Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA
DE VALLADOLID

Máster en Ingeniería Informática
(No Presencial)

Estudio y Evaluación de Arquitecturas Deep Learning para la Predicción de Trayectorias de Vuelo

---

Alumno: Gustavo Cortés Jiménez

Tutores: Miguel A. Martínez Prieto
Aníbal Bregón Bregón

Fecha: 19 de julio de 2024

Estudio y Evaluación de Arquitecturas Deep Learning para la Predicción de Trayectorias de Vuelo

Gustavo Cortés Jiménez

19 de julio de 2024

A mis padres y hermano,
por siempre estar a mi lado.

"El 90 % de lo que se considera "imposible"
es, de hecho, posible.
El 10 % restante será posible
con el paso del tiempo y la tecnología".
– Hideo Kojima

"Los programadores del mañana
son los magos del futuro.
Parecerá que tienen poderes mágicos
en comparación con los demás".
– Gabe Newell

Resumen

El avión es uno de los medios de transporte más utilizados en la actualidad. Sin embargo, la gestión del tráfico aéreo es compleja, por lo que resulta esencial predecir con precisión las trayectorias de vuelo para mejorar la seguridad, eficiencia y puntualidad de los viajes. El Aprendizaje Profundo (*Deep Learning*) es clave para mejorar estas predicciones, debido al elevado volumen de datos empleados. La información se organiza en "trayectorias 4d", que tienen en cuenta el tiempo para posicionar a las aeronaves.

Este Trabajo Fin de Máster se enfoca en desarrollar modelos de *Deep Learning* novedosos para predecir trayectorias de vuelo con mayor precisión. Las arquitecturas modeladas son TSMixer y *Liquid Neural Networks* (LNN), concretamente *Closed-form Continuous-time Neural Networks* (CfC), con las que se han obtenido resultados competitivos en la predicción de trayectorias de vuelo. Junto a esto, se diseña e implementa un visor de trayectorias 4D para facilitar el análisis visual de los resultados obtenidos.

Palabras clave: Predicción de trayectorias de vuelo, Aprendizaje Profundo, TSMixer, Closed-form Continuous-time Neural Networks, Redes Neuronales Líquidas

Abstract

Air travel is one of the most widely used means of transportation nowadays. However, air traffic management is complex, making it essential to accurately predict flight paths to improve travel safety, efficiency and punctuality. Deep Learning plays a crucial role in improving these predictions, due to the large volume of data involved. Information is organized into "4D trajectories", which account for time to position aircraft.

This project focuses on developing novel Deep Learning models to predict flight trajectories with greater accuracy. The architectures explored are TSMixer and Closed-form Continuous-time Neural Networks (CfC), which have demonstrated competitive results in flight trajectory prediction. Additionally, a 4D trajectory visualization tool is designed and implemented to facilitate the visual analysis of the obtained results.

Keywords: Flight trajectory prediction, Deep Learning, TSMixer, Closed-form Continuous-time Neural Networks, Liquid Neural Networks

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Hoy en día, los aviones son una de las principales opciones para viajes a media y larga distancia por su velocidad, comodidad y seguridad. Sin embargo, la gestión de este tráfico aéreo resulta una tarea compleja que se dificulta aún más con el aumento del número de vuelos, especialmente en las regiones cercanas a los aeropuertos, donde se encuentra una mayor densidad de aeronaves compartiendo los recursos, que por la concentración geográfica resultan escasos en dicho contexto. Ejemplos de estos recursos son el uso del espacio aéreo (asegurando las debidas condiciones de seguridad), o la disponibilidad de pistas de aterrizaje y despegue, entre otros.

En este ámbito, la predicción precisa de las trayectorias de vuelo de las aeronaves resulta esencial para garantizar la seguridad, eficiencia y puntualidad en el transporte aéreo. Prever con exactitud cómo se desarrollarán las trayectorias no solo facilita una gestión más eficiente del tráfico aéreo, sino que también permite una asignación anticipada y óptima de los recursos, lo que resulta fundamental para evitar congestiones y minimizar tiempos de espera, contribuyendo así a una operación aérea más fluida y segura. Junto con esto, al conocer con antelación las trayectorias planificadas de cada avión, se mejora significativamente la capacidad de los controladores aéreos para prevenir conflictos de tráfico y coordinar maniobras seguras, mitigando riesgos potenciales y promoviendo una gestión del tráfico aéreo más ordenada y eficiente.

Con el avance de la tecnología y el acceso a grandes volúmenes de datos, las técnicas de Aprendizaje Profundo (*Deep Learning*) se alzan como una herramienta poderosa para abordar este desafío, siguiendo un enfoque dirigido por datos (*data-driven*). Al aprovechar la capacidad de los modelos de Deep Learning para extraer patrones complejos en los datos, es posible desarrollar sistemas de predicción de trayectorias de aeronaves capaces de estimar con precisión el estado futuro del espacio aéreo.

Este Trabajo Fin de Máster (TFM) surge de la necesidad de mejorar la precisión y eficiencia de las predicciones de las trayectorias de vuelo, con la finalidad de obtener un sistema capaz de contribuir y simplificar la gestión del tráfico aéreo. Es por esto que en este proyecto se analizarán e implementarán arquitecturas de *Deep Lerning* novedosas, con el objetivo de avanzar en la capacidad de predecir trayectorias futuras de aeronaves con mayor exactitud, tanto en las regiones cercanas a los aeropuertos como en el resto del viaje, e independiente de la ruta que le sea asignada.

1.1. Planteamiento del problema (*Problem Statement*)

El sistema de predicción de trayectorias ideal sería aquel capaz de predecir los vectores restantes de las trayectorias con total precisión. Esto incluiría cualquier posible cambio no planeado que surja por condiciones climáticas adversas, problemas técnicos o restricciones impuestas por la gestión del tráfico aéreo. De esta manera, los controladores de tráfico aéreo podrían prever el estado del espacio aéreo futuro, e informar a los pilotos que estén volando en ese momento o vayan a comenzar un vuelo, lo que permitiría optimizar el tráfico aéreo. Así, se podría planificar con antelación los recursos y maniobras necesarias para garantizar que las operaciones se realicen de forma segura y eficiente.

Sin embargo, los sistemas de predicción de trayectorias aéreas actuales no ofrecen la precisión necesaria para poder estimar correctamente el estado futuro del espacio aéreo, especialmente en las regiones próximas a los aeropuertos. En estas zonas existe una mayor concentración de aeronaves que se encuentran ascendiendo o descendiendo, lo que provoca que el control de este tráfico aéreo resulte una tarea compleja. Además, estas condiciones aumentan la probabilidad de que ocurran errores humanos, acrecentados por la presión y el cansancio generados por las largas jornadas de trabajo.

Una mala gestión en el control del tráfico aéreo provoca que sean necesarias maniobras adicionales, lo cual conlleva retrasos en las trayectorias y un aumento del combustible consumido, generando pérdidas económicas significativas a las aerolíneas y aumentando la huella de carbono. Además de esto, la falta de coordinación entre aeronaves por fallos humanos o falta de precisión en la posición de las aeronaves puede desembocar en accidentes, poniendo en riesgo la seguridad de los pasajeros y la tripulación.

En este proyecto se busca mejorar la precisión en las predicciones de las trayectorias, de tal forma que, tanto pilotos como controladores, puedan recibir una estimación del estado futuro del espacio aéreo, lo que les permitirá tomar decisiones en consecuencia. Para ello, se analizarán y evaluarán arquitecturas de *Deep Learning* de vanguardia: **TSMixer** [4] y **Liquid Neural Networks** [12]. Estas arquitecturas han demostrado una mayor precisión a la hora de realizar predicciones sobre series temporales (como son las trayectorias aéreas), frente a otras alternativas en el estado del arte [4] [12]. Otra de las características de las propuestas de este proyecto es que no se diseñarán para predecir trayectorias de rutas concretas, si no que tendrán la capacidad de predecir la trayectoria de vuelo independientemente de la ruta en la que se encuentre la aeronave.

Los datos que se utilizarán para entrenar estos modelos se basan en los mensajes de los sistemas ADS-B (*Automatic Dependent Surveillance – Broadcast*) [6], que transmiten de forma periódica (cada varios segundos) datos sobre el estado actual de la aeronave. Esta información se ha organizado en forma de "trayectorias 4D", en las que se tiene en cuenta el factor temporal para definir la posición de la aeronave (junto con la longitud, latitud y altitud).

1.2. Objetivos del trabajo

Este proyecto busca analizar, crear y comparar modelos de *Deep Learning* para la predicción de trayectorias de aeronaves. Las arquitecturas que se utilizarán son TSMixer y Liquid Neural Networks (LNN), y se desarrollará una interfaz común para la utilización de los modelos de estas arquitecturas (entrenamiento, predicción, etc). Tras esto, se evaluará para cada arquitectura el impacto de los hiperparámetros sobre las mismas, a través de entrenamientos sobre un subconjunto representativo de los datos (datos de vuelos producidos a lo largo de un mes), y analizando

y comparando los resultados obtenidos. Finalmente, se llevará a cabo el entrenamiento con el conjunto de datos completo (correspondiente a nueve meses de datos de vuelo), obteniendo los modelos finales que, una vez más, se analizarán y compararán. Sobre esta visión del proyecto, se describen los siguientes objetivos:

OBJ-01 Implementar las arquitecturas para la predicción de trayectorias (TSMixer y LNN).

- **OBJ-01.1** Definir una interfaz común para la utilización de las modelos de las arquitecturas propuestas.

OBJ-02 Estudiar el impacto de los hiperparámetros sobre los modelos.

OBJ-03 Analizar y comparar los modelos finales.

- **OBJ-03.1** Optimizar los hiperparámetros de los modelos sobre el conjunto de datos completo.

OBJ-04 Desarrollar una herramienta de visualización para las predicciones.

- **OBJ-04.1** Interpretar y analizar de los resultados de las predicciones de los modelos.

1.3. Estructura de la memoria

En los tres primeros capítulos de este TFM se describe el proyecto a alto nivel. Se comienza con una explicación de los conceptos clave del proyecto (capítulo 1), para a continuación describir la planificación que se ha realizado del mismo (capítulo 2): qué metodología se utiliza, las estimaciones temporales y presupuestarias, y cómo se han ajustado estas estimaciones con la realidad tras completar el proyecto.

La siguiente sección trata sobre los antecedentes del proyecto (capítulo 3), donde se describe el contexto de negocio y el contexto científico-técnico, tras lo cual se estudiarán y compararán otros sistemas de predicción de trayectorias aéreas del estado del arte con las de este proyecto.

