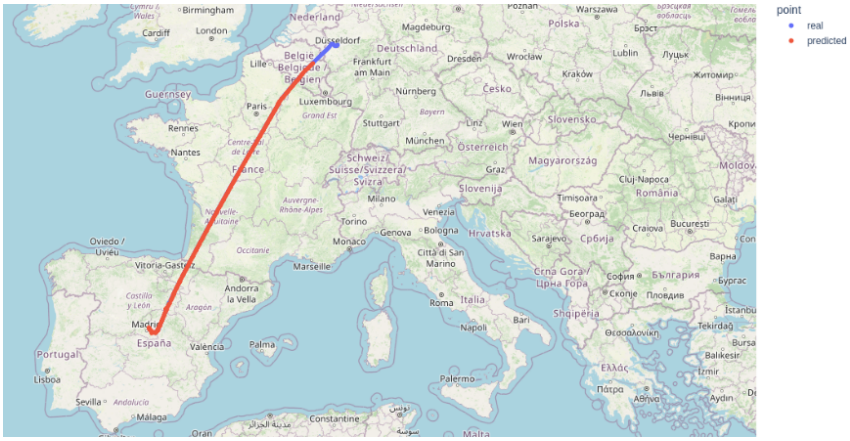

Figura 6.24: Ejemplo de predicción con la herramienta de visualización

Es importante mencionar que, aunque la herramienta ha sido proporcionada por el cotutor del proyecto, se ha modificado según las necesidades que han ido surgiendo a lo largo de la experimentación. Además, los errores que presentaba han sido solventados. La estructura final de *callbacks* que representa el funcionamiento del sistema de visualización puede consultarse en la Figura 6.25.

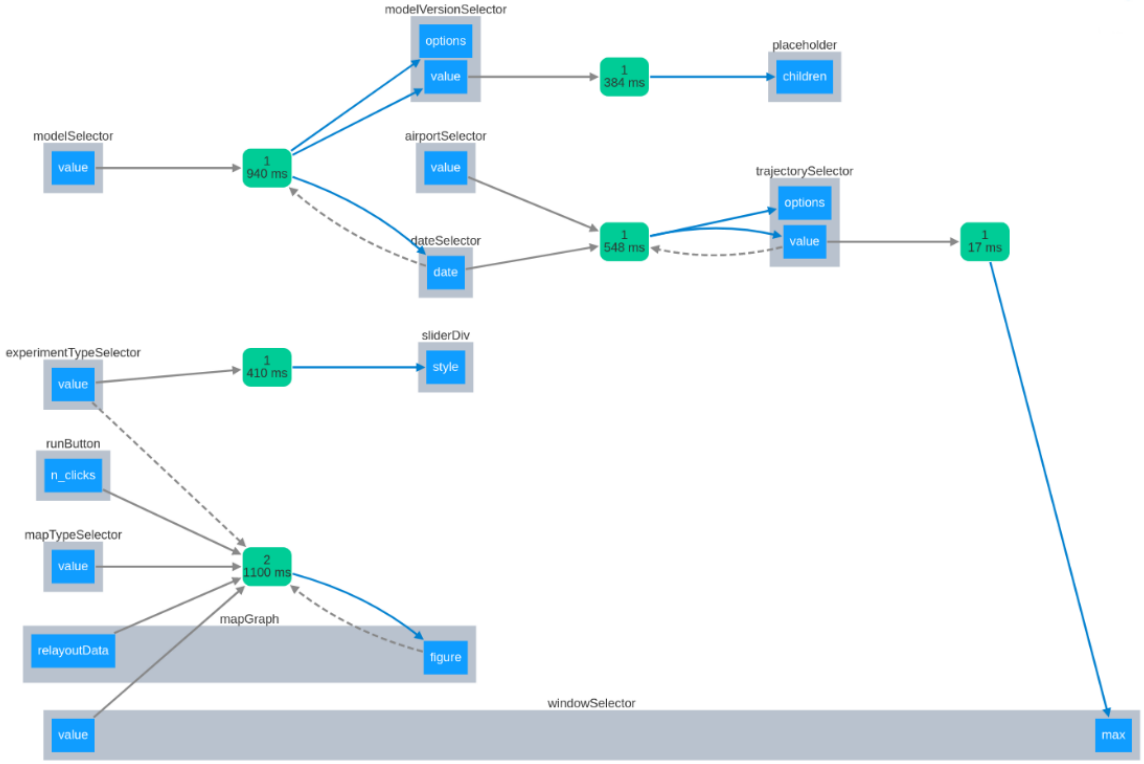

Figura 6.25: Estructura de *callbacks* de la herramienta de visualización

6.3. Proceso de experimentación

En este apartado se describe el proceso de experimentación llevado a cabo junto con los resultados obtenidos. Se ha decidido dividir el proceso en 3 apartados, uno por cada tipo de arquitectura: LSTM, LSTM añadiendo dos capas densas y TFT. Durante todas ellas, los datos se normalizan entre 0 y 1, y las variables categóricas se transforman a numéricas mediante un Label Encoder. Por otro lado, como se ha comentado anteriormente, se realizarán predicciones a 10 pasos.

Además, es importante mencionar que todas las figuras presentes en esta sección (generadas automáticamente por Weights & Biases) mostrarán los resultados de la métrica seleccionada sobre los datos de validación, pero no siempre desde la primera época. Esta decisión se ha tomado debido a que, durante las primeras iteraciones, el modelo presenta una disminución muy rápida del MSE en comparación con el resto de ciclos. De esta forma, se puede visualizar mejor el progreso obtenido durante el entrenamiento.

LSTM

En primer lugar, se realizan los experimentos con las redes LSTM. Para ello, se parte de la clase *Experimento*, desarrollada por Jorge Silvestre Vilches (cotutor del proyecto), y se generan los cuadernos necesarios para completar los diferentes entrenamientos. Además, según avance el proceso de experimentación y surjan necesidades no contempladas inicialmente en el código proporcionado (como en el Experimento 2), se añadirá la funcionalidad correspondiente para su posible ejecución.

Se ha decidido nombrar cada uno de los modelos entrenados según el valor de los hiperparámetros que se pretenden optimizar. De esta forma, se presentarán con el siguiente formato:

modelName_additionalInformation_sampling_lookback_units

donde *modelName* se corresponderá con el tipo de modelo utilizado, *additionalInformation* puede aportar o no algún tipo de peculiaridad sobre el modelo (como la función de activación utilizada), *sampling* indica el valor de equiespaciado de los datos (que como vimos antes siempre será 15 en el *smart data* proporcionado), *lookback* hace referencia al tamaño de la ventana utilizada y *units* especifica el número de unidades de la capa de procesamiento. Como ejemplo, se analiza la siguiente cadena:

LSTMTray_s15_lb16_u18

Se trata de un modelo LSTM, que no presenta información adicional, cuyos vectores están espaciados cada 15 segundos, el tamaño de la ventana es de 16 y su capa LSTM posee 18 unidades. Por otro lado, los modelos que obtengan mejores resultados se numerarán para una mejor identificación en la fase de evaluación.

Experimento 0: análisis exploratorio inicial

Para esta primera etapa, que se realiza con un subconjunto de los datos (los correspondientes al mes de septiembre), se pretende observar con qué rangos de parámetros se comportan mejor los modelos para poder extrapolar estas conclusiones al dataset completo. De esta forma, se hace una aproximación inicial de la tendencia que siguen las redes al modificar los hiperparámetros realizando pruebas más rápidas y ligeras.

Todas las pruebas se realizan con las características básicas del conjunto de datos: altitud, latitud y longitud. En primer lugar, se completa un entrenamiento con 80 épocas (manteniendo con los valores por defecto tanto el lookback como el número de unidades) para observar la tendencia de las curvas de entrenamiento y fijar un número de épocas a partir de las cuales se entiende que el modelo no debería mejorar. Como puede consultarse en la Figura 6.26, la red LSTM parece no mejorar los resultados sobre el conjunto de validación alrededor de la época 30. Por ello, se toma la decisión de no superar las 40 iteraciones en las siguientes pruebas.

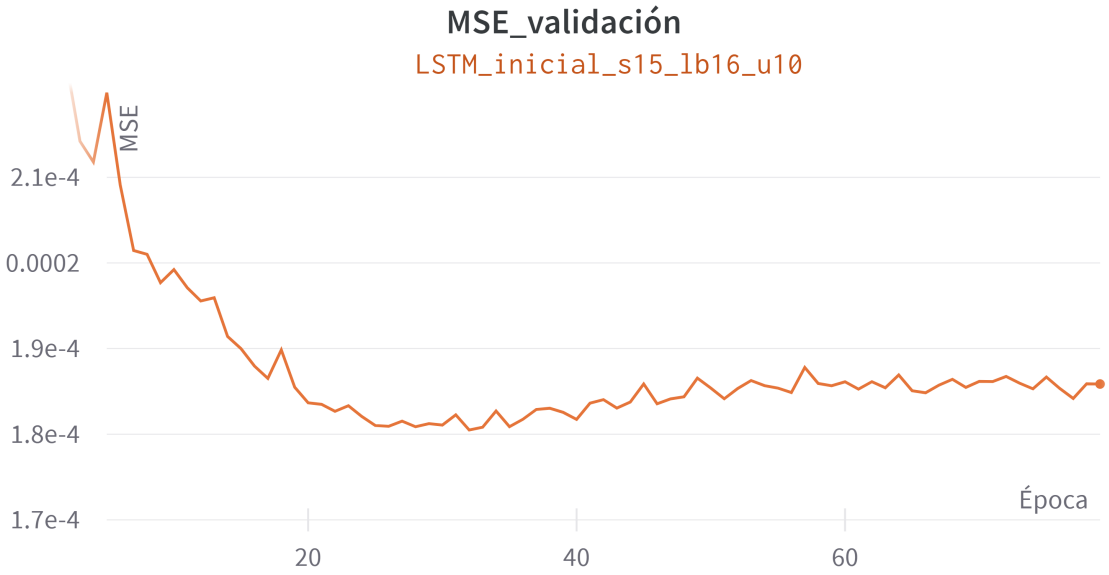

Figura 6.26: Experimento 0: 80 épocas

Tras esto, se realizan diferentes ejecuciones para determinar la función de activación a usar. En concreto, se seleccionan la tangente hiperbólica, la sigmoide y la lineal. Como se puede observar en la Figura 6.27, el modelo entrenado con sigmoide es el que presenta una curva más regular y con mayor tasa de decrecimiento.

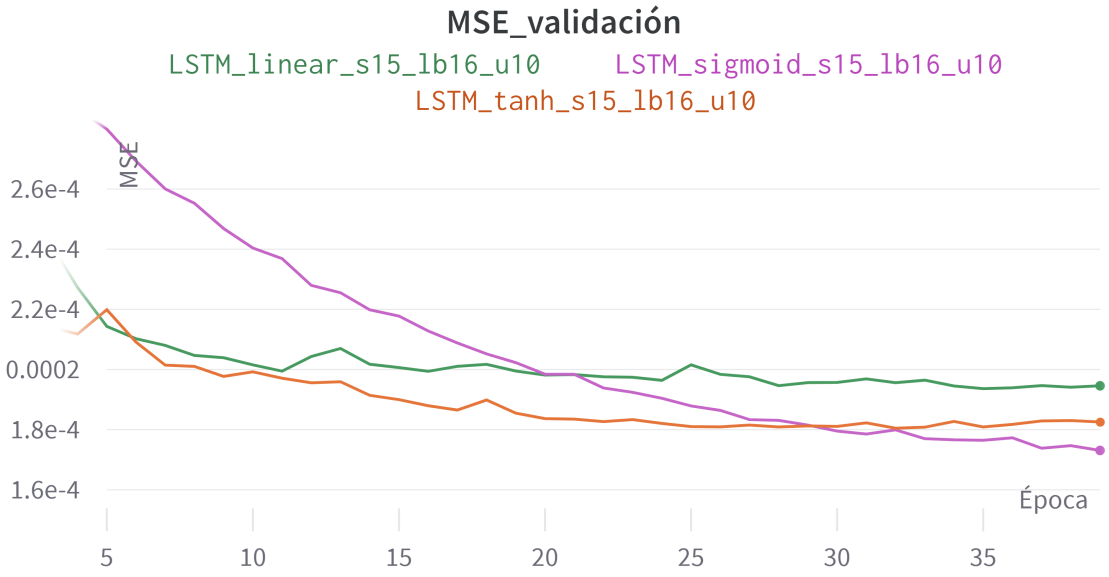

Figura 6.27: Experimento 0: función de activación

Por otro lado, una vez fijada la función sigmoide, se realizan algunas pruebas con diferente número de unidades de procesamiento en la capa LSTM. En concreto, se ejecutan experimentos con 10, 20, 30 y 40 unidades. Como se puede observar en la Figura 6.28, los resultados mejoran según se aumenta el número de unidades hasta alcanzar las 30, momento en el que los valores comienzan a ser peores.