En el capítulo 4 se describe el conjunto de datos utilizado, junto con la descripción de la implementación de las propuestas, así como las métricas utilizadas en los entrenamientos y evaluaciones. Seguidamente, se describirá la especificación e implementación de la herramienta de visualización de predicciones.

El capítulo 5 describe el proceso y las conclusiones del estudio del impacto de los hiperparámetros sobre cada modelo. Junto a esto, se detalla la fase de optimización de hiperparámetros de cada modelo, y se concluye con el análisis y comparación de los modelos finales con los del estado del arte.

El último capítulo (capítulo 6) contiene las conclusiones que surgen a partir del proyecto, evaluando los resultados obtenidos y el desarrollo del proyecto. Junto a esto, se proponen líneas de trabajo futuro, así como la perspectiva personal sobre el proyecto.

Capítulo 2

Planificación

En este apartado se describe la organización del proyecto: la metodología que se utiliza, las estimaciones temporales y presupuestarias que definen la línea de base, y el balance temporal y económico a lo largo del proyecto.

2.1. Metodología de trabajo

La metodología utilizada para la organización y planificación de este proyecto es ASAP (*Agile Student Academic Projects*) [18], una adaptación de Scrum [ref] para la realización de trabajos académicos. Así, ASAP reinterpreta los roles, eventos y artefactos de Scrum y añade algunos elementos nuevos para abordar con éxito las necesidades particulares del entorno de aprendizaje en el que se llevan a cabo este tipo de proyectos.

ASAP define un conjunto de roles para las personas que intervienen en el proyecto:

- **Estudiante:** es el encargado principal de construir un producto de calidad. Debe participar en todos los eventos de la metodología, definir y planificar tareas para alcanzar los objetivos de cada sprint, completar estas tareas según los criterios establecidos y mantener actualizado el estado del proyecto en el espacio de trabajo compartido.

- **Tutor:** Los tutores, que son uno o más profesores, proponen un proyecto acorde a los requerimientos de la guía docente. Ellos definen los objetivos a abordar al inicio de cada sprint, aseguran la realización satisfactoria de los eventos y proporcionan retroalimentación (*feedback*) de forma regular, resolviendo dudas o bloqueos que el estudiante pueda tener.

- **Comunidad:** Compuesta por personas que pueden aportar valor al producto. La comunidad asiste a eventos de comunicación de progresos y retrospectivas, ofreciendo feedback adicional para mejorar tanto el producto como el proceso.

- **Tribunal:** Formado por profesores designados para evaluar el TFM. El tribunal evalúa objetivamente el producto entregado y el acto de defensa, basándose en los criterios establecidos en el reglamento de la asignatura.

Los eventos que define ASAP permiten llevar un seguimiento continuo del TFM y asegurar una interacción efectiva entre el estudiante y los tutores, estableciendo un ritmo de trabajo sostenido durante todo el TFM. ASAP estructura los TFM en *sprints*, períodos de tiempo con

una duración máxima de un mes en los que se planifican y abordan un conjunto de objetivos concretos. En cada *sprint* se realizan cuatro tipos de eventos:

- La **reunión de inicio**, en la que se establecen los objetivos del *sprint* y las tareas necesarias para alcanzarlo.
- Las **reuniones de sincronización**, realizadas semanalmente para llevar un seguimiento del TFM y comunicar posibles bloqueos.
- La **comunicación de progresos** al final del *sprint*, donde el estudiante presenta su TFM frente a los profesores y la comunidad de forma comparable a la defensa del TFM.
- La **retrospectiva**, en la que el alumno, los profesores y la comunidad reflexionan sobre la calidad del proceso seguido, indicando de forma anónima los aspectos positivos y negativos, así como formas de mejorar.

Finalmente, los artefactos que propone ASAP son el incremento, que reúne los resultados consolidados hasta el final del *sprint*; y la retroalimentación o *feedback* que generan los tutores sobre el incremento.

ASAP requiere un entorno tecnológico compuesto por tres elementos: un espacio de trabajo compartido entre el estudiante y el tutor, para facilitar la comunicación y la transmisión de recursos; un tablero de proyecto Kanban, para organizar las tareas; y un cuaderno de trabajo, donde el estudiante lleva un registro de las tareas realizadas y el tiempo invertido en las mismas, con el objetivo de mejorar la gestión del tiempo.

2.1.1. Herramientas para la gestíón

La plataforma Teams [19] facilita un espacio de trabajo virtual a través de la cual se lleva a cabo la comunicación oral y escrita entre el estudiante y los tutores, así como con los miembros de la comunidad. También se ha utilizado esta plataforma para almacenar y organizar los artefactos generados en el proyecto.

Se utiliza Trello [9] para administrar las tareas del proyecto. Esta herramienta ofrece un tablero Kanban donde se organizan tarjetas que representan a las tareas, asignándolas un nombre, una descripción, listas de subtareas, etc.

La herramienta en línea Online Gantt [8] permite elaborar diagramas de Gantt de forma gratuita, y exportarlos en forma de Excel, PDF o PNG. Estos diagramas se utilizan en este proyecto para describir la planificación y el balance temporal.

2.2. Planificación temporal

Como se especifica en la sección anterior, la metodología utilizada en este proyecto es ASAP, por lo que el proyecto se divide en *sprints*. En cada *sprint* se abordan un conjunto de objetivos que se deciden en la reunión de inicio (al comienzo del *sprint*). Para realizar la planificación temporal del proyecto, se establecen los objetivos de cada *sprint* (como un subconjunto de los objetivos del proyecto), y en la reunión de inicio se define qué se abordará (incluyendo la retroalimentación del *sprint* anterior) y cómo se llevará a cabo (es decir, las tareas que se realizarán). Cada *sprint* tiene una duración de un mes, por lo que se llevarán a cabo dos *sprints*: uno equivalente al proyecto de la asignatura I+D+I, y otro para el TFM.

Entonces, el primer *sprint* comienza el 6 de mayo de 2024 y termina el 7 de junio de 2024 y, siguiendo la guía docente de la asignatura I+D+I, se llevarán a cabo 190 horas de trabajo. En este primer *sprint* se estudian e implementan las arquitecturas propuestas (OBJ-01), para a continuación analizar el impacto en los resultados que provocan los hiperparámetros que definen cada modelo, utilizando un subconjunto representativo de los datos (el mes de enero). De esta forma, se podrán establecer conjuntos de hiperparámetros cuyos modelos muestren una menor tasa de error, y así encontrar el modelo óptimo sobre el subconjunto de datos (OBJ-02).

El segundo *sprint*, correspondiente al TFM, comienza el 10 de junio de 2024 y termina el 19 de julio de 2024, correspondiente a 150 horas de trabajo. En este *sprint* se lleva a cabo un análisis más preciso del impacto de los hiperparámetros sobre las propuestas (OBJ-02), y se entrenan algunos modelos con el conjunto de datos completo para compararlos con las soluciones actuales (LSTM y TFT [26]) (OBJ-03). De forma paralela, se desarrollará la herramienta de visualización de trayectorias, de tal forma que se utilice para comprender los resultados de los modelos entrenados (OBJ-04). En la figura 2.1 se muestra un diagrama de Gantt que describe la planificación temporal descrita.

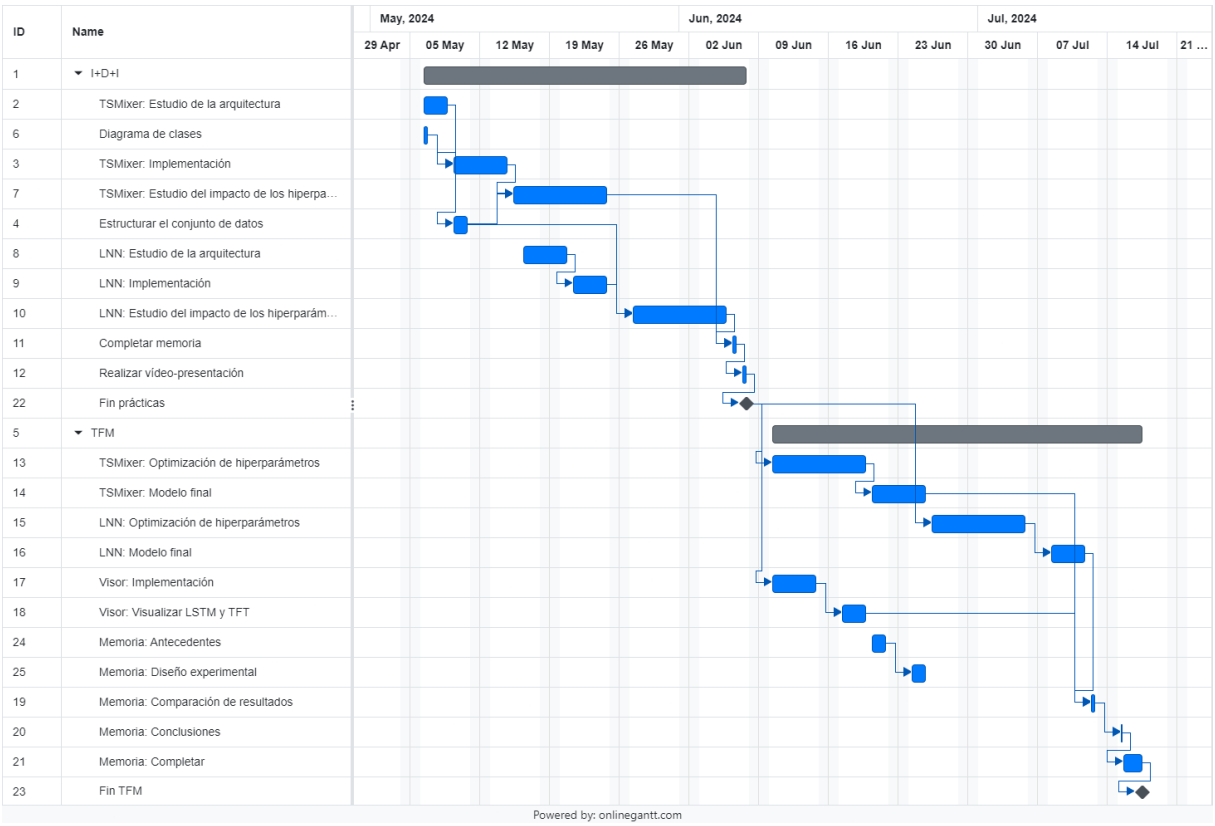

Figura 2.1: Diagrama de Gantt sobre la planificación temporal.