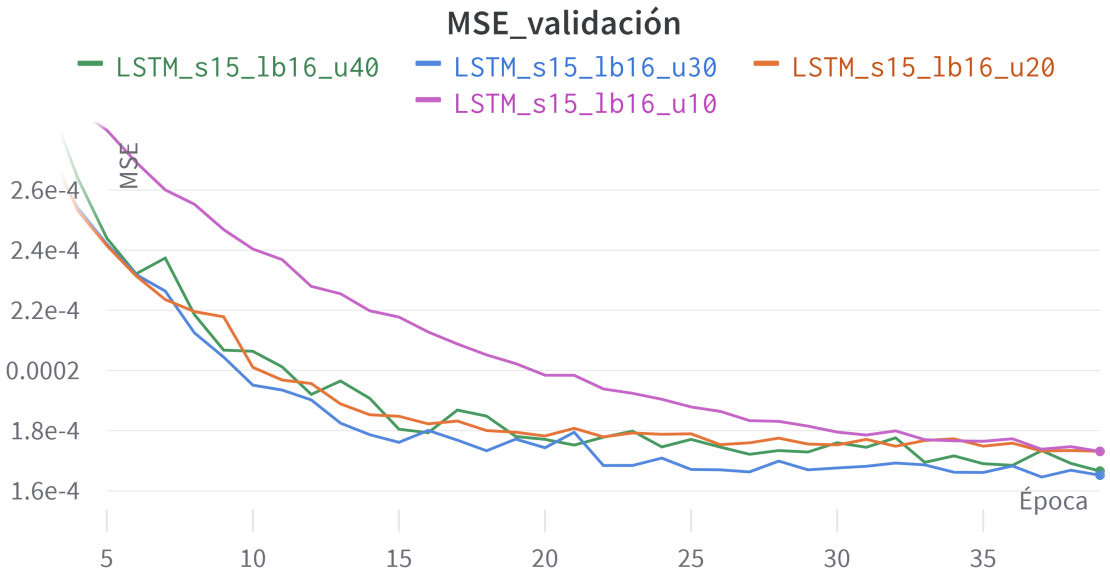

Figura 6.28: Experimento 0: unidades de procesamiento

Por último, se ejecuta una prueba para observar el comportamiento de la red al variar el tamaño de la ventana deslizante, probando con valores de 24, 32, 40, 48 y 56. De la Figura 6.29 se puede deducir que el modelo mejora su desempeño a medida que se aumenta el tamaño de la ventana, teniendo como punto de inflexión 48 registros, momento en el cual comienza a presentar peores valores de métrica.

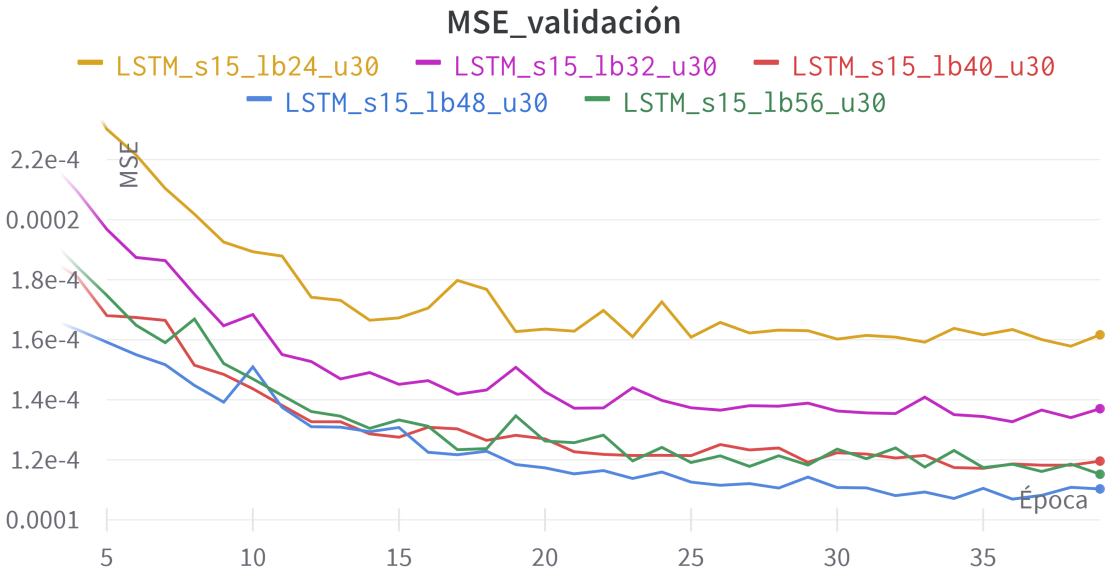

Figura 6.29: Experimento 0: tamaño de la ventana

Experimento 1: primeras pruebas con el *dataset* completo

A partir de los resultados obtenidos en la fase inicial, se realizan pruebas análogas con el conjunto de datos al completo (de enero a septiembre). No obstante, se tienen en cuenta las aproximaciones y conclusiones obtenidas, de forma que se acotan los rangos de prueba. De esta forma, se ha realizado el grueso de experimentos con solo un mes de datos, extendiéndose el tiempo de entrenamiento de cada experimento durante unos 40 minutos, y ahora se realizan pruebas más concretas con el total, alargándose a partir de ahora todas las ejecuciones a unas 6 o 7 horas por entrenamiento.

De aquí en adelante, se fija la sigmoide como función de activación y se mantiene el número de épocas a 40 en vista de la buena tendencia presente en el experimento anterior. Lo que si variará, en este caso, es el tamaño de ventana y el número de unidades en la capa LSTM.

En primer lugar, se realizan pruebas con números de unidades de procesamiento próximos al obtenido en el Experimento 0. Las curvas y valores obtenidos pueden visualizarse en la Figura 6.30 y en la Tabla 6.3.

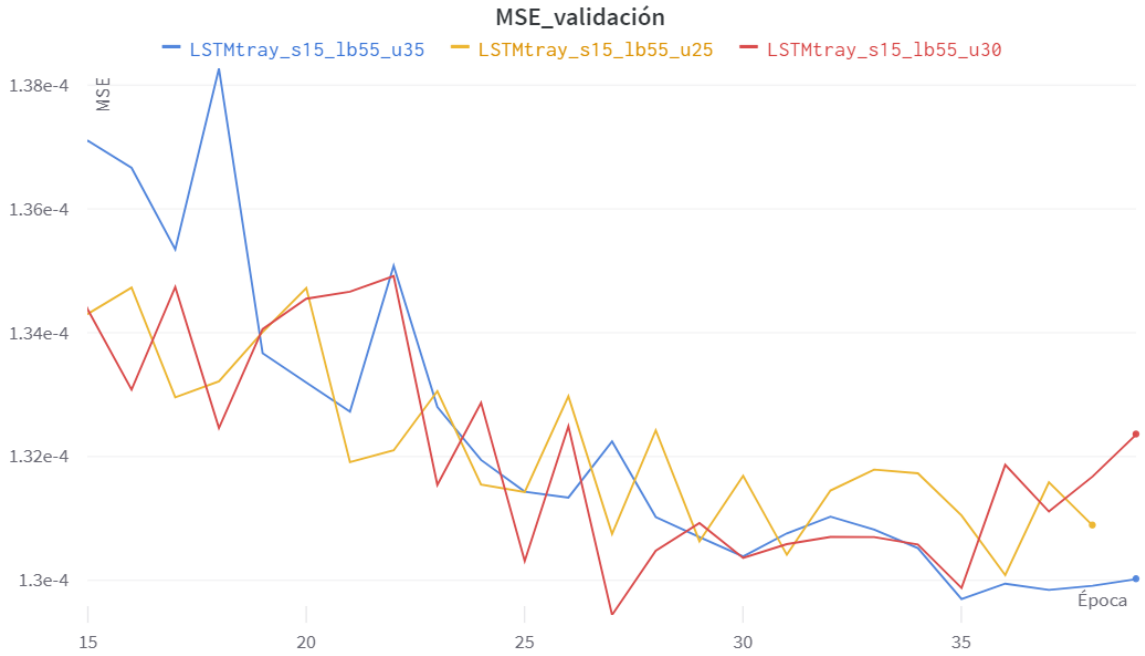

Figura 6.30: Experimento 1: unidades de procesamiento

Modelo	Unidades	Mejor época	Mejor MSE en validación
LSTMtray_s15_lb55_u25	25	37	$1{,}30 \cdot 10^{-4}$
LSTMtray_s15_lb55_u30	30	28	$1{,}29 \cdot 10^{-4}$
LSTMtray_s15_lb55_u35	35	37	$1{,}31 \cdot 10^{-4}$

Tabla 6.3: Evaluación de modelos que varían el número de unidades de procesamiento

Como se puede observar, al usar 30 unidades no solo se obtiene el mejor valor de MSE, sino que, además, este modelo parece converger con mayor velocidad que el resto.

Por otro lado, se completan ejecuciones similares para el tamaño de ventana (ver Figura 6.31), obteniéndose el mejor resultado con 55 registros (ver Figura 6.31 y Tabla 6.4).

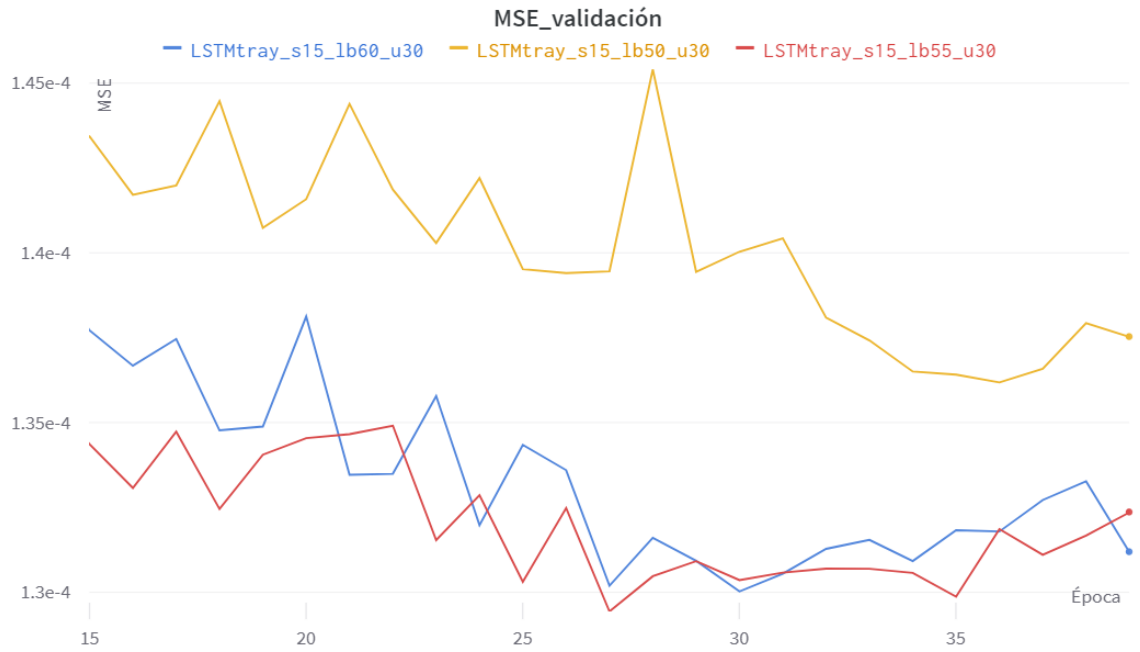

Figura 6.31: Experimento 1: tamaño de la ventana

Modelo	Unidades	Mejor época	Mejor MSE en validación
LSTMtray_s15_lb50_u30	50	37	$1{,}36 \cdot 10^{-4}$
LSTMtray_s15_lb55_u30	55	28	$1{,}29 \cdot 10^{-4}$
LSTMtray_s15_lb60_u30	60	31	$1{,}30 \cdot 10^{-4}$

Tabla 6.4: Evaluación de modelos que varían el número de unidades de procesamiento

Las curvas obtenidas durante el proceso de entrenamiento muestran que la disminución de la ventana de entrenamiento por debajo de 50 unidades impacta muy negativamente en los resultados obtenidos. Por otro lado, el aumento de ese mismo valor por encima de 60 presenta una tendencia similar a la obtenida con 55. Por ese motivo, se selecciona el valor inferior para disminuir la complejidad de entrenamiento del modelo.