2.3. Presupuesto

El presupuesto es una estimación de los gastos que supondrá el proyecto. Para calcular estos gastos se tendrán en cuenta los recursos utilizados y la cantidad de tiempo o unidades que se

estiman utilizar, así como el esfuerzo humano que requiere el proyecto.

Personal Se tienen en cuenta los roles de la tabla 2.1. Para definir el coste de cada rol se parte del sueldo bruto anual, para el cual se considera el nivel de experiencia (junior) y la media en España en 2023 [2], y a partir de él se calcula el coste para la empresa (costes para cubrir contingencias comunes, accidentes de trabajo, enfermedades profesionales, etc), utilizando una calculadora en línea [3]. Según [20], desde el 1 de enero de 2024 la jornada laboral efectiva es de 1620 horas anuales, por lo que se utilizará este valor para calcular el coste por hora.

Hardware Se tiene en cuenta el uso del hardware en el proyecto respecto al tiempo de vida útil del dispositivo. Es necesario un equipo de alto rendimiento para poder entrenar al modelo (1250 €). Se estiman unas 50 horas de uso semanales durante 4 años de vida útil, obteniendo un total de 10400 horas de vida útil y 12.02 céntimos por hora de uso. Se tendrán en cuenta para el presupuesto 340 horas de uso por parte del alumno (con entrenamiento en segundo plano) y 500 horas dedicadas únicamente a entrenamiento.

Software Todas las herramientas y librerías son gratuitas.

Conectividad Se tiene en cuenta el uso de la red destinado al proyecto.

Rol	Tareas	Coste anual	Coste por hora
Gestor	Planificación, coordinación y supervisión	48673 €	30,05 €
Científico de datos	Investigación, modelado y ajuste de las propuestas.	34671 €	21,40 €

Tabla 2.1: Roles en el proyecto.

Descripción	Coste / Unidad	Unidades	Coste total
Gestor	30,05 €/h	34h (10 %)	1021,70 €
Científico de datos	21,40 €/h	306h (90 %)	6548,40 €
Ordenador	12,02 cént/h	840h	100,97 €
Conexión a Internet	28,95 €/mes (4,02 cént/h)	840h	33,77 €
Total			7704,84 €

Tabla 2.2: Presupuesto para el proyecto

2.4. Balance temporal

En esta sección se evalúa la concordancia de la planificación temporal descrita en el apartado 2.2 con los resultados reales. Para facilitar este análisis, se ha desarrollado un diagrama de Gantt (figura 2.2) que recoge las tareas realizadas a lo largo del proyecto, y su duración real.

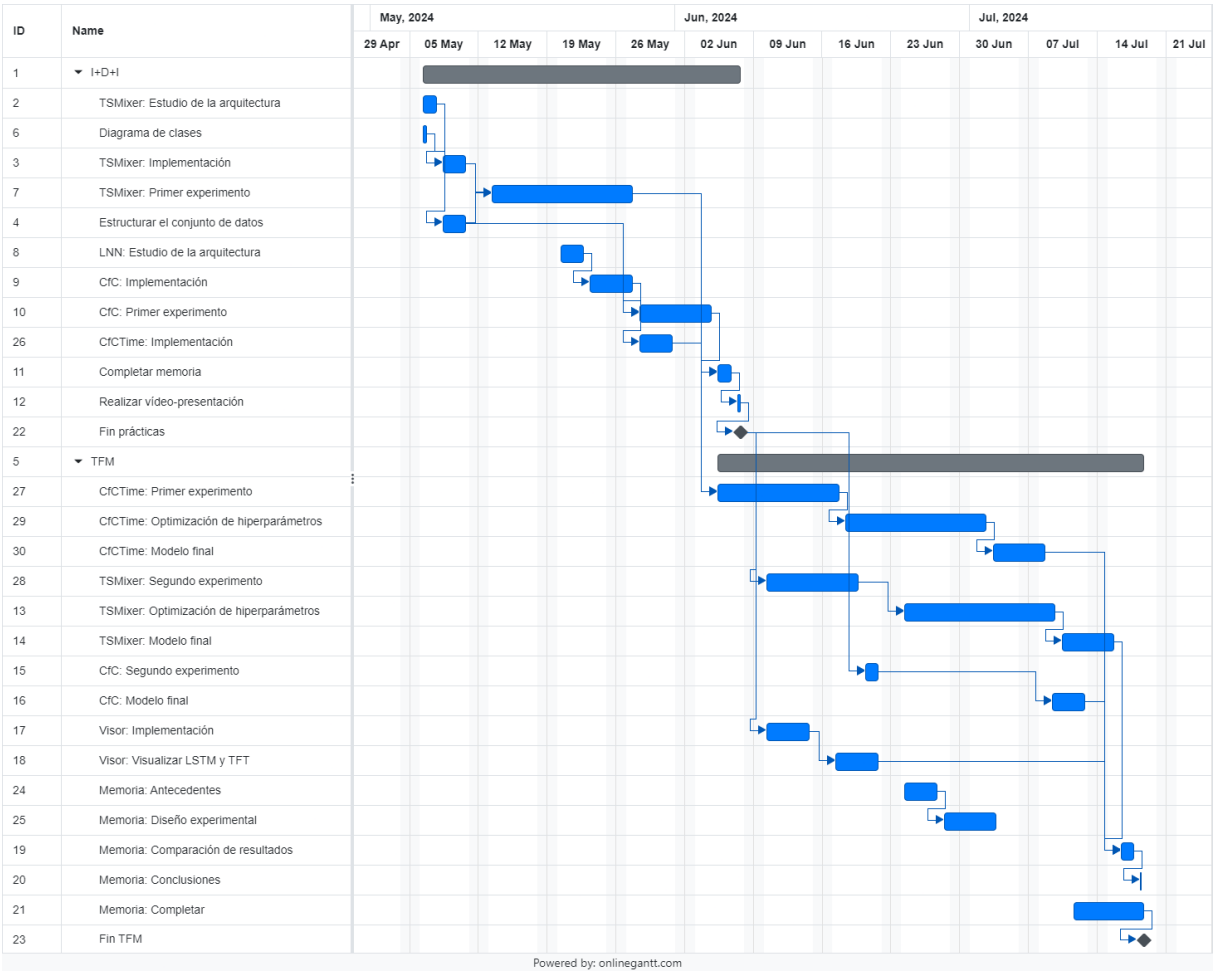

Figura 2.2: Diagrama de Gantt sobre el balance temporal.

Durante el primer *sprint*, correspondiente a las prácticas de I+D+I, se ha mantenido, en general, la distribución de tareas. Como se ha desarrollado una propuesta adicional que no se contemplaba en la planificación (CfCTime), se ha incluido su implementación en este *sprint*. Por otro lado, el estudio del impacto de los hiperparámetros ha requerido de un mayor esfuerzo, por lo que se ha repartido entre los dos *sprints*. El número de horas de esfuerzo en este *sprint* es de 190 horas, ajustándose a la planificación temporal establecida.

El segundo *sprint* ha sufrido más cambios sobre la planificación inicial, aunque los objetivos principales se han mantenido (OBJ-02, OBJ-03, OBJ-04). La organización de este *sprint* se basa en el entrenamiento de los modelos de forma paralela al trabajo del alumno. Debido a la larga duración de los entrenamientos, y disponer únicamente de dos entornos de ejecución (el ordenador personal y Google Colab), ha sido necesario coordinar de forma precisa los entrenamientos con la implementación del visor y la redacción de la memoria, para aprovechar al máximo la disponibilidad de los entornos para el entrenamiento. La cantidad de horas de esfuerzo en este *sprint* es de 150 horas, por lo que concuerda con la planificación temporal establecida.

2.5. Balance económico

De forma similar al apartado anterior, en esta sección se realizará una comparación de los costes reales del proyecto frente a la planificación presupuestaria. En la tabla 2.3 se describen los costes reales del proyecto junto con el total, donde se puede comprobar que el coste total del proyecto (7646,86 €) es muy similar al presupuestado (7704,84 €). Aunque el número de horas que se ha utilizado el ordenador es superior al estimado (70 horas más), el coste adicional se compensa al reducirse el número de horas de esfuerzo del gestor respecto al presupuesto, cuyo coste por hora es superior al del científico de datos.

Descripción	**Coste / Unidad**	**Unidades**	**Coste total**
Gestor	30,05 €/h	26h	781,30 €
Científico de datos	21,40 €/h	314h	6719,60 €
Ordenador	12,02 cént/h	910h	109,38 €
Conexión a Internet	28,95 €/mes (4,02 cént/h)	910h	36,58 €
Total			7646,86 €

Tabla 2.3: Tabla de costes del proyecto

Capítulo 3

Antecedentes

En esta sección se analiza el entorno en el que se desarrolla el proyecto, se describen las arquitecturas utilizadas en las propuestas, y se estudian otros modelos desarrollados para la predicción de trayectorias aéreas. Finalmente, se indican las herramientas y librerías que se utilizan.

3.1. Contexto de negocio

La gestión del tráfico aéreo (ATM) es el conjunto de funciones, tanto a bordo como en tierra, que aseguran el movimiento seguro y eficiente de las aeronaves. El control del tráfico aéreo (ATC) es una parte clave de esta gestión, emitiendo autorizaciones y restricciones en función del tránsito y las condiciones del entorno. Para ello, el espacio aéreo se divide en zonas de gestión ATC, cada una gestionada por un grupo de controladores, con el objetivo de acotar la franja de actuación.

Para que el control del tráfico aéreo sea viable resulta necesario recopilar y procesar los datos de posición de las aeronaves que circulan en cada zona ATC, esencialmente la longitud, latitud y altitud. Junto con esto, resulta necesario incluir una dimensión temporal para poder medir posibles variaciones sobre la planificación del vuelo, dando lugar al concepto de "trayectoria 4D" [25]. Una trayectoria 4D consiste en la sucesión de los puntos que recorre una aeronave desde el despegue hasta el aterrizaje, y que están definidos por su longitud, latitud, altitud y tiempo. El uso de trayectorias 4D facilita la asignación del espacio aéreo y los recursos de los aeropuertos, y permite la realización de tareas avanzadas, como la gestión de la capacidad del espacio aéreo, la mejora de la planificación táctica y una mejor evaluación de la seguridad [26].

3.1.1. ADS-B

Una de las técnicas utilizadas tradicionalmente para la transmisión de información de posición (y otros datos adicionales) son los radares primarios y secundarios, que funcionan mediante la emisión y recepción de ondas electromagnéticas [25]. Sin embargo, estos sistemas presentan limitaciones en zonas extensas sin antenas receptoras, como océanos, mares y áreas montañosas. Otro problema que presentan estos sistemas es el tiempo de retardo, que puede alcanzar los doce segundos [25], lo cual es bastante significativo considerando la velocidad a la que se mueven las aeronaves.

Para hacer frente a estos inconvenientes, se desarrolló la tecnología ADS-B (*Automatic Dependent Surveillance Broadcasting*) [6]. Este sistema, que sustituye al radar secundario, permite a

las aeronaves transmitir automáticamente y de manera periódica información sobre su trayectoria. Esto permite que otras aeronaves o centros de control, equipados con receptores compatibles con ADS-B, puedan recibir esta información [25].

Una de las ventajas de este sistema es la mejora en la precisión de la información respecto al radar secundario, con un tiempo de retardo de sólo dos segundos, lo que facilita el seguimiento y la gestión de un mayor volumen de tráfico aéreo. Otro aspecto a tener en cuenta es que el coste de estos sistemas es inferior al de los radares secundarios, y aunque ADS-B sigue mostrando problemas en zonas oceánicas (puesto que actualmente no se puede desplegar en estas áreas), mejora la cobertura en las áreas montañosas.

La recopilación y procesamiento de los mensajes de ADS-B complementa la información de los controladores aéreos, mejorando así la gestión del tráfico aéreo. Existen redes de receptores destinadas a la recopilación y almacenamiento de mensajes ADS-B, como OpenSky Network [29], que es la fuente utilizada en este proyecto.

3.1.2. Aeropuerto Adolfo Suárez Madrid-Barajas

Los datos utilizados en este proyecto se corresponden con vuelos que tienen como destino el aeropuerto Adolfo Suárez Madrid-Barajas, que es el aeropuerto más transitado de España (figura 3.1). Su base aérea se extiende a lo largo de aproximadamente 2000 hectáreas de superficie, y cuenta con cuatro terminales, una Terminal Ejecutiva, un centro de carga aérea y dos zonas de hangares. Dispone de cuatro pistas paralelas dos a dos, cuyo uso (despegues o aterrizajes) depende de factores climatológicos y de seguridad, generalmente optando por una configuración norte para minimizar la contaminación acústica. Cabe destacar que, en el período nocturno (de 23:00 a 7:00), se utilizan únicamente dos pistas.

3.2. Contexto científico-técnico

3.2.1. Series temporales

Una serie temporal es una sucesión de vectores o puntos de datos registrados en intervalos de tiempo regulares y ordenados cronológicamente. Los valores en un momento concreto dependen de su evolución en el tiempo. El objetivo de la predicción de series temporales es obtener una serie de puntos de datos futuros (horizonte) a partir de un conjunto de vectores pasados (ventana). Como se puede ver en la figura 3.2. los vectores que describen el horizonte y la ventana dependen del punto actual, donde la ventana incluye a este punto, y el horizonte comienza desde el siguiente vector.

La predicción de trayectorias 4D se puede enfocar desde el punto de vista de una serie temporal, de tal forma que la posición de una aeronave en una trayectoria depende de su evolución a lo largo de la misma. Siguiendo este enfoque, el objetivo será predecir para cualquier punto de la trayectoria los siguientes vectores futuros que forman el horizonte, descritos por la longitud, latitud y altitud (al estar separados en intervalos regulares, el tiempo está implícito); y para realizar esta predicción, se utilizará la evolución de la posición de la aeronave hasta el punto actual (es decir, los vectores que forman la ventana).

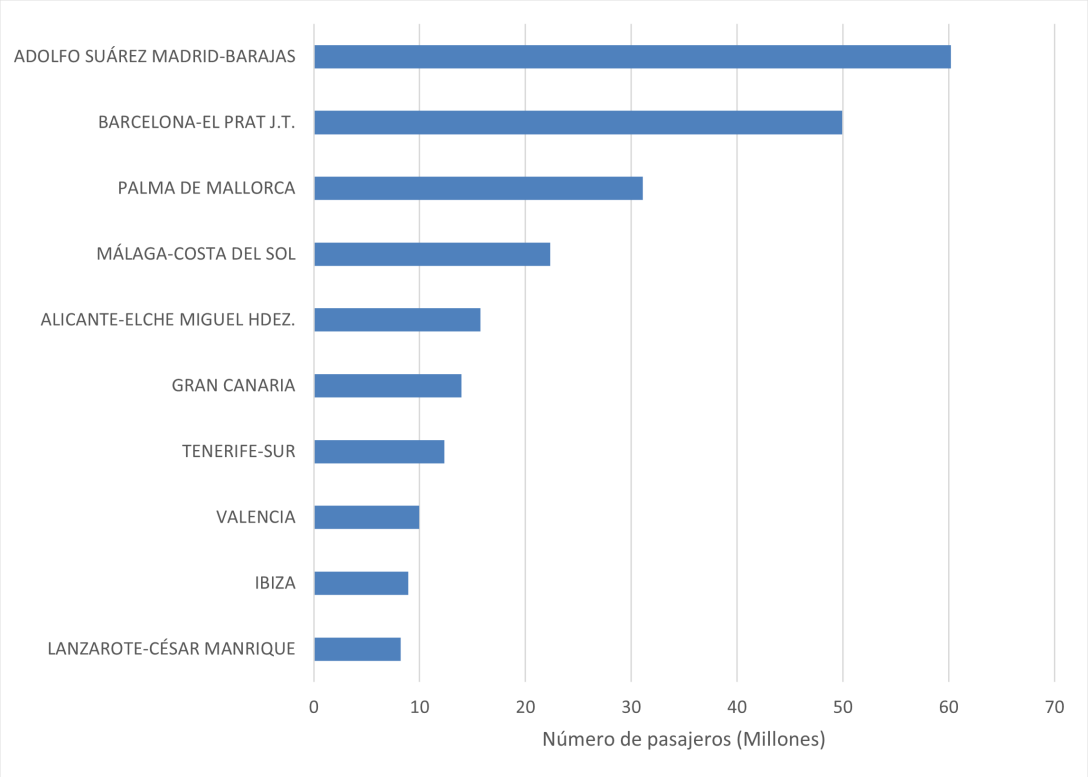

Figura 3.1: Volumen de pasajeros (millones) en los principales aeropuertos de España en el año 2023. Datos obtenidos de [14].

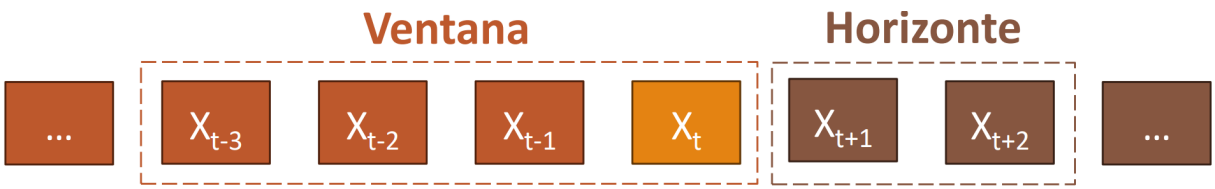

Figura 3.2: Ejemplo de serie temporal. El instante actual es t, el tamaño de la ventana es 4 y el del horizonte, 2.

3.2.2. Aprendizaje Profundo (Deep Learning)

Una red neuronal artificial es un algoritmo de Aprendizaje Automático (Machine Learning) capaz de reconocer y extraer patrones de un conjunto de datos, con el objetivo de generalizarlos y realizar predicciones a partir de nueva información. Estas redes buscan imitar el comportamiento y estructura de las neuronas humanas; para ello, las redes se definen a partir de neuronas artificiales interconectadas y agrupadas en capas, de tal forma que sólo se pueden conectar neuronas de una capa con las de la siguiente capa (esto se conoce como una red *feedforward*). Cada neurona recibe la información que generan las neuronas de la capa anterior, la procesa, y transmite su resultado a las neuronas de la siguiente capa. Las redes neuronales artificiales reciben como entrada una muestra de los datos, que procesan en las capas intermedias, y devuelven como salida el resultado (figura 3.3).

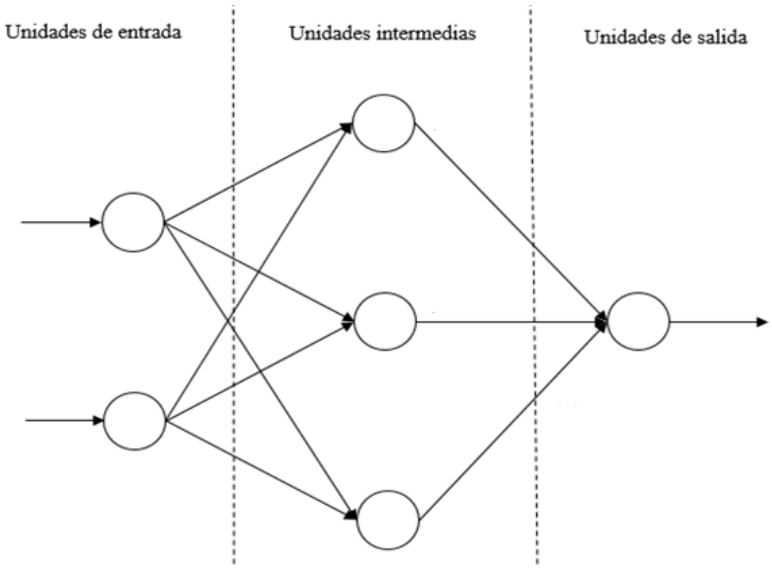

Figura 3.3: Representación de las capas de una red neuronal artificial. Adaptado de [7]

El Aprendizaje Profundo o *Deep Learning* es una rama del Aprendizaje Automático que se caracteriza por utilizar redes neuronales artificiales con una gran cantidad de neuronas y capas, con el objetivo de modelar y descubrir patrones complejos en los datos. Estas redes, inspiradas en la estructura del cerebro humano, son capaces de aprender representaciones jerárquicas, donde las primeras capas capturan características simples de los datos, mientras que las últimas aprenden patrones más complejos.

Los algoritmos de Aprendizaje Profundo son capaces de encontrar patrones complejos en los datos, pero para ello requieren de un gran volumen de información de calidad, puesto que el ruido en los datos y otros factores afectan negativamente a su desempeño; además, es necesaria una alta capacidad computacional para entrenar los modelos.