Experimento 2: predicción acumulada frente a predicción directa

Una vez fijados los hiperparámetros según los resultados obtenidos en la Experimentación 1, la siguiente decisión a tomar es si optar por un modelo que genere como salida los diez puntos que se pretenden predecir (predicción directa) o por uno que genere un único punto y construir mediante la inclusión iterativa del punto predicho a la ventana deslizante los otros nueve (predicción iterativa). El funcionamiento de los modelos puede consultarse en la Figura 6.32.

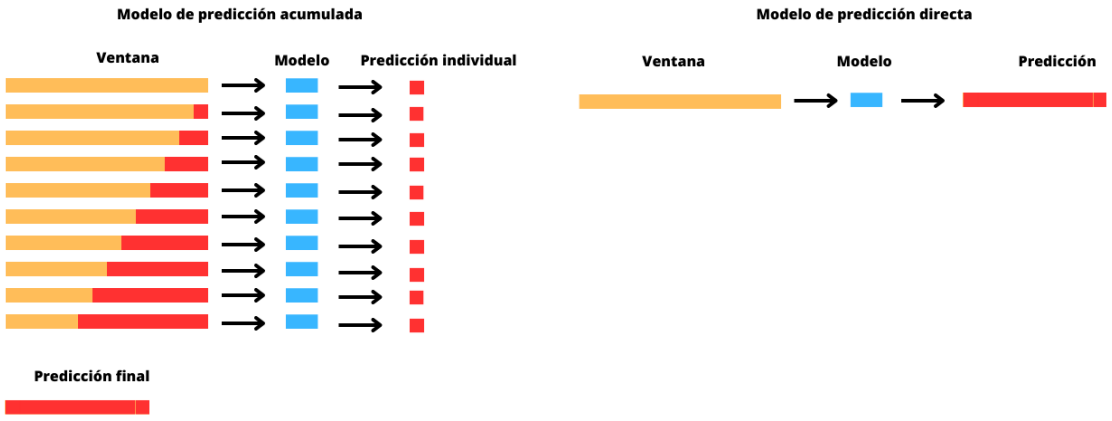

Figura 6.32: Esquema del modelo de predicción acumulada y directa

Por ello, se entrenan dos modelos, uno para predecir 10 valores a futuro (lookforward 10, modelo directo), y otro para predecir solo 1 (lookforward 1, modelo acumulado). De esta forma, a la hora de evaluar (sobre el conjunto de validación) el modelo iterativo, se construye una función para calcular los 10 puntos futuros como se indica en la Figura 6.32 y, posteriormente, se calcula el MSE. Los resultados obtenidos en entrenamiento pueden observarse en la Figura 6.33

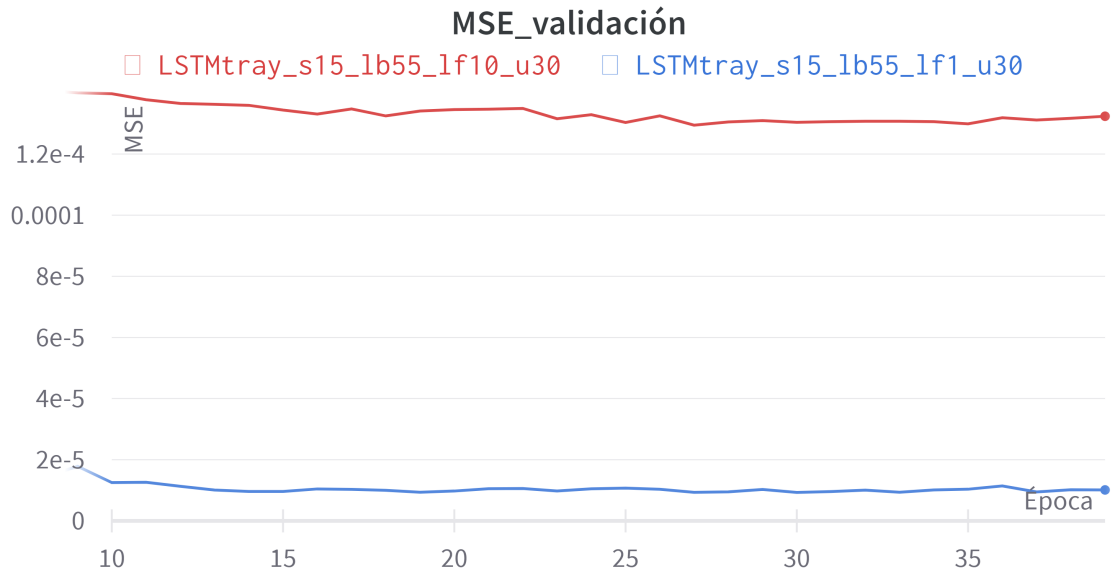

Figura 6.33: Experimento 2: predicción acumulada frente a predicción directa

Como se puede ver, el MSE difiere en un orden de magnitud dada la diferente complejidad de predicción de los modelos. No obstante, al evaluar sobre el conjunto de validación

utilizando el mecanismo descrito arriba, se obtienen los siguientes resultados (ver Tabla 6.5):

Tipo de predicción	Mejor época	Mejor MSE en validación
Acumulada	26	$4{,}40 \cdot 10^{-3}$
Directa	28	$1{,}29 \cdot 10^{-4}$

Tabla 6.5: Evaluación del modelo de predicción acumulada y directa

Es claro, en vista de los resultados obtenidos, que el modelo que realiza directamente las 10 predicciones futuras se comporta mejor que el que las calcula de forma acumulada. Esto parece deberse, de manera intuitiva, a que el último modelo acumula el error cometido en predicciones anteriores. Por ello, a partir de ahora solo se tendrán en cuenta los modelos de predicción directa.

Experimento 3: características adicionales

El siguiente paso en la fase de experimentación consiste en determinar si el uso de características adicionales ayuda al modelo a mejorar las predicciones. Por ello, se incorporan algunas combinaciones de atributos en base al conocimiento y entendimiento del caso de estudio. Las características seleccionadas son: *track*, aeropuerto de origen, distancia a destino y velocidad horizontal.

Es importante mencionar que se aplica una pequeña transformación sobre el *track*, que originalmente está expresado en grados y que, por tanto, no permitirá al modelo inferir correctamente la similitud entre los 0 y 359 grados. Por ello, se decide aplicar la función seno al *track*. En concreto, se aplica la función $sen(\frac{x}{2})$, que permite que la función se mantenga siempre en la rama positiva (ver Figura 6.34).

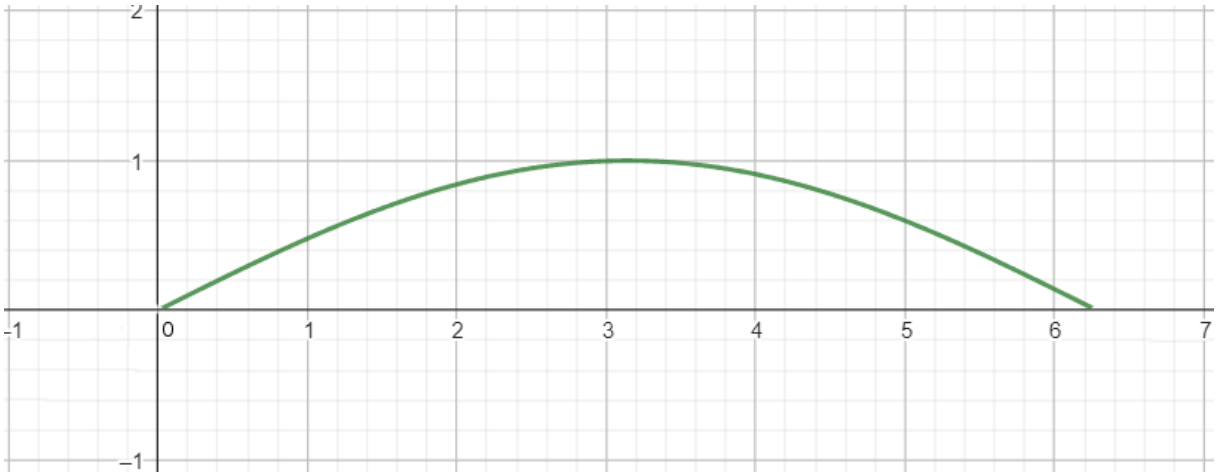

Figura 6.34: Función $sen(\frac{x}{2})$ en el periodo $[0, 2\pi]$

Dada la especial relevancia que, en base a la experiencia previa del proyecto, parecen tener el *track* y el aeropuerto de origen, se realizarán experimentos añadiendo cada uno de estos por separado para, finalmente, añadirlos junto con las otras características mencionadas. Los resultados obtenidos, junto con el caso que no incluye ninguna característica, pueden observarse en la Figura 6.35 y la Tabla 6.6.

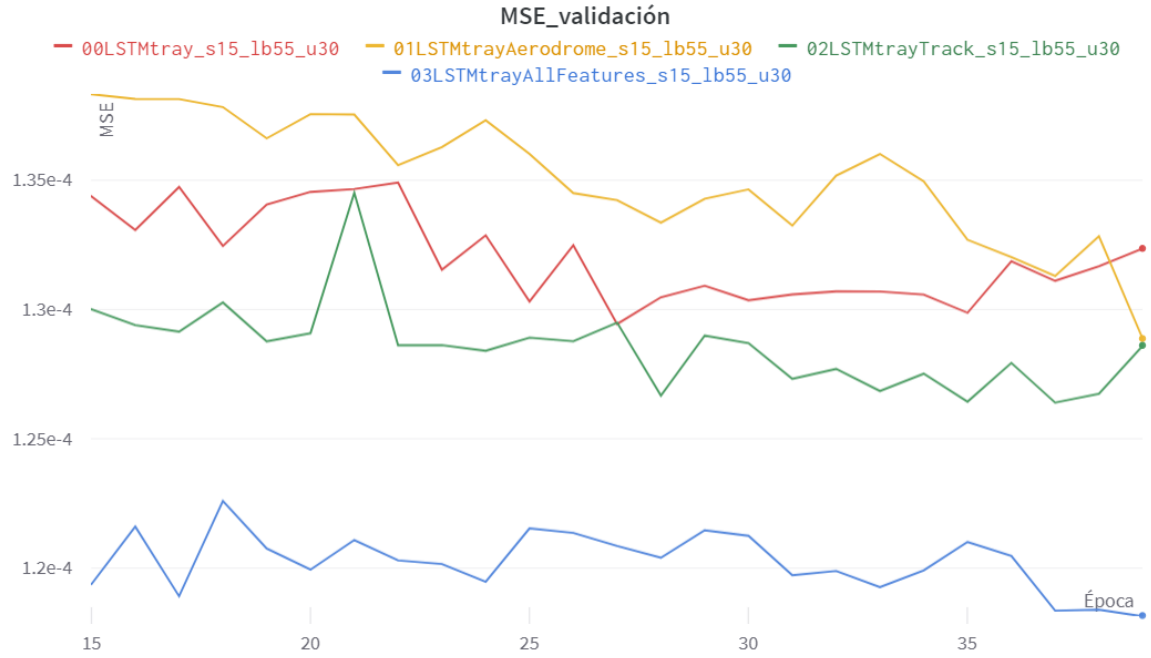

Figura 6.35: Experimento 3: características adicionales

Modelo	Features	Mejor época	Mejor MSE en validación
00LSTMtray	Latitud, longitud y altitud	28	$1{,}29 \cdot 10^{-4}$
01LSTMtrayAerodrome	Latitud, longitud, altitud y aeropuerto de origen	40	$1{,}29 \cdot 10^{-4}$
02LSTMtrayTrack	Latitud, longitud, altitud y *track*	38	$1{,}26 \cdot 10^{-4}$
03LSTMtrayAllFeatures	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino y velocidad horizontal	40	$1{,}18 \cdot 10^{-4}$

Tabla 6.6: Resultados del experimento 3

Como se puede observar, todas las características, tanto por separado, como combinadas parecen aportar información de relevancia al modelo. En concreto destacan el *track* y, sobre todo, la incorporación de todas las características consideradas que disminuyen considerablemente el error cometido.