En la actualidad, el Aprendizaje Profundo destaca en áreas como la visión computacional, el procesamiento del lenguaje natural o el reconocimiento del habla, entre otras. Las arquitecturas que más relevancia han conseguido son las redes Convolucionales (*Convolutional Neural Networks*, CNN), las redes Recurrentes (*Recurrent Neural Networks*, RNN) y *Transformer*. Estas arquitecturas se pueden adaptar a problemas de predicción de series temporales, aunque las redes

Convolucionales resultan más adecuadas para procesar conjuntos de datos basados en imágenes y vídeos, por lo que no se contemplarán en este proyecto.

3.2.2.1. Redes Recurrentes (RNN)

Las redes Recurrentes (RNN) surgen de la necesidad de procesar series temporales de forma eficiente. Este tipo de redes presentan el concepto de "memoria", al mantener información de las entradas anteriormente procesadas; esto se debe a la capacidad de estas neuronas de mantener conexiones consigo mismas (denominadas "ciclos") o con otras neuronas de su misma capa, de tal forma que su salida actual se ve influenciada por las salidas anteriores. Una forma de representar las RNN es mediante el *unrolling*, que consiste en conectar un conjunto de redes neuronales no recurrentes, una por cada entrada de la secuencia (figura 3.4).

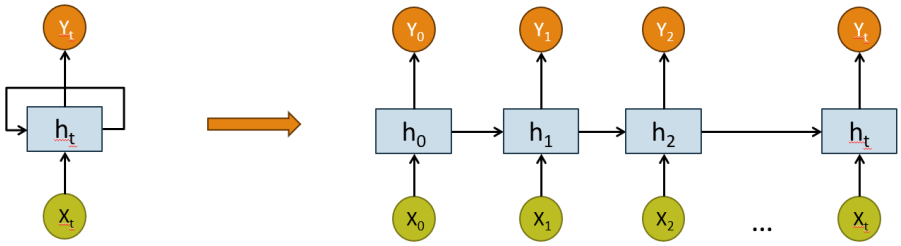

Figura 3.4: Representación del *unrolling* de una RNN, donde "X" representa a las entradas, "h" a la neuronas e "Y" a las salidas.

3.2.2.2. Redes LSTM

Las redes LSTM (*Long-Short Term Memory*) [13] son un tipo de red neuronal recurrente (RNN) diseñada para aprender dependencias a largo plazo, puesto que hacen frente al problema del desvanecimiento del gradiente, el cual es común en las RNN estándar. Estas redes han demostrado ser muy efectivas a la hora de modelar series temporales, como son las trayectorias 4D [26].

Las redes LSTM se componen de celdas LSTM, que tienen la capacidad de aprender patrones temporales a corto y largo plazo. Para conseguir esto, las celdas contienen una celda de estado, que se encarga de mantener la información a lo largo del tiempo; junto a esta, las celdas LSTM se componen de tres puertas, que utilizan los datos de entrada junto con la salida y el estado de las celdas anteriores para calcular la salida y el nuevo estado (figura 3.5):

- **Puerta de olvido**: Se encarga de definir el grado en que el estado de la celda anterior afectará al nuevo estado.
- **Puerta de entrada**: Se encarga de definir qué valores de la entrada y la salida de la celda anterior se utilizarán para calcular el nuevo estado.
- **Puerta de salida**: Calcula la salida de la celda a partir de los datos de entrada, la salida de la celda anterior y el resultado de la puerta de entrada.

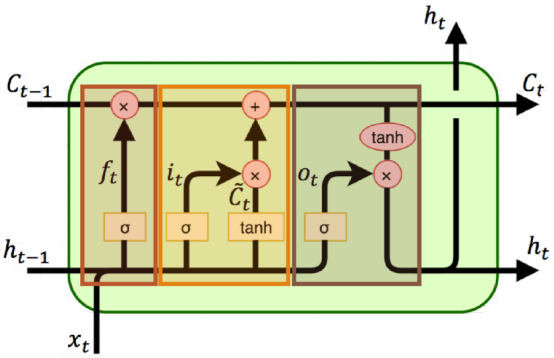

Figura 3.5: Representación de una celda LSTM, El rectángulo rojo representa a la puerta de olvido, el naranja a la de entrada, y el marrón a la de salida.

3.2.2.3. *Transformer*

Transformer [30] es una arquitectura basada en el uso de mecanismos de atención, que le permite procesar secuencias de datos en paralelo mejorando significativamente la eficiencia computacional. Los mecanismos de atención permiten encontrar relaciones y enfocarse en las partes relevantes de las secuencias de entrada, y de esta forma capturar la importancia de cada fragmento en el contexto de la secuencia completa. Concretamente, *Transformer* utiliza *Multi-Head Attention*, que combina varios mecanismos de atención que actúan en paralelo. La arquitectura *Transformer* se estructura en dos componentes: el codificador y el decodificador (figura 3.6):

- El **codificador** (*encoder*) se encarga de la extracción de las características de los datos de entrada, descartando la información redundante y obteniendo, como consecuencia, una representación más sencilla de las entradas.
- El **decodificador** (*decoder*) realiza la operación inversa del codificador: utiliza la representación sencilla de las entradas para generar la salida del modelo.

3.2.2.4. *Temporal Fusion Transformer* (TFT)

Temporal Fusion Transformer o TFT [17] es una arquitectura propuesta por Google que se basa en *Transformer*, y que utiliza mecanismos de atención para extraer relaciones existentes en la serie temporal de manera eficiente. Esta arquitectura, a diferencia de *Transformer*, incorpora celdas LSTM para procesar series temporales, y las combina con la estructura *encoder-decoder* (figura 3.7). Finalmente, los mecanismos de atención permiten enfocarse en diferentes partes de la serie temporal en distintos momentos, lo que facilita capturar relaciones complejas y dependencias temporales en los datos. TFT procesa los datos de forma diferente dependiendo de su categoría:

- ***Static:*** Atributos estáticos, que no varían a lo largo del tiempo.
- ***Time-varying known:*** Variables temporales conocidas, es decir, características que varían en el tiempo, pero que se conoce su valor futuro.
- ***Time-varying unknown:*** Variables temporales desconocidas, que varían en el tiempo pero se desconoce su valor futuro.

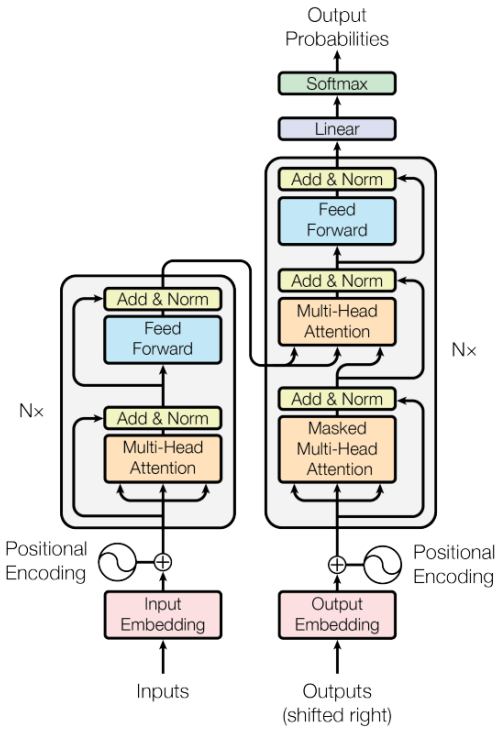

Figura 3.6: Representación de la arquitectura *Transformer*. A la izquierda, el codificador, y a la derecha, el decodificador. Fuente: [30]

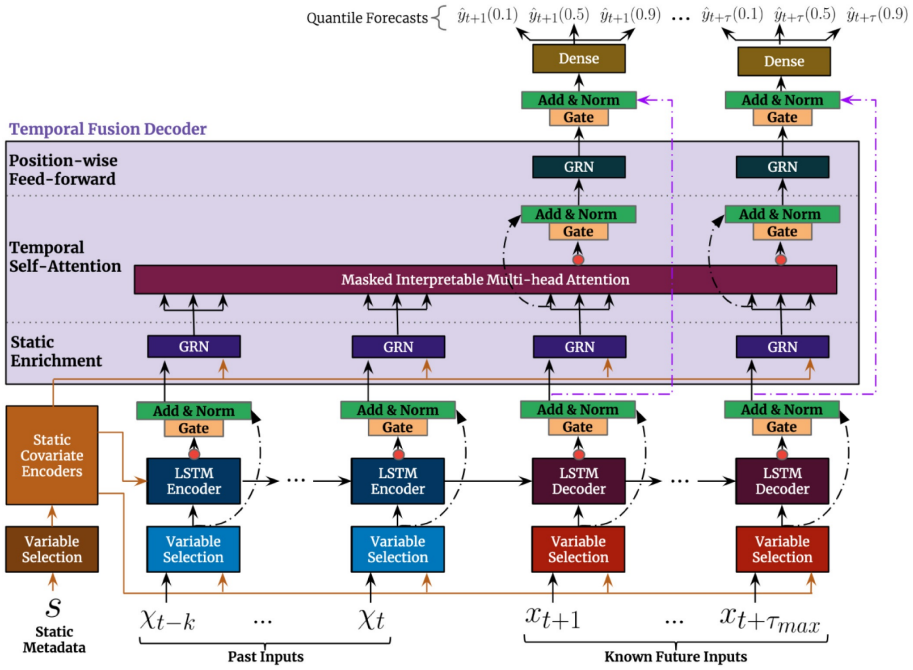

Figura 3.7: Representación de la arquitectura TFT. Fuente: [17]

TFT procesa cada una de estas categorías de forma diferente. Además, esta arquitectura utiliza una red para seleccionar las características más importantes, evitando centrar la atención en atributos menos relevantes. Otro aspecto a tener en cuenta es que TFT ofrece la posibilidad de realizar predicciones por cuantiles, de tal forma que se obtengan rangos de predicciones en lugar de un único valor. Esto permite que las salidas resulten más robustas y realistas, y ofrece una medida de la variabilidad de los resultados.

3.2.2.5. TSMixer

TSMixer [4] es una arquitectura para la predicción de series temporales que surge de la concatenación de Perceptrones Multicapa (*Multi-Layer Perceptron*, MLP), lo que la convierte en una arquitectura ligera y rápida de entrenar. TSMixer se compone de dos tipos de bloques: las capas (o bloques) de mezcla y la proyección temporal (figura 3.8).