Experimento 4: inclusión del sector

Partiendo de los progresos obtenidos en experimentos anteriores y con la intención de mejorar aún más el desempeño del modelo, se introduce en esta sección una nueva característica construida a partir del *track*. El problema básico que se pretende solventar es la no biyectividad del seno, que puede dificultar el aprendizaje de la red. Por ello, se añade una nueva característica binaria denominada "sector" definida de la siguiente manera:

- sector $= 0$ si $track \in [0, 180)$
- sector $= 1$ si $track \in [180, 360)$

De esta forma, combinando ambas características (el *track* con la transformación $sin(\frac{x}{2})$ descrita en el apartado anterior y el sector) el modelo podría distinguir de forma única cada ángulo que forme el morro del avión. Además, esto tiene otro factor muy importante a su favor, y es que permite diferenciar diferentes partes de la trayectoria. Por las características particulares que posee el conjunto de datos con el que se trabaja , formado por vuelos con origen en diferentes ciudades de Europa y destino el aeropuerto Adolfo Suárez Madrid-Barajas, en la mayoría de los casos (salvo algunas excepciones puntuales en las que el aeropuerto considerado presenta una latitud inferior a la del aeropuerto español) el sector valdrá 1 en la fase de crucero de la trayectoria (puesto que España se sitúa al suroeste del continente) y 0 en las fases de maniobras cercanas a los aeropuertos (ya que para encarar las pistas en la configuración preferente se hacen giros de incluso 180 grados). Una representación de trayectoria procedente de un aeropuerto alemán en la Figura 6.36.

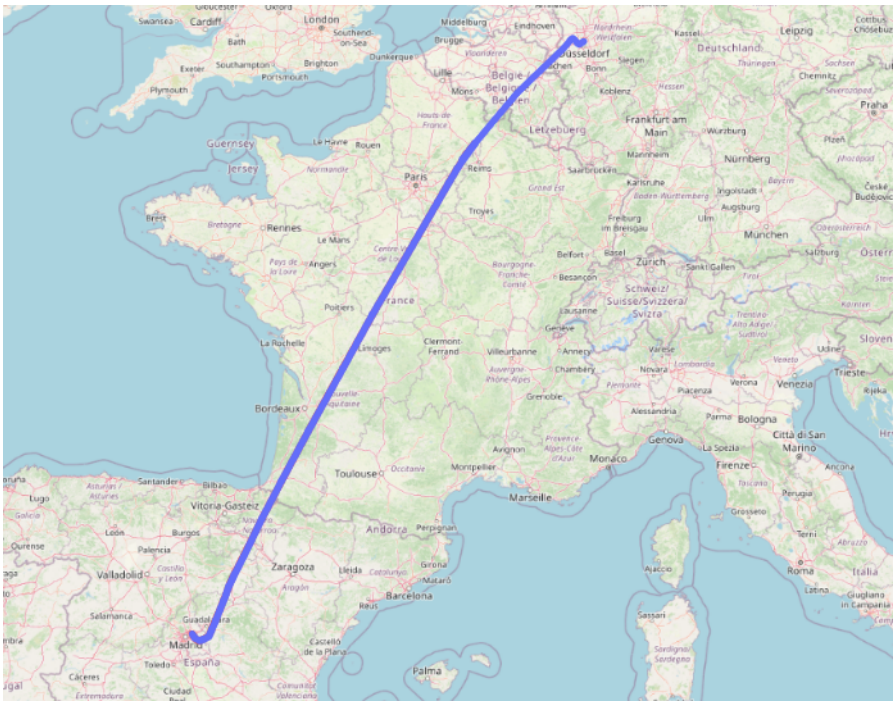

Figura 6.36: Ejemplo de trayectoria con origen en Alemania

En este caso, se han realizado dos pruebas independientes con las dos mejores combinaciones de características obtenidas en el experimento anterior. Los resultados pueden consultarse en la Figura 6.37 (en la que también se muestran los resultados anteriores para facilitar la comparación) y en la Tabla 6.7.

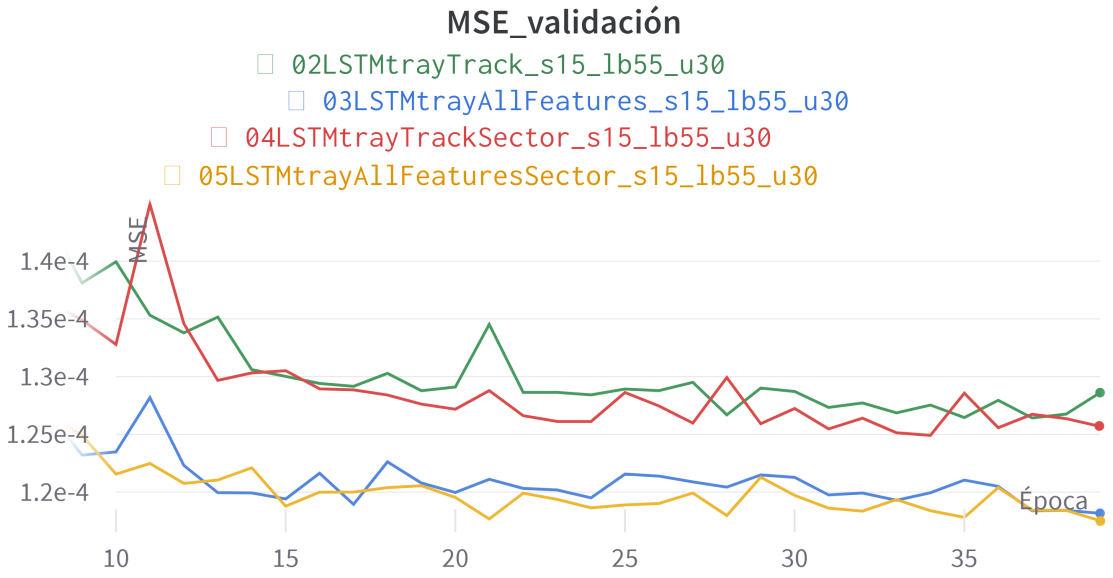

Figura 6.37: Experimento 4: inclusión del sector

Modelo	Features	Mejor época	Mejor MSE en validación
04LSTMtrayTrackSector	Latitud, longitud, altitud, *track* y sector	35	$1{,}25 \cdot 10^{-4}$
05LSTMtrayAllFeaturesSector	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino, velocidad horizontal y sector	40	$1{,}17 \cdot 10^{-4}$

Tabla 6.7: Resultados del experimento 4

Como se puede ver, en ambos casos se obtiene una ligera mejora en los mejores valores de MSE en validación, por lo que parece que la inclusión del sector permite aumentar la precisión del modelo. El mejor modelo obtenido con la arquitectura LSTM es el 05, que posee la siguiente configuración (ver Tabla 6.8):

Función de activación	Unidades en LSTM	Tamaño de ventana	Características
Sigmoide	30	55	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino, velocidad horizontal y sector

Tabla 6.8: Valores del modelo 05

Una vez concluida la experimentación con la red LSTM sencilla se consulta la herramienta de representación de trayectorias para analizar, sobre el conjunto de validación, cómo se comportan los modelos entrenados. Todos ellos tienen una característica en común, y es que generan puntos muy dispersos que derivan en trayectorias poco regulares. Un ejemplo del problema descrito puede consultarse en la Figura 6.38, donde se ha generado una predicción con el modelo 05.

Figura 6.38: Ejemplo de predicción con el modelo 05

LSTM añadiendo dos capas densas

Con el objetivo de continuar con la optimización del modelo, se tomó la decisión de probar una nueva arquitectura basada en la anterior, pero añadiendo dos capas densas. Con ello se pretende, como indica la literatura consultada [42] [26], que la red sea capaz de suavizar la tendencia de las trayectorias, realizando una interpolación cuadrática y disminuyendo, por tanto, la dispersión de los puntos a lo largo de la predicción. Por otro lado, se utiliza el conocimiento obtenido con los experimentos realizados sobre la red LSTM sencilla con el fin de reducir el volumen de experimentos a realizar.

Experimento 5: prueba inicial con LSTM + densas

El primer paso es realizar un primer experimento sencillo para comparar el desempeño de la nueva arquitectura frente a la anterior. Para ello, se entrena un modelo con las características básicas, cuyos resultados se muestran junto con los del mejor modelo obtenido en el Experimento 1 en la Figura 6.39 y en la Tabla 6.9.

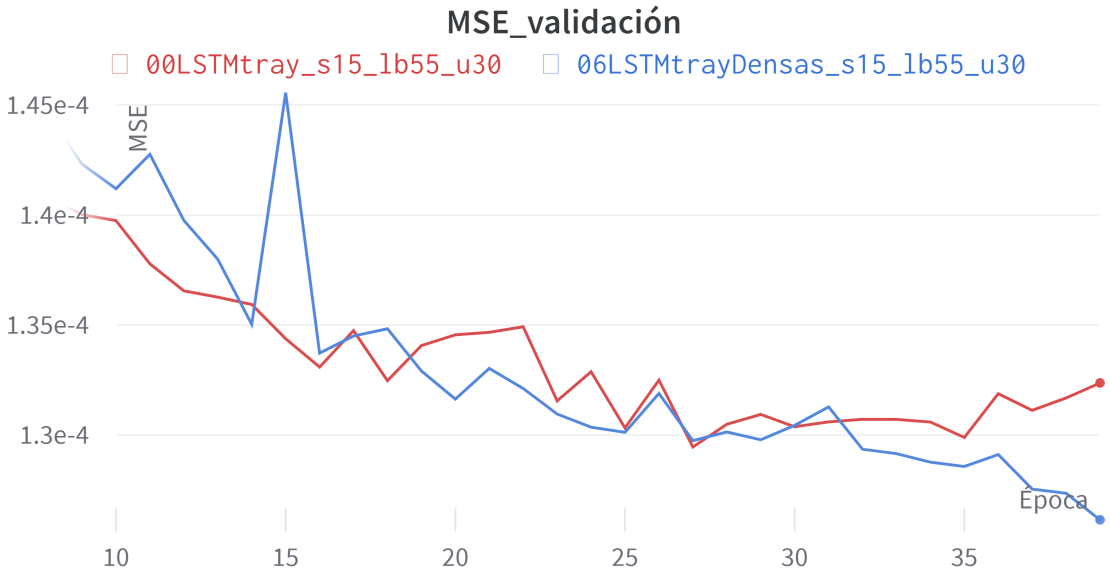

Figura 6.39: Experimento 5: prueba inicial con LSTM + densas

Modelo	Features	Mejor época	Mejor MSE en validación
00LSTMtray	Latitud, longitud y altitud	28	$1{,}30 \cdot 10^{-4}$
06LSTMtrayDensas	Latitud, longitud y altitud	40	$1{,}26 \cdot 10^{-4}$

Tabla 6.9: Resultados del experimento 5

En vista de los resultados y, en concreto, de la tendencia final de las curvas a partir de la época 35, se puede deducir que la incorporación de las capas densas mejora las predicciones generadas por el modelo.

Experimento 6: inclusión de características

Para concluir las pruebas con esta arquitectura y, en vista de los buenos resultados obtenidos anteriormente, se llevan a cabo 4 experimentos con las mejores combinaciones de hiperparámetros obtenidas con la red LSTM sencilla. En concreto las versiones con el *track*, el *track* y el sector, todas las características destacadas en el Experimento 3 y todas

las características del Experimento 3 junto con el sector. Los resultados obtenidos pueden consultarse en la Figura 6.40 y en la Tabla 6.10.

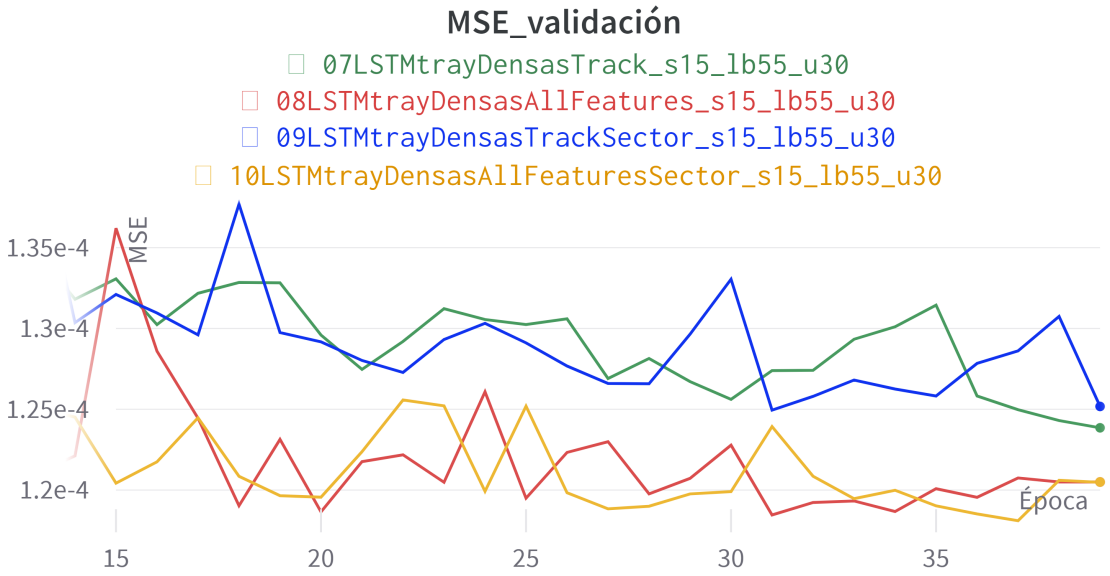

Figura 6.40: Experimento 6: inclusión de características

Modelo	Features	Mejor época	Mejor MSE en validación
07LSTMtrayDensasTrack	Latitud, longitud, altitud y *track*	40	$1{,}24 \cdot 10^{-4}$
08LSTMtrayDensasAllFeatures	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino y velocidad horizontal	32	$1{,}19 \cdot 10^{-4}$
09LSTMtrayDensasTrackSector	Latitud, longitud, altitud, *track* y sector	32	$1{,}24 \cdot 10^{-4}$
10LSTMtrayDensasAllFeaturesSector	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino, velocidad horizontal y sector	38	$1{,}17 \cdot 10^{-4}$

Tabla 6.10: Resultados del experimento 6

En este caso, el mejor modelo obtenido con la red LSTM extendida con las capas densas se obtiene en el modelo 10 con la siguiente configuración (ver Tabla 6.11):

Función de activación	Unidades en LSTM	Tamaño de ventana	Características
Sigmoide	30	55	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino, velocidad horizontal y sector

Tabla 6.11: Valores del modelo 10

Una vez concluida la experimentación con las redes LSTM con capas densas, es necesario comprobar nuevamente en la herramienta de visualización si estos modelos realizan predicciones más regulares como se pretendía. Repitiendo la misma prueba con el modelo 10 puede comprobarse que, efectivamente, los puntos generados han disminuido notablemente su grado de dispersión. Un ejemplo de esto puede observarse en la Figura 6.41.

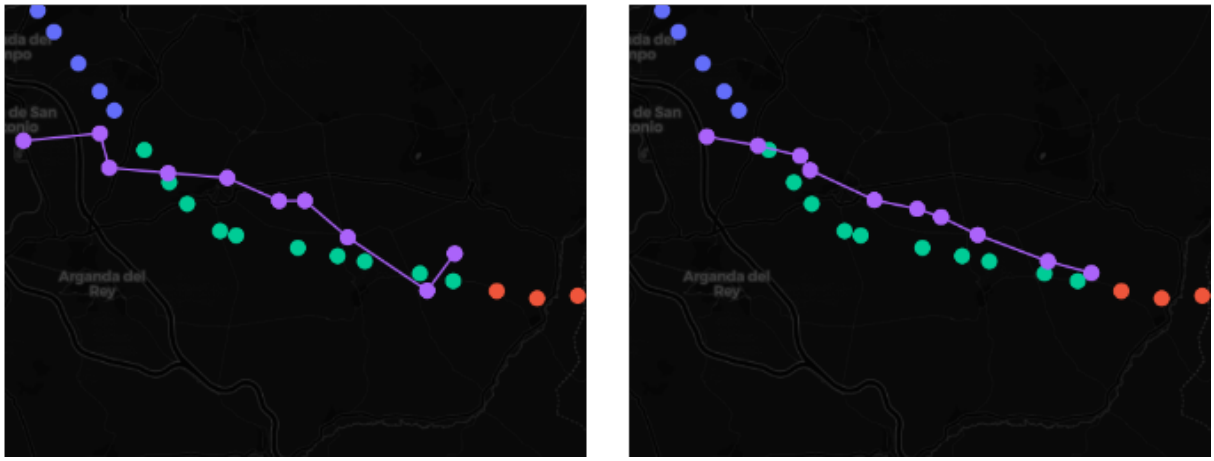

Figura 6.41: Ejemplo de predicción realizada por el modelo 05 frente a la obtenida con el modelo 10

Temporal Fusion Transformer

Tras haber realizado las pruebas con diferentes versiones de redes LSTM, y con la intención de experimentar con algunas arquitecturas más modernas, se realiza en este apartado una batería de experimentos con un Temporal Fusion Transformer. Para ello, se ha tomado como base un ejemplo de TFT implementado sobre Pytorch [8] y se ha adaptado al problema que se pretende resolver. Con este fin, no solo se han realizado transformaciones de formato sobre el conjunto de datos, sino que también se han modificado algunos de los parámetros que recibe la red para permitir la predicción simultánea de las tres variables objetivo (latitud, longitud y altitud).

Por otro lado, con el fin de simplificar la comprensión de las tendencias de las curvas de validación, en este apartado los resultados aparecen representados en escala logarítmica. Además, también es importante destacar que dadas las diferentes características de convergencia de las arquitecturas *transformers*, en este caso se establecerá un máximo de 120 épocas. No obstante, se añade un mecanismo de *early stopping*, es decir, se configura cada experimento de forma que si no se observa una cierta mejora en las métricas de validación en un periodo determinado, el entrenamiento se detiene. En concreto, se fija un valor mínimo de mejora de 10^{-4} en la pérdida sobre el conjunto de validación, que debe alcanzarse en menos de 5 iteraciones.

En este caso, los nombres de los modelos se ajustarán al siguiente patrón:

modelName_additionalInformation_attentionHeads_units_lookback

donde *modelName* se corresponderá con el tipo de modelo utilizado, *additionalInformation* puede aportar o no algún tipo de peculiaridad sobre el modelo, *attentionHeads* indica el número de cabezas de atención usadas, *units* especifica el número de unidades de la capa oculta y *lookback* hace referencia al tamaño de la ventana utilizada.

Experimento 7: optimización de hiperparámetros

La experimentación con la arquitectura TFT comienza con el ajuste de los hiperparámetros seleccionados en el apartado 6.1. En primer lugar, se optimiza el número de cabezas de atención (ver Figura 6.42 y Tabla 6.12).

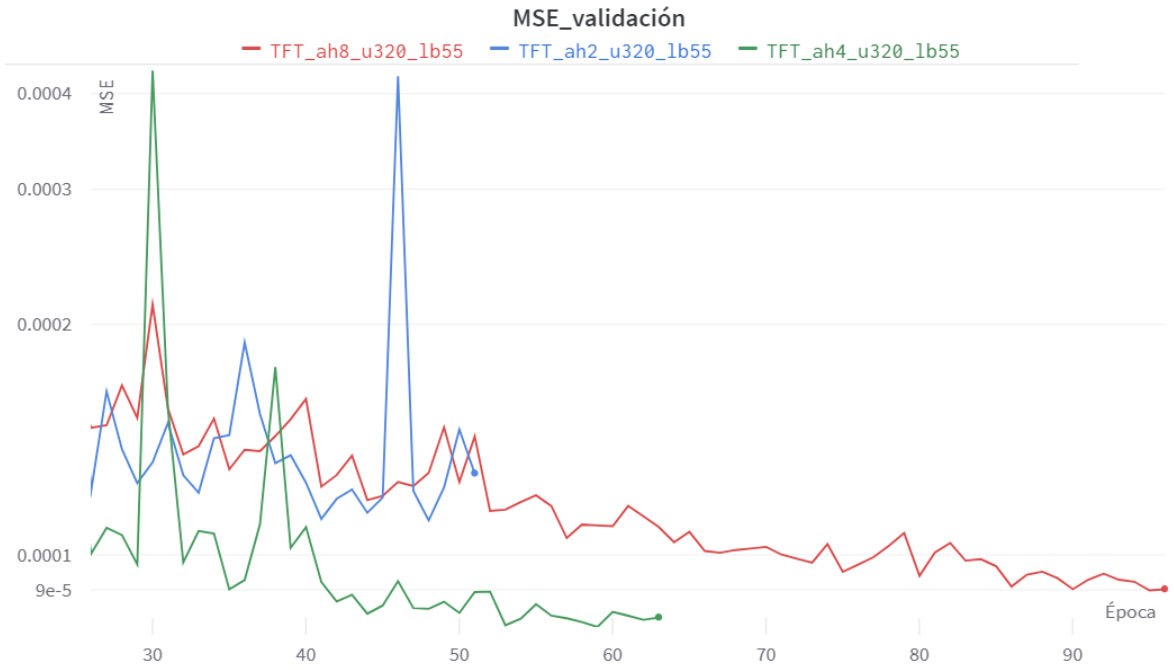

Figura 6.42: Experimento 7: cabezas de atención

Modelo	Cabezas de atención	Mejor época	Mejor MSE en validación
TFT_ah2_u320_lb55	2	49	$1{,}11 \cdot 10^{-4}$
TFT_ah4_u320_lb55	4	60	$8{,}06 \cdot 10^{-5}$
TFT_ah8_u320_lb55	8	96	$9{,}00 \cdot 10^{-5}$

Tabla 6.12: Experimento 7: cabezas de atención

En vista de los resultados obtenidos se puede concluir que el aumento del número de cabezas de atención hasta 8, aunque supone que las curvas presenten una tendencia más regular a lo largo de las épocas, arroja valores ligeramente inferiores que al utilizar la mitad de ellas. Por ello, se fija el número de cabezas de atención a 4.