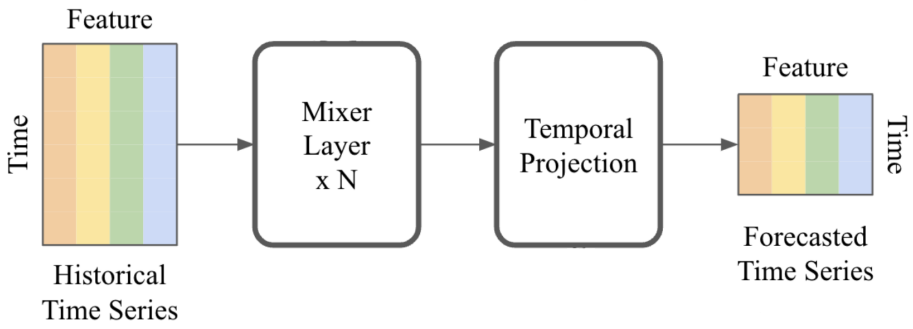

Figura 3.8: Estructura de TSMixer. Fuente: [4]

Cada capa de mezcla se divide en dos fases: la fase de mezcla temporal y la de mezcla de características (figura 3.9). Al inicio de ambas fases se realiza una normalización del lote de datos. En la primera fase se capturan los patrones temporales de las características, por lo que se transponen los datos para alimentar a una capa densa (capa de Perceptrones con todas las conexiones posibles) en la dimensión temporal. El resultado se vuelve a transponer y se suman los valores de las entradas (esta técnica se denomina "conexiones residuales", que facilita el entrenamiento al aprender variaciones en las entradas en vez de los valores completos).

La segunda fase (mezcla de características) recibe la salida de la fase anterior y se procesa en dos capas densas (la primera de tamaño variable), con el objetivo de capturar las correlaciones entre las características. Finalmente, se aplican las conexiones residuales de las entradas de esta fase. El resultado será la salida del bloque de mezcla, y se entregará al siguiente bloque o a la proyección temporal.

La proyección temporal se encarga de realizar la predicción del horizonte, a partir de la salida de la última capa de mezcla. Este bloque está compuesto por una única capa densa del tamaño de la predicción, por lo que es necesario transponer los datos para procesarlos en la dimensión temporal (figura 3.10). La salida de este bloque es el resultado de la red, es decir, la predicción final.

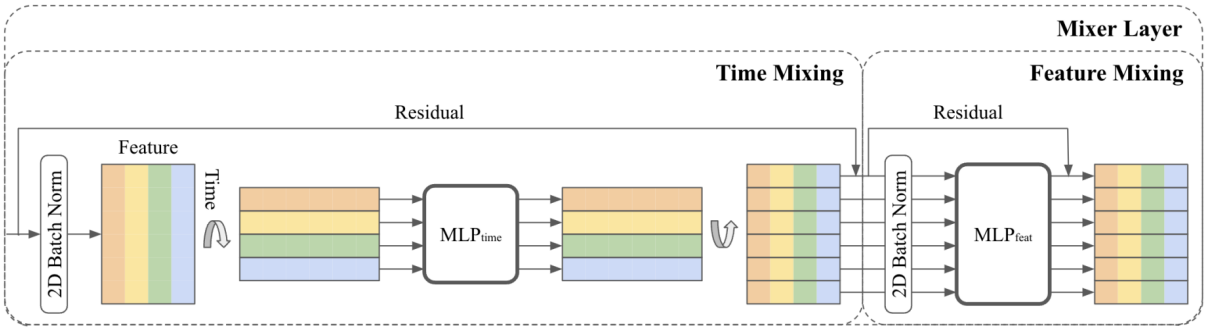

Figura 3.9: Representación de la capa de mezclas. Fuente: [4]

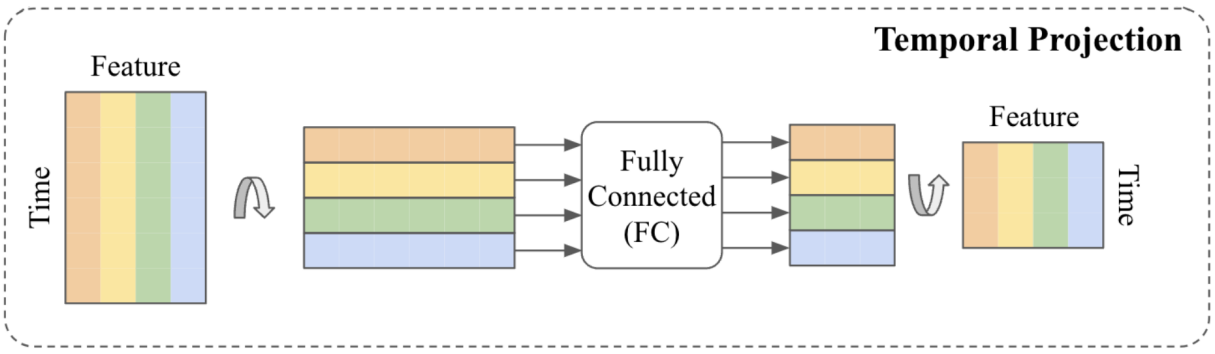

Figura 3.10: Representación de la proyección temporal. Fuente: [4]

3.2.2.6. Liquid Neural Networks (LNN)

En las redes neuronales tradicionales, las conexiones entre las neuronas se definen al crear el modelo y se mantienen fijas. Sin embargo, las Redes Neuronales Líquidas (LNN) tienen la capacidad de modificar estas conexiones de forma dinámica, lo que les permite adaptarse a entradas irregulares (como series temporales con intervalos variables), ser más resistentes al ruido y encontrar nuevas soluciones a problemas complejos. El objetivo de las LNN es reducir el número de neuronas necesarias al aumentar la capacidad de representación de la información de cada una, lo que a su vez facilita la transparencia e interpretación de la red.

Las Redes de Constante Temporal Líquida (*Liquid Time-Constant Networks*, LTC) [12] son una clase de LNN en las que se utiliza una constante temporal para definir los pesos de las conexiones entre las neuronas, la cual depende de los valores de las entradas. El estado interno de la red se define a través de ecuaciones diferenciales ordinarias, por lo que es necesario un solucionador o *solver* para resolverlas.

El problema es que este *solver* resulta considerablemente lento, suponiendo un cuello de botella en las LTC. Para afrontar este problema surgen las Redes Neuronales de Tiempo Continuo de Forma Cerrada (*Closed-form Continuous-time Neural Networks*, CfC) [11], donde se realiza una aproximación de las ecuaciones diferenciales anteriormente descritas, de tal forma que no es necesario utilizar el *solver*. El error cometido por estas aproximaciones es despreciable, mientras que el tiempo de entrenamiento de la red se reduce de dos a diez veces.

En las LNN, no existe el concepto de "capa", puesto que puede existir una conexión entre

cualquier par de neuronas (o consigo mismas, lo que se conoce como ciclo). Para definir el número de conexiones que tendrá la red, se pueden seguir tres estrategias (figura 3.11):

- **Densa (*fully connected*)**: Establecer todas las conexiones posibles entre las neuronas.
- **Aleatoria (*random*)**: Se define un subconjunto aleatorio de conexiones. El número total de conexiones se establece mediante un "nivel de dispersión" (*sparsity level*).
- **NCP**: *Neural Circuit Policies* es la estrategia propuesta en [12] para estructurar las conexiones de las LTC y CfC. NCP recupera el concepto de capa, definiendo cuatro grupos de neuronas de tal forma que las neuronas de un grupo sólo pueden tener conexiones salientes hacia las del siguiente grupo. Estos grupos de neuronas son:
  - **Sensoriales**: Representan a la entrada.
  - **Intermedias**: Capturan patrones en los datos.
  - **De mando**: Son las únicas neuronas que pueden tener ciclos. Procesan los resultados de las neuronas intermedias para definir la salida.
  - **Motor**: Representan a la salida.

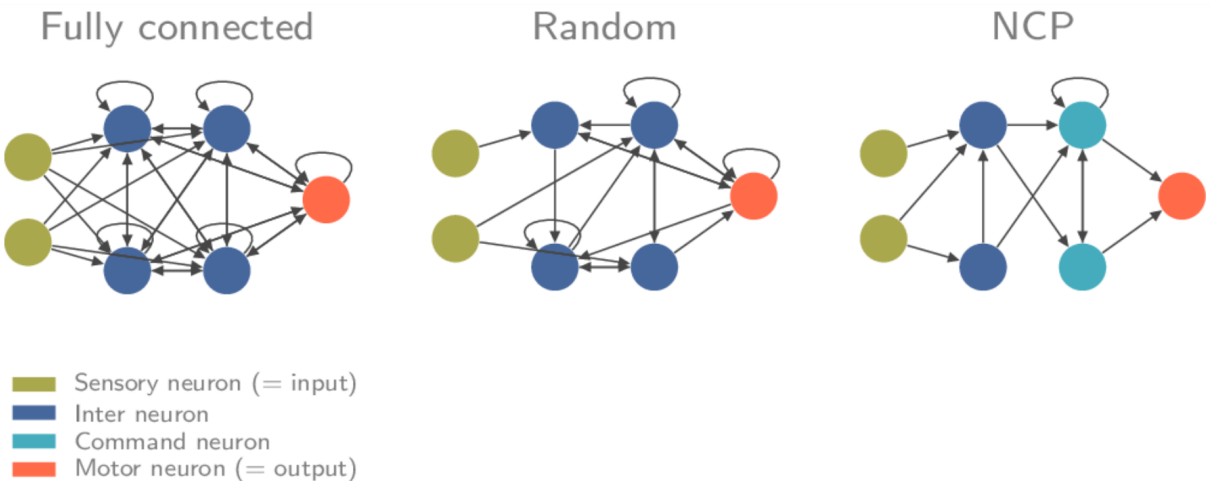

Figura 3.11: Estrategias para definir las conexiones en LNN. Fuente: [23]

La estrategia densa produce redes demasiado complejas, lo que provoca que los recursos necesarios para el entrenamiento sean elevados y que el resultado empeore, al aumentar el ruido interno en la red. Con la estrategia aleatoria se mitigan estos problemas, aunque se añade un factor de aleatoriedad en la creación de la red, que puede provocar que las conexiones se establezcan de forma desfavorable (por ejemplo, que algunas neuronas no reciban conexiones). Finalmente, la estrategia NCP consigue mejores resultados frente a las anteriores utilizando menos conexiones entre las neuronas [12], aunque mantiene el factor de aleatoriedad. En este proyecto se utilizará esta estrategia para definir las conexiones de las redes CfC.