El siguiente paso es determinar un buen valor para el número de unidades de procesamiento en la capa oculta. Se realizan pruebas con valores por encima y por debajo del valor por defecto en incrementos de 60 unidades. Los resultados obtenidos pueden consultarse en la Figura 6.43 y en la Tabla 6.13.

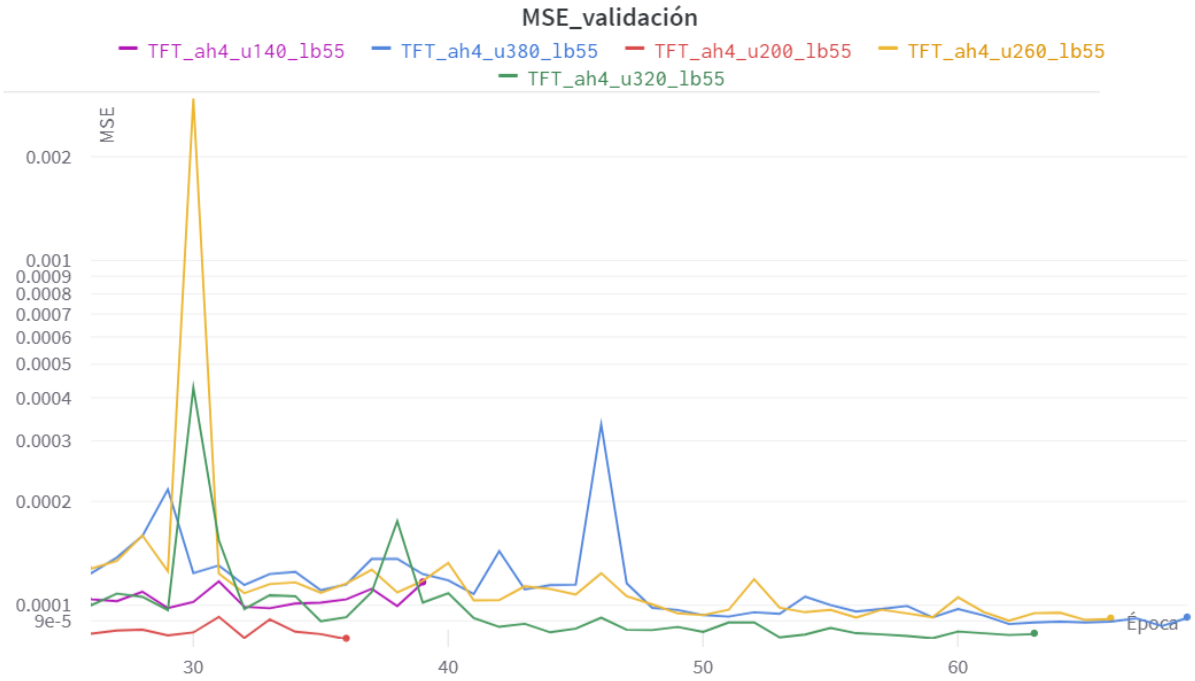

Figura 6.43: Experimento 7: unidades de procesamiento

Modelo	Unidades de procesamiento	Mejor época	Mejor MSE en validación
TFT_ah4_u140_lb55	140	34	$9{,}85 \cdot 10^{-5}$
TFT_ah4_u200_lb55	200	37	$8{,}02 \cdot 10^{-5}$
TFT_ah4_u260_lb55	260	63	$9{,}07 \cdot 10^{-5}$
TFT_ah4_u320_lb55	320	60	$8{,}06 \cdot 10^{-5}$
TFT_ah4_u380_lb55	380	69	$8{,}75 \cdot 10^{-5}$

Tabla 6.13: Experimento 7: unidades de procesamiento

Los resultados muestran que el uso de 200 unidades de procesamiento no solo mejora los resultados, sino que también parece acelerar el número de iteraciones necesarias para la convergencia del modelo.

Para concluir este primer experimento con la arquitectura TFT, se realiza una nueva batería de pruebas variando, en este caso, el tamaño de la ventana de entrada a la red (ver Figura 6.44 y Tabla 6.14).

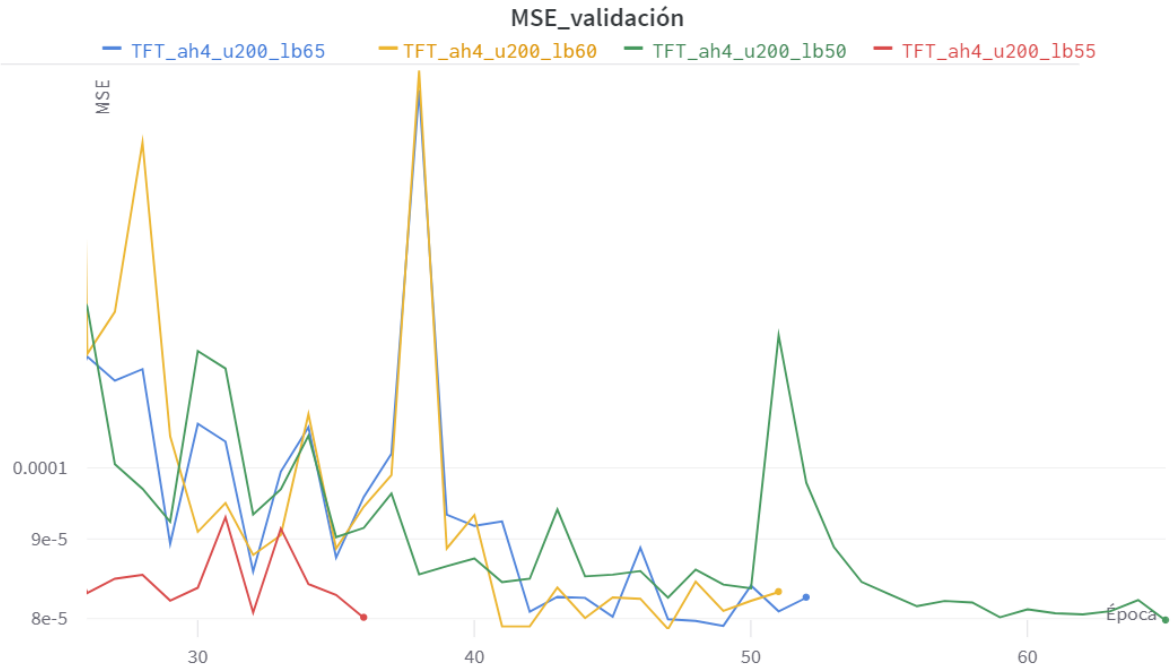

Figura 6.44: Experimento 7: tamaño de ventana

Modelo	Tamaño de ventana	Mejor época	Mejor MSE en validación
TFT_ah4_u200_lb50	50	66	$7{,}98 \cdot 10^{-5}$
TFT_ah4_u200_lb55	55	37	$8{,}02 \cdot 10^{-5}$
TFT_ah4_u200_lb60	60	47	$7{,}88 \cdot 10^{-5}$
TFT_ah4_u200_lb65	65	50	$7{,}92 \cdot 10^{-5}$

Tabla 6.14: Experimento 7: tamaño de ventana

En este caso, parece que un tamaño de ventana de 60 unidades permite mejorar los resultados obtenidos en el caso anterior. Esto resulta bastante curioso puesto que el *transformer* parece comportarse de manera similar a como lo hacían las redes LSTM vistas anteriormente (cuyo valor óptimo era de 55).

Experimento 8: inclusión de características

Para concluir con la fase de experimentación y, siguiendo la línea de lo realizado con las redes LSTM, se realizará un último conjunto de pruebas seleccionando las características mencionadas en los experimentos 3, 4 y 6. Los resultados obtenidos pueden observarse en las Figuras 6.45 y 6.46, y en la Tabla 6.15.

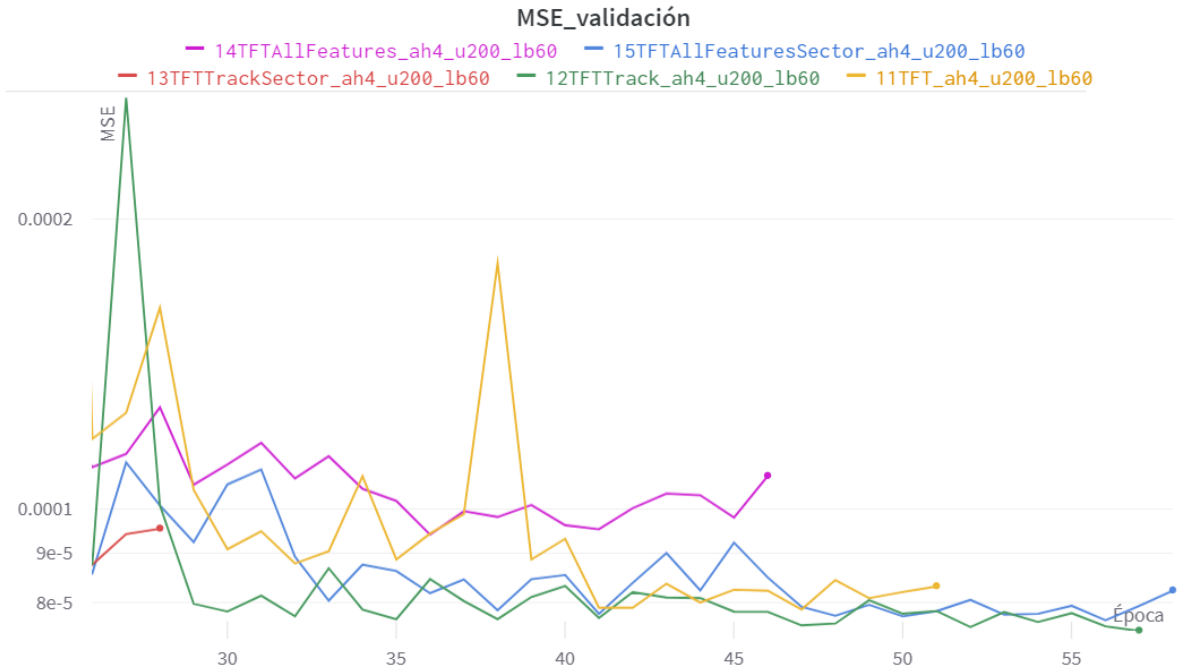

Figura 6.45: Experimento 8: inclusión de características

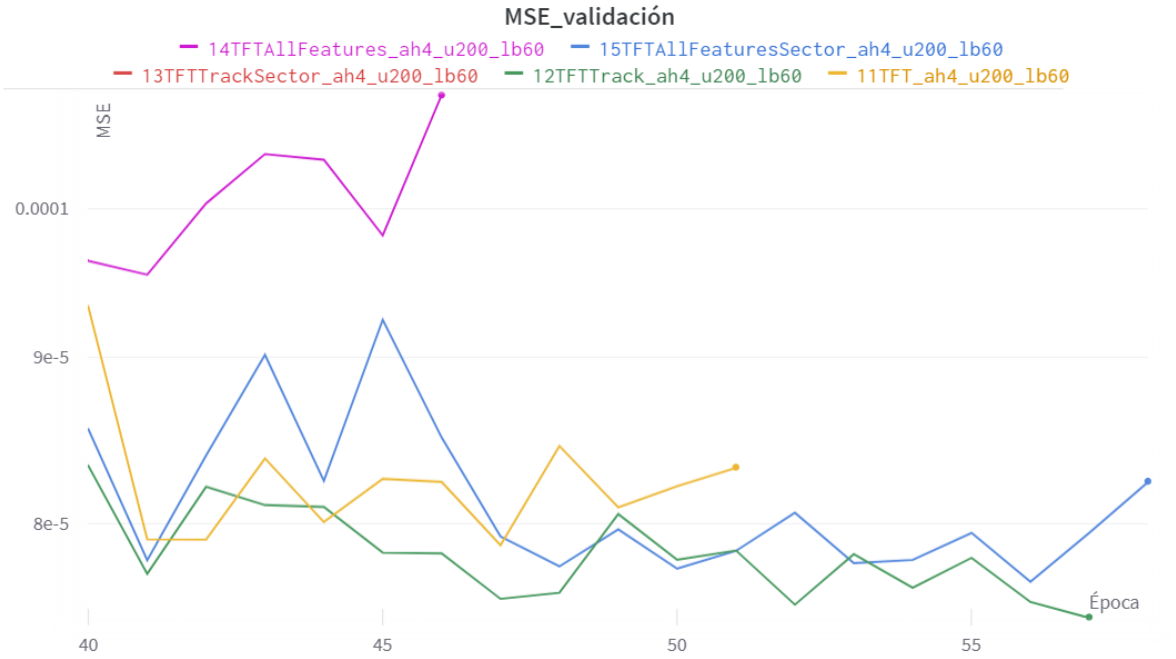

Figura 6.46: Experimento 8: inclusión de características. Zoom

Modelo	Características	Mejor época	Mejor MSE en validación
11TFT_ah4_u200_lb60	Latitud, longitud y altitud	47	$7{,}88 \cdot 10^{-5}$
12TFTTrack_ah4_u200_lb60	Latitud, longitud, altitud y *track*	58	$7{,}49 \cdot 10^{-5}$
13TFTTrackSector_ah4_u200_lb60	Latitud, longitud, altitud, *track* y sector	27	$8{,}77 \cdot 10^{-5}$
14TFTAllFeatures_ah4_u200_lb60	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino y velocidad horizontal	37	$9{,}42 \cdot 10^{-5}$
15TFTAllFeaturesSector_ah4_u200_lb60	Latitud, longitud, altitud, *track*, aeropuerto de origen, distancia a destino, velocidad horizontal y sector	57	$7{,}68 \cdot 10^{-5}$