3.3. Estado del arte

En esta sección se revisan otros proyectos sobre la predicción de trayectorias utilizando *deep learning*. El estado del arte se compone principalmente de arquitecturas basadas en redes LSTM. Sin embargo, en [25] y [26] se presentan tres modelos para la predicción de trayectorias 4D, que se alinean con los objetivos de este proyecto. Las propuestas que se describen son dos modelos basados en redes LSTM, y uno en TFT.

La primera propuesta basada en LSTM se compone de una capa LSTM seguida de una capa densa y una capa que modifica la dimensionalidad del vector de salida, con el objetivo de predecir el horizonte completo. La segunda propuesta (denominada LSTM-FC [25]) es similar a la primera, pero con tres capas densas en vez de una (figura 3.12). La superposición de estas dos capas densas adicionales aplica una operación de interpolación cuadrática, con el objetivo de "suaviza" la curva definida por las posiciones predichas, y que se asimile mejor a la noción de trayectoria propia de los aviones. Ambas propuestas se componen de 30 celdas en cada capa LSTM, utilizan un tamaño de ventana de 55 vectores y la función de activación sigmoide.

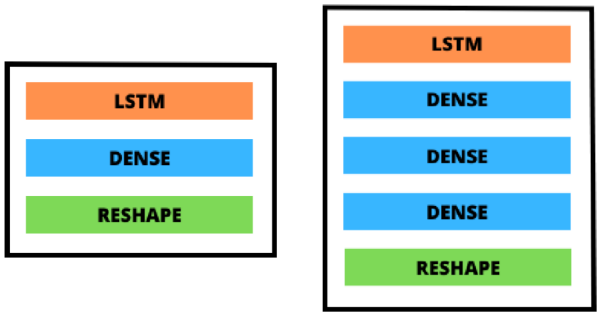

Figura 3.12: Estructura de las propuestas basadas en LSTM. Fuente: [25]

La tercera propuesta es una TFT similar a la descrita en el apartado 3.2.2.4. En este modelo se utilizan cuatro cabezas de atención, un tamaño de ventana de 60 vectores y 320 celdas LSTM. Para entrenar y evaluar a las propuestas, se ha utilizado un conjunto de datos que contiene las trayectorias de vuelos con destino el aeropuerto Adolfo Suárez Madrid-Barajas, desde enero hasta septiembre de 2022. Este conjunto de datos es el mismo que se utiliza en este proyecto, y que se describe en el apartado 4.1. Tanto el entrenamiento como la evaluación utiliza trayectorias de los nueve meses.

En la tabla 3.1 se muestra el error cometido por estas propuestas sobre las características que predicen (longitud, latitud y altitud). Las métricas utilizadas son el error absoluto medio (MAE) y la raíz del error cuadrático medio (RMSE) (estas métricas se desarrollan en el apartado 4.4). La longitud y latitud se miden en grados, mientras que la altitud se mide en pies.

3.3.1. Discusión

La propuesta TFT presenta el mejor desempeño global, destacando especialmente en la predicción de la altitud, y también muestra buenos resultados en latitud y longitud cuando se considera el RMSE. Aunque LSTM-FC tiene el menor MAE en latitud, TFT presenta un RMSE notablemente menor, lo que indica que comete errores menos drásticos. Sucede lo mismo respecto a la longitud: aunque LSTM tiene el menor MAE, TFT tiene un RMSE menor. Sobre la

Modelo	Dataset entrenamiento	Dataset prueba	MAE			RMSE		
			Latitud	Longitud	Altitud	Latitud	Longitud	Altitud
LSTM	Completo	Completo	0,0216	**0,0364**	242,8	0,091	0,147	820,2
LSTM-FC	Completo	Completo	**0,0208**	0,0373	259,2	0,091	0,149	820,2
TFT	Completo	Completo	0,0216	0,0574	**189,8**	**0,0403**	**0,0834**	**465,9**

Tabla 3.1: Resultados del estado del arte. La latitud y la longitud se expresan en grados, y la altitud, en pies.

altitud, TFT es claramente el mejor modelo, con el menor MAE y RMSE, demostrando una gran precisión en esta variable.

TFT muestra una capacidad de predicción de trayectorias 4D considerablemente buena, pero resulta una arquitectura compleja que requiere de elevados recursos computacionales y temporales para su entrenamiento. Junto con esto, el tamaño de ventana que requieren estos tres modelos es de 55 vectores para las basadas en LSTM y 60 para TFT, lo que supone un problema a la hora de realizar predicciones en el intervalo temporal tras el despegue y en trayectorias cortas, puesto que no hay suficientes vectores para alimentar a la red.

TSMixer es una arquitectura ligera, por lo que la cantidad de recursos que requiere es mucho menor que TFT. Sin embargo, para que esta mejora en el entrenamiento resulte realmente interesante, es necesario que ofrezca unos resultados similares o mejores que TFT. Por su parte, CfC no es tan eficiente como TSMixer, pero lo es más que TFT (gracias a que no es necesario el *solver*). Junto con esto, CfC es capaz de adaptarse a entradas con intervalos irregulares, por lo que es posible que requiera un tamaño de ventana inferior para conseguir el mejor resultado posible, lo cual a su vez permite comenzar a realizar predicciones poco tiempo después del despegue.

Capítulo 4

Diseño experimental

En este apartado se detalla el conjunto de datos empleado, junto con la descripción de la implementación de las propuestas, así como las métricas utilizadas para los entrenamientos y evaluaciones. A continuación, se explica la especificación e implementación de la herramienta de visualización de predicciones.

4.1. Descripción de los datos

Los datos utilizado en este proyecto son, principalmente, vectores de estado reconstruidos a partir de mensajes ADS-B obtenidos de la red OpenSky, que se complementan con otros archivos que contienen información sobre los planes de vuelo junto con datos básicos sobre los aeropuertos. Los vectores de estado reúnen los datos recibidos en distintos mensajes ADS-B cada 5 segundos, con el fin de obtener una descripción completa del estado del avión de forma periódica. Este conjunto de datos es el mismo que se utiliza en [26], donde también se documenta el proceso de preparación seguido para asegurar la calidad de los datos para el propósito predictivo planteado en este proyecto.

El conjunto de datos procesado (*smart data*) consiste un conjunto de 7146 trayectorias con destino al aeropuerto Adolfo Suárez Madrid-Barajas, provenientes de una selección de 40 de los aeropuertos internacionales europeos con mayor volumen de tráfico anual, realizadas entre el 1 de enero y el 30 de septiembre de 2022. El conjunto, almacenado en varios archivos en formato parquet (un formato de compresión de datos orientado a columnas), se compone de los vectores que forman dichas trayectorias y que están espaciados temporalmente cada 15 segundos. Cada vector tiene 46 atributos, de los cuales solo se describirán los más relevantes para el proyecto (tabla 4.1), que son las mismas que las usadas en [26] (para asegurar la comparabilidad de los resultados), añadiendo la velocidad vertical, puesto que se considera que es un atributo que puede aportar información relevante. Así, las características utilizadas para las predicciones serán *latitude*, *longitude*, *altitude*, *track*, *sector*, *hav_ distance*, *speed*, *vspeed* y *aerodromeOfDeparture*, mientras que las características predichas serán *latitude*, *longitude* y *altitude*.

El conjunto de datos se divide en meses, y para cada mes se reparten las trayectorias en tres subconjuntos disjuntos de entrenamiento, validación y prueba (tabla 4.2) para asegurar una estratificación homogénea de los datos desde el punto de vista temporal. Este conjunto de datos, que denominamos çompleto", se utilizará para el entrenamiento de los modelos finales. No obstante, el tamaño de este conjunto de datos dificultaría el proceso de optimización de hiperpa-

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 YYYY-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 4.1: Descripción de los atributos del conjunto de datos

rámetros de los diferentes modelos, por lo que previamente se realizará un estudio exploratorio de los hiperparámetros sobre un subconjunto representativo de los datos. En particular, utilizaremos los datos correspondientes al mes de enero, entrenando los sucesivos modelos con el subconjunto de entrenamiento y validándolos con el subconjunto de validación correspondiente

Subconjunto	**Trayectorias**	**Vectores**	**Porcentaje**
Entrenamiento (train)	5.153	2.141.283	72 %
Validación (val)	915	383.183	13 %
Evaluación (test)	1.078	446.642	15 %

Tabla 4.2: Distribución de los subconjuntos del conjunto de datos

4.2. Estrategia de predicción

Definir una estrategia para llevar a cabo la predicción de múltiples pasos futuros resulta complejo, debido al incremento de la incertidumbre y la acumulación de errores al aumentar el horizonte. En el artículo [27] se estudian y comparan diferentes estrategias para la predicción de múltiples pasos futuros:

- **Recursiva**: Predice iterativamente el siguiente paso, incorporándolo a la siguiente entrada, lo que provoca la acumulación de errores a largo plazo.
- **Directa**: Entrena un modelo diferente para predecir cada paso individualmente. Evita la acumulación de errores, pero requiere múltiples modelos.
- **DirRec**: Combina las estrategias recursiva y directa, entrenando un modelo para cada paso que utiliza tanto pasos originales como predicciones anteriores.
- **MIMO (Multi-Input Multi-Output)**: Entrena un único modelo que predice todo el horizonte al mismo tiempo. Mantiene la dependencia estocástica y evita la acumulación de errores.
- **DIRMO**: Similar a MIMO. Entrena un modelo para predecir un subconjunto de pasos futuros, y aplica el mismo modelo recursivamente para predecir el resto de pasos, equilibrando la dependencia estocástica y la flexibilidad del modelado.

El artículo concluye que las estrategias de múltiples salidas (MIMO y DIRMO) son las que ofrecen mejores resultados en la predicción de múltiples pasos futuros. Sin embargo, DIRMO incorpora un hiperparámetro para el cual resulta complicado escoger un valor adecuado, por lo que MIMO es una mejor opción. En [26] se estudia qué estrategia ofrece mejores resultados (recursiva o MIMO) al predecir trayectorias de aeronaves utilizando redes LSTM, con idénticos resultados: el modelo que utiliza MIMO obtiene un valor de MSE (0,000129) un orden de magnitud menor que el obtenido con la la estrategia recursiva (0,0044). En base a estos resultados, se utilizará la estrategia MIMO en las arquitecturas propuestas en este proyecto.