Tabla 6.15: Experimento 8: inclusión de características

Como se puede observar, en este caso el uso de todas las características no parece ser de utilidad ya que el mejor valor en validación se obtiene al añadir únicamente el *track*. Se obtiene así el mejor modelo entrenado con la arquitectura TFT, que posee las siguientes propiedades (ver Tabla 6.16):

Cabezas de atención	**Unidades de procesamiento**	**Tamaño de ventana**	**Características**
4	200	60	Latitud, longitud, altitud y *track*

Tabla 6.16: Valores del modelo 15

7: Evaluación

En este capítulo se presenta la evaluación de los modelos seleccionados en el proceso de experimentación. Como se mencionó anteriormente, se mostrarán tres métricas parciales en escala real (una por cada característica a predecir) y una global con los datos normalizados para que sea comparable con las métricas de entrenamiento.

Los resultados obtenidos al evaluar sobre el conjunto de *test* pueden consultarse en la Tabla 7.17.

Modelo	**MSE latitud**	**MSE longitud**	**MSE altitud**	**MSE global**
05LSTMtrayAllFeaturesSector	$8{,}33 \cdot 10^{-3}$	$2{,}17 \cdot 10^{-2}$	$6{,}58 \cdot 10^{5}$	$1{,}17 \cdot 10^{-4}$
10LSTMtrayDensasAllFeaturesSector	$8{,}30 \cdot 10^{-3}$	$2{,}20 \cdot 10^{-2}$	$6{,}71 \cdot 10^{5}$	$1{,}19 \cdot 10^{-4}$
12TFTTrack	$5{,}46 \cdot 10^{-3}$	$2{,}90 \cdot 10^{-2}$	$3{,}85 \cdot 10^{5}$	$7{,}26 \cdot 10^{-5}$

Tabla 7.17: Resultados de evaluación

Los resultados obtenidos al calcular el MSE global con los valores escalados son muy similares a los obtenidos sobre el conjunto de validación durante el proceso de entrenamiento, por lo que los modelos parecen generalizar correctamente los patrones aprendidos sobre datos nuevos. Por otro lado, siguiendo la tendencia de la experimentación, la arquitectura TFT es la que presenta mejores resultados. Además, y en contra de la idea intuitiva inicial, las capas densas no solo no mejoran la métrica calculada, sino que dificultan la predicción de nuevos valores.

Los errores parciales cometidos en latitud, longitud y altitud no parecen fácilmente interpretables puesto que los sistemas de medida utilizados (grados y pies) pueden resultar poco familiares en la región europea. Por ello, teniendo en cuenta que 1^{o} equivale a unos 111,12 kilómetros y que el valor aproximado de un pie es de $3{,}048 \cdot 10^{-4}$ kilómetros, se obtienen los resultados de la Tabla 7.18.

Modelo	**MSE latitud**	**MSE longitud**	**MSE altitud**
05LSTMtrayAllFeaturesSector	102,23	267,80	0,061
10LSTMtrayDensasAllFeaturesSector	102,23	273,31	0,062
12TFTTrack	67,78	357,81	0,036

Tabla 7.18: MSE con kilómetro como unidad de medida

Por otro lado, con la finalidad de mejorar aún más la interpretabilidad de los resultados, se calculará el MAE sobre los valores desescalados, para poder tener una noción del error medio, medido en kilómetros, que comete cada uno de los modelos al hacer las predicciones. Esto puede consultarse en la Tabla 7.19.

Modelo	**MAE latitud**	**MAE longitud**	**MAE altitud**
05LSTMtrayAllFeaturesSector	2,40	4,05	0,074
10LSTMtrayDensasAllFeaturesSector	2,31	4,14	0,079
12TFTTrack	4,44	12,18	0,066

Tabla 7.19: MAE con kilómetro como unidad de medida

De esta última tabla parece poder conluirse que los modelos basados en redes LSTM presentan errores de mayor magnitud que los TFTs (puesto que su MSE era mayor), pero en menor cantidad que estos últimos (ya que el MAE es menor). Para comprobarlo se representan histográmas con los errores cuadráticos y absolutos de cada modelo, los cuales confirman la teoría planteada. Solo se mostrará un ejemplo (Figura 7.47) para no sobrecargar la memoria, aunque en todos los casos las figuras siguen la misma tendencia.

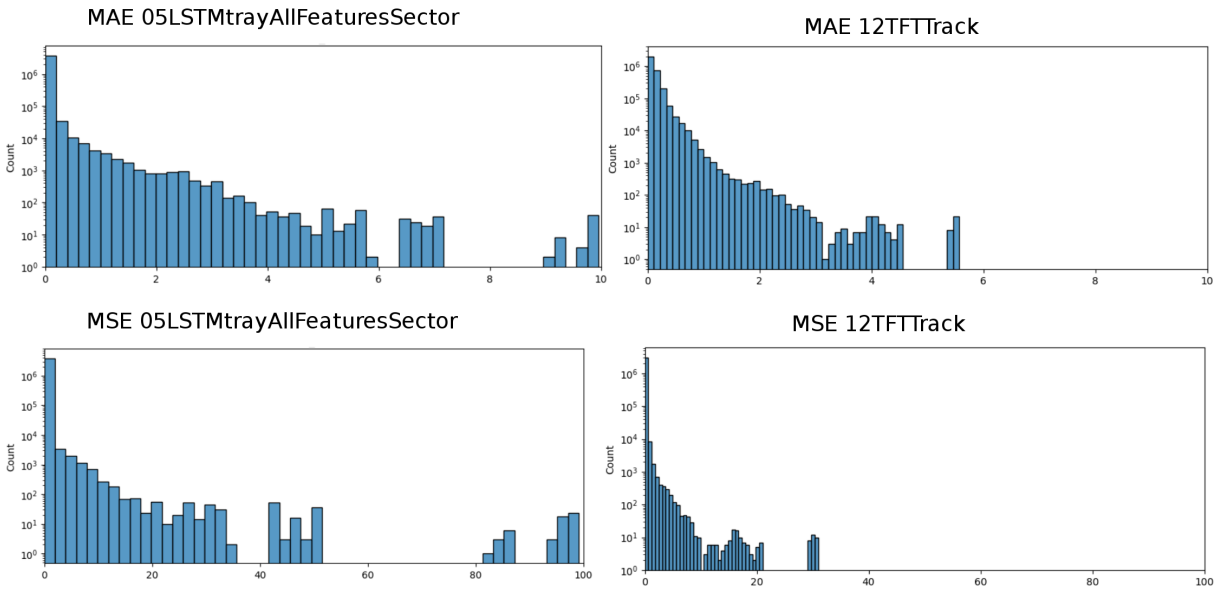

Figura 7.47: Histogramas de MAE y MSE sobre la longitud

En cualquier caso, los errores medios obtenidos en la Tabla 7.19 son considerablemente buenos si se tiene en cuenta que la velocidad horizontal media de una aeronave es de unos 14 kilómetros por minuto. Además, son especialmente reseñables los valores obtenidos para la altitud (que en todos los casos son inferiores a 100 metros), sobre todo considerando que las aerovías por las que transitan los aviones difieren entre sí verticalmente una distancia de un kilómetro. Por otro lado, parece razonable el hecho de que se aproximen mejor en todos los casos los valores de latitud que los de longitud, por las características concretas del conjunto de datos: las trayectorias abarcan en la mayoría de casos un mayor desplazamiento en el eje horizontal que en el vertical.

8: Conclusiones y líneas de trabajo futuro

Se presentan en este último capítulo las conclusiones obtenidas una vez finalizado el proyecto así como algunas posibles líneas de trabajo futuro.

8.1. Conclusiones

En primer lugar se analiza si se ha cumplido o no con los objetivos específicos propuestos.

- El Capítulo 4 documenta la revisión del estado del arte. En el mismo, se presentan diferentes arquitecturas y conjuntos de datos utilizados por otros equipos de investigación para la predicción de trayectorias aéreas (OBJ-1.1).
- En el Capítulo 6 se entrenan diferentes modelos LSTM (OBJ-1.2), optimizando algunos de sus hiperparámetros e incluyendo algunas características y transformaciones adicionales. Por otro lado, se realiza un proceso similar con la arquitectura TFT (OBJ-1.3).
- Por último, los modelos seleccionados en el Capítulo 6, se comparan y evalúan en el Capítulo 7 (OBJ-1.4), demostrando que los *transformers* arrojan mejores resultados que las arquitecturas LSTM.

De esta forma, englobando toda esta información, podemos afirmar la consecución del objetivo principal del proyecto OBJ-1.

Por otro lado, la elección de la metodología ASAP ha sido acertada, puesto que la planificación a corto plazo, junto con las reuniones de sincronización semanales con los tutores, han posibilitado tener una carga de trabajo regular a lo largo de los 6 *sprints* de aprendizaje. Todo ello ha permitido que el trabajo se haya completado en tiempo y forma.

Personalmente, este proyecto me ha permitido trabajar con tecnologías desconocidas para mí, como es el caso de Dash, o con otras que sí conocía, pero con las que nunca

había realizado ningún proyecto, como Pytorch. Por otro lado, he tenido la posibilidad descubrir soluciones novedosas no vistas en el Máster, como es el caso de los *transformers*. Me ha resultado tremendamente interesante conocer el funcionamiento de los mecanismos de atención en los que se basa, y tener la posibilidad de corroborar personalmente el alto grado de paralelización que presenta este tipo de arquitecturas, que redujeron los 50 minutos de ejecución por época en CPU a 40 segundos utilizando la GPU. Además, ha sido todo un reto adaptar el ejemplo de TFT del que se parte, al conjunto de datos con el que se ha trabajado, puesto que no se ha encontrado ninguna referencia que utilizase tres variables objetivo.

8.2. Trabajo futuro

Se plantean, a continuación, algunas posibles líneas de trabajo futuro que no han podido abordarse en el presente proyecto.

- En primer lugar, vistos los buenos resultados obtenidos con el Temporal Fusion Transformer, sería interesante probar otras arquitecturas basadas en mecanismos de atención como puede ser Informer [51] o DeepAR [44].
- Por otro lado, dados los requerimientos del problema que se pretende solventar, sería importante añadir la posibilidad de procesamiento de los datos en tiempo real, de forma que se pudiesen generar predicciones de forma casi inmediata.
- Finalmente, siguiendo la línea de algunas propuestas analizadas en el estado del arte, podrían implementarse algoritmos de *clustering* que permitiesen agrupar las rutas, de forma que se entrenasen varios modelos que se ajustaran mejor a las tendencias de las mismas.

Apéndices

Apéndice A

Manual de instalación

Se presenta en este primer apéndice los pasos necesarios para crear un entorno local de Anaconda, que permita la ejecución de los diferentes cuadernos generados en el proyecto. Este breve manual se realizará sobre un sistema basado en Linux por lo que, en caso de la máquina con que se desea trabajar no cumpla este requisito, se recomienda la creación de una máquina virtual que lo satisfaga. No se aportan detalles sobre este proceso puesto que puede consultarse en [5].

Una vez se cuente con una versión funcional de un sistema Linux, el primer paso es la actualización de repositorios.

```
$ sudo apt update
```

El siguiente paso es instalar Anaconda, para lo cual previamente se debe contar con el paquete *curl*.

```
$ sudo apt install curl -y
```

```
$ cd / tmp
```

```
$ curl -output anaconda.sh https://repo.anaconda.com/archive/Anaconda3-5.3.1-
```

```
Linux-x86_64.sh
```

Se puede verificar que la descarga se ha completado de manera satisfactoria comprobando el *checksum* del archivo, que debe coincidir con el indicado en el enlace de la versión.

```
$ sha256sum anaconda.sh
```

A continuación, se ejecuta el script que realiza la instalación del entorno.

```
$ bash anaconda.sh
```

Finalmente se activa el entorno y se abre la aplicación de Jupyter Notebook.

```
$ source  /.bashrc $ jupyter notebook
```

Apéndice B

Contenido adjunto

En este último apéndice se enumera el contenido adjunto a la memoria, disponible en el siguiente enlace de GitLab:

https://gitlab.inf.uva.es/paumiel/tfm_predicciontrayectoriasaereasdeeplearning/

- **LSTM**. Directorio que contiene tanto los scripts y cuadernos necesarios para llevar a cabo el proceso de experimentación con redes LSTM, como los mejores modelos obtenidos en el apartado 6.3. La carpeta principal (RTA-main) contiene los siguientes ficheros:
  - *data*. Carpeta vacía puesto que los datos no pueden proporcionarse en el repositorio de GitLab dado su carácter confidencial.
  - *models*. Mejores modelos obtenidos durante el entrenamiento.
  - *rtaUtils*. Carpeta que posee los *scripts* que definen las clases base del proyecto.
  - *0.1.preprocessing.ipynb*. Preparación de los datos: ordenación de vectores, limpieza de *outliers*, ...
  - *0.2.ML_utils*. Generación de *encoders* y *scalers* según el número de características seleccionadas.
  - *0.3.data_preparation.ipynb*. Generación de ventanas.
  - *0.4.sin_transformation.ipynb*. Transformación del *track* a seno e inclusión del sector.
  - *1.baseTrays.ipynb*. Cuaderno base sobre el que se entrenan los modelos.
  - *8.1.trajectory_prediction_dashboard.ipynb*. Dashboard de visualización.
- **TFT**. Contiene los cuadernos utilizados para el entrenamiento de modelos con la arquitectura TFT. Contiene los siguientes ficheros:
  - *data*. Carpeta vacía por motivo similar al de LSTM.

- *tft*. Contiene la lógica del modelo TFT que se utiliza en los cuadernos. Además, dentro de la carpeta *trayectorias_logs4*, se encuentra el mejor modelo obtenido en experimentación.
- *Environment_preparation.ipynb*. Creación de carpetas y descarga del repositorio tft necesario para la ejecución de entrenamientos.
- *TFT_trayectorias.ipynb*. Cuaderno base para la experimentación con la arquitectura TFT.

Bibliografía

[1] ADS-B in trail procedures (ITP). https://www.faa.gov/air_traffic/technology/adsb/pilot/itp. Último acceso: 2023-05-20.

[2] ADS-B technologies. http://www.ads-b.com/. Último acceso: 2023-05-20.

[3] Aeropuerto Madrid-Barajas Adolfo Suárez. https://www.aeropuertomadrid-barajas.com/. Último acceso: 2023-05-20.

[4] Control de tráfico aéreo (ATC). https://www.enaire.es/servicios/atm/servicios_de_transito_aereo_ats/control_de_trafico_aereo_atc. Último acceso: 2023-05-20.

[5] Cómo instalar Ubuntu 22.04 en VirtualBox. https://comoinstalar.me/como-instalar-ubuntu-22-04-lts-jammy-jellyfish-en-virtualbox/. Último acceso: 2023-07-08.

[6] Dash Documentation and User Guide | Plotly. https://dash.plotly.com/. Último acceso: 2023-07-03.

[7] El perceptrón como neurona artificial. Jose Mariano Álvarez. https://blog.josemarianoalvarez.com/2018/06/10/el-perceptron-como-neurona-artificial/. Último acceso: 2023-05-07.

[8] Github - andresc98/tsf_transformers_tfm: Repository containing my Master Thesis for the M.Sc. Big Data Analytics, titled "Time Series Forecasting with Transformers". https://github.com/andresC98/TSF_Transformers_TFM. Último acceso: 2023-06-24.

[9] Introducción a redes neuronales recurrentes (RNN). Mariano Rivera. http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/RNN_LTSM/introduccion_rnn.html. Último acceso: 2023-04-11.

[10] Lecture 16: Building blocks of deep learning. https://sailinglab.github.io/pgm-spring-2019/notes/lecture-16/. Último acceso: 2023-06-20.

[11] Métricas en regresión. Arrioja Landa, Nicolás. https://medium.com/@nicolasarrioja/metricas-en-regresion-5e5d4259430b. Último acceso: 2023-06-14.

[12] Primary Surveillance Radar (PSR). https://www.skybrary.aero/articles/primary-surveillance-radar-psr. Último acceso: 2023-07-07.

[13] Redes neuronales artificiales. https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i. Último acceso: 2023-05-07.

[14] Secondary Surveillance Radar (SSR). https://www.skybrary.aero/articles/secondary-surveillance-radar-ssr. Último acceso: 2023-07-07.

[15] Sueldo: Analista en julio, 2023. https://www.glassdoor.es/Sueldos/analista-sueldo-SRCH_KO0,8.htm. Último acceso: 2023-07-10.

[16] Sueldo: Data scientist en julio, 2023. https://www.glassdoor.es/Sueldos/data-scientist-sueldo-SRCH_KO0,14.htm. Último acceso: 2023-07-10.

[17] Sueldo: Programador en julio, 2023. https://www.glassdoor.es/Sueldos/programador-sueldo-SRCH_KO0,11.htm. Último acceso: 2023-07-10.

[18] Weights and Biases. https://wandb.ai/. Último acceso: 2023-07-05.

[19] ¿Cuánto cuesta contratar a un trabajador? https://getquipu.com/blog/cuanto-cuesta-contratar-un-trabajador/. Último acceso: 2023-01-21.

[20] Regulation (ec) no 552/2004 of the european parliament and of the council of 10 march 2004 on the interoperability of the european air traffic management network. *Regulation, SES Interoperability 96*, 31.3 (2004).

[21] Documento de regulación aeroportuaria 2022-2026. *Secretaría de Estado de Transportes. Movilidad y Agenda Urbana Secretaría General de Transportes y Movilidad. Dirección General de Aviación Civil* (2021).

[22] Informe Anual de Ruido. Aeropuerto Adolfo Suárez Madrid-Barajas. *Envirosuite* (2021).

[23] Eurocontrol forecast update 2023-2029. *Supporting European Aviation EUROCONTROL* (2023).

[24] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[25] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. The agile manifesto. *The Agile Alliance* (2001).

[26] Cheng, X., Khomtchouk, B., Matloff, N., and Mohanty, P. Polynomial regression as an alternative to neural nets. *arXiv preprint arXiv:1806.06850* (2018).

[27] Choi, H. C., Deng, C., and Hwang, I. Hybrid machine learning and estimation-based flight trajectory prediction in terminal airspace. *IEEE Access 9* (2021), 151186–151197.

[28] Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., and Campbell, J. P. Introduction to machine learning, neural networks, and deep learning. *Translational vision science & technology 9*, 2 (2020), 14–14.

[29] Cruz, I. B., Martínez, S. S., Abed, A. R., Ábalo, R. G., and Lorenzo, M. M. G. Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas 1*, 4 (2007), 48–57.

[30] de Transporte Aéreo. Área de Estudios Estratégicos y Análisis de Mercado. Dirección General de Aviación Civil, S. G. Informe anual tráfico en los aeropuertos españoles, 2021.

[31] Deng, C., Kim, K., Choi, H.-C., and Hwang, I. Trajectory pattern identification for arrivals in vectored airspace. In *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)* (2021), IEEE, pp. 1–8.

[32] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[33] Le, T.-H., Tran, P. N., Pham, D.-T., Schultz, M., and Alam, S. Short-term trajectory prediction using generative machine learning methods. *Proceedings of the ICRAT 2020 Conference, Tampa, FL, USA 15* (2020).

[34] Lim, B., Arik, S. Ö., Loeff, N., and Pfister, T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting 37*, 4 (2021), 1748–1764.

[35] Ma, L., and Tian, S. A hybrid cnn-lstm model for aircraft 4d trajectory prediction. *IEEE Access 8* (2020), 134668–134680.

[36] Marcos, R., Herranz, R., Vázquez, R. R., García-Albertos, P., and Cantú Ros, O. Application of machine learning for atm performance assessment–identification of sources of en-route flight inefficiency. *Proceedings of the Eighth SESAR Innovation Days, Salzburg, Austria* (2018), 3–7.

[37] Martínez-Prieto, M. A., Silvestre, J., Bregón, A., Baz, P., Gándara-González, C., Mielgo, P., and Peñas, I. Una metodología basada en prácticas ágiles para la realización de trabajos fin de grado. *Actas de las XXVIII JENUI 8* (2023).

[38] Meneses-Bautista, F. D., and Alvarado, M. Pronóstico del tipo de cambio usd/mxn con redes neuronales de retropropagación. *Research in Computing Science 139* (2017), 97–110.

[39] Neto, E. C. P., Baum, D. M., de Almeida, J. R., Camargo, J. B., and Cugnasca, P. S. Deep learning in air traffic management (atm): A survey on applications, opportunities, and open challenges. *Aerospace 10* (2023), 358.

[40] Pang, Y., and Liu, Y. Conditional generative adversarial networks (CGAN) for aircraft trajectory prediction considering weather effects. In *AIAA Scitech 2020 Forum* (2020), p. 1853.

[41] Pilehvar, M. T., and Camacho-Collados, J. *Embeddings in natural language processing: Theory and advances in vector representations of meaning.* Morgan & Claypool Publishers, 2020.

[42] Raturi, R. Large data analysis via interpolation of functions: Interpolating polynomials vs artificial neural networks. *American Journal of Intelligent Systems 8* (2018), 6–11.

[43] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[44] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting 36*, 3 (2020), 1181–1191.

[45] Shi, Z., Xu, M., and Pan, Q. 4-d flight trajectory prediction with constrained lstm network. *IEEE Transactions on Intelligent Transportation Systems 22* (2021), 7242–7255.

[46] Tablada, C. J., and Torres, G. A. Redes neuronales artificiales. *Revista de Educación Matemática 24*, 3 (2021).

[47] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[48] Viji, S., Kannan, R., and Jayalashmi, N. Y. Intelligent anomaly detection model for atm booth surveillance using machine learning algorithm : intelligent atm survillance model. *Proceedings - IEEE 2021 International Conference on Computing, Communication, and Intelligent Systems, ICCCIS 2021* (2021), 1007–1012.

[49] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 10 (1990), 1550–1560.

[50] Zeng, W., Quan, Z., Zhao, Z., Xie, C., and Lu, X. A deep learning approach for aircraft trajectory prediction in terminal airspace. *IEEE Access 8* (2020), 151250–151266.

[51] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (2021), vol. 35, pp. 11106–11115.