4.3. Modelos propuestos

Se proponen tres modelos para la predicción de trayectorias aéreas. El primero, TSMixer, se ha implementado siguiendo exactamente la arquitectura descrita en [4], y que se ha detallado en el apartado 3.2.2.5. Las otras dos propuestas se han diseñado utilizando CfC como parte de la arquitectura. La primera de estas dos propuestas consiste en una capa CfC que recibe y procesa las entradas, a la cual se le aplica *dropout*, y se le concatena un bloque de proyección temporal similar al de TSMixer, con el objetivo de realizar la predicción del horizonte completo (figura 4.1).

La tercera arquitectura propuesta, a la cual se le ha denominado "CfCTime", se basa en procesar como característica adicional (nombrada *time*) los segundos transcurridos desde el vector anterior, con el objetivo de aprovechar la capacidad de CfC para adaptarse a series temporales con intervalos irregulares. De esta forma, esta arquitectura recibe dos entradas que se procesan de forma diferente (figura 4.2): primero, se reciben todas las características (excepto los segundos transcurridos) y se conectan a una capa densa (con *dropout*). El resultado se concatena con *time*,

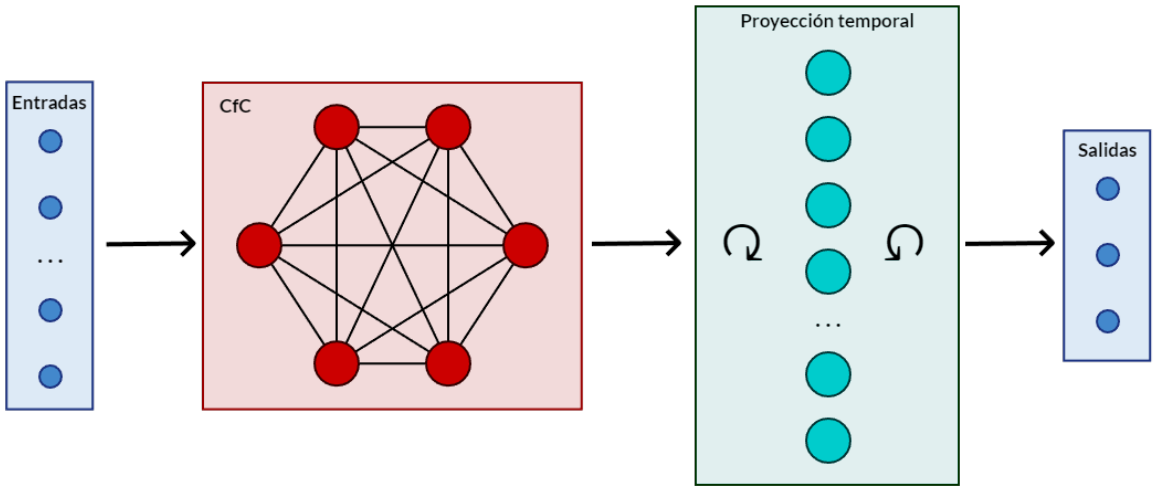

Figura 4.1: Representación de la propuesta “CfC”.

y se entrega el conjunto a una capa CfC que, como en el caso anterior, devuelve tres salidas, a las cuales se les suma el valor de la longitud, latitud y altitud recibidas como entrada (conexión residual). Finalmente, se realiza la predicción final en una capa de proyección temporal.

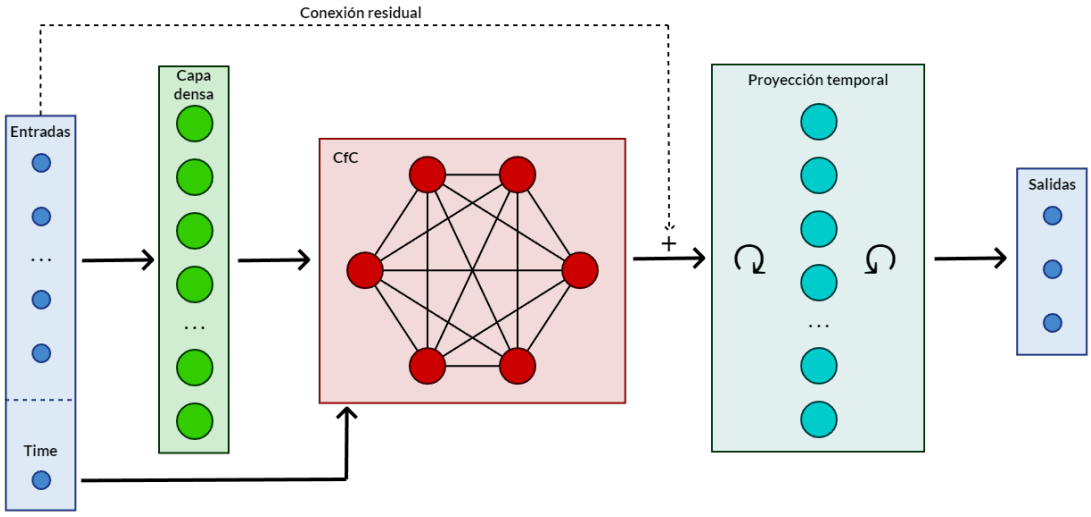

Figura 4.2: Representación de la propuesta “CfCTime”.

4.4. Métricas

La métrica escogida para representar la función de pérdida de los modelos es el Error Cuadrático Medio (MSE). Esta medida se define como la media de los cuadrados de los errores, donde el error es la diferencia entre el valor real (y_i) y el valor predicho por el modelo ($\hat{y}_i$) (ecuación 4.1).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.1}$$

Como las diferencias se elevan al cuadrado, el MSE siempre es positivo, penaliza en mayor medida los errores más grandes, en relación a los errores de menor magnitud. Cuanto más baja sea esta métrica, mejores serán los resultados del modelo. Sin embargo, como el MSE utiliza diferencias al cuadrado las magnitudes obtenidas también lo estarán, por lo que resulta interesante utilizar otras métricas que mantengan la misma magnitud que los atributos a predecir. Así, las métricas que se utilizan para comparar los modelos resultantes son el Error Absoluto Medio (MAE), que se define como la media de los valores absolutos de los errores (ecuación 4.2); y la Raíz del Error Cuadrático Medio (RMSE) que, como indica el nombre, se calcula aplicando la raíz cuadrada al MSE (ecuación 4.3). Con el objetivo de realizar comparaciones más precisas, se calculará el MAE y el RMSE cometido en cada característica objetivo (longitud, latitud y altitud) por cada modelo.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.2}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.3}$$

4.5. Visor de trayectorias 4D

4.5.1. Análisis

Las métricas definidas en el apartado 4.4 permiten evaluar el desempeño general de los modelos, pero no ofrecen información específica sobre cómo los modelos realizan las predicciones (por ejemplo, detectar tramos de las trayectorias donde se realizan, generalmente, peores predicciones).

Resulta interesante poder visualizar las trayectorias y las predicciones que realizan los modelos, ya que esto facilita el análisis de los resultados y la detección de patrones en los datos y las predicciones. Es por esto que se ha propuesto el desarrollo de un *dashboard* o visor de trayectorias de vuelo. Esta herramienta debe ser capaz de representar las trayectorias de las aeronaves junto con las predicciones de los modelos, de tal forma que se pueda comprender de forma sencilla las diferencias entre los datos reales y las predicciones. Debido al volumen de trayectorias disponibles, resulta necesario disponer de filtros para facilitar la búsqueda y selección de trayectorias. Una característica necesaria para esta herramienta es que sea agnóstica, es decir, debe ser capaz de utilizar modelos de *deep learning* sin la necesidad de realizar adaptaciones específicas para cada uno. Entonces, los objetivos principales del visor son los siguientes:

- **VIS-01** Visualizar trayectorias 4D.
  - **VIS-01.1** Comparar las predicciones de los modelos con los vectores reales.
  - **VIS-01.2** Filtrar trayectorias 4D.
- **VIS-02** Desacoplar el visor de la implementación específica de los modelos.

Los modelos realizan la predicción del horizonte en base a un vector, utilizando como secuencia de entrada los puntos de la ventana. Por ello, el visor debe facilitar que el usuario seleccione cualquier punto de la trayectoria para visualizar qué pasos futuros se han predicho y qué vectores se han utilizado para ello. A su vez, esto permitirá visualizar cómo evolucionan las predicciones al avanzar en la trayectoria. Por otro lado, resulta interesante poder visualizar la predicción de una trayectoria completa, de tal forma que se pueda comprender el desempeño general que muestran los modelos.

El conjunto de datos utilizado contiene trayectorias 4D agrupadas por el mes en el que se realizaron (apartado 4.1), por lo que resulta intuitivo poder localizarlas a partir de la fecha en que se realizaron. Como el aeropuerto de llegada es siempre el mismo (Adolfo Suárez Madrid-Barajas), las trayectorias que comienzan desde el mismo aeropuerto de salida mostrarán un desarrollo similar, lo que motiva a incluir un filtro de trayectorias a partir del aeropuerto de salida.

4.5.2. Diseño

El visor de trayectorias 4D se estructurará en dos secciones principales: el panel de control y los gráficos (figura 4.3). El panel de control se compone de una serie de selectores y filtros para gestionar los datos que se visualizarán en las representaciones gráficas. Concretamente, permite la selección del modelo, filtrar y escoger las trayectorias, y elegir el tipo de predicción representada: la trayectoria completa o un vector específico, permitiendo elegir este vector.

En la sección de los gráficos se representa visualmente la trayectoria y las predicciones siguiendo la configuración del panel de control. Se compone de dos gráficos: un mapa terrestre que representa los vectores de la trayectoria y las predicciones según su longitud y latitud, y un diagrama de dispersión que muestra la evolución de la altitud en función de la longitud. La combinación de estos dos diagramas permite visualizar las tres variables a predecir: longitud, latitud y altitud.

Los dos diagramas permiten la visualización de las predicciones de un vector específico y de la trayectoria completa. Sin embargo, representar las predicciones realzadas en todos los vectores de la trayectoria supondría la superposición de las secuencias predichas, dificultando la comprensión de los resultados. Para evitar esto, se representan las predicciones en intervalos del tamaño del horizonte, de tal forma que para cada punto real exista un único punto predicho (figura 4.4).

Se utilizará un esquema de colores para diferenciar entre los vectores que representen la ventana, el horizonte (valores reales), las predicciones y el resto de puntos de la trayectoria (figura 4.5). Junto a esto, se conectarán los puntos dentro de una secuencia predicha, para facilitar la comprensión de la evolución de las predicciones.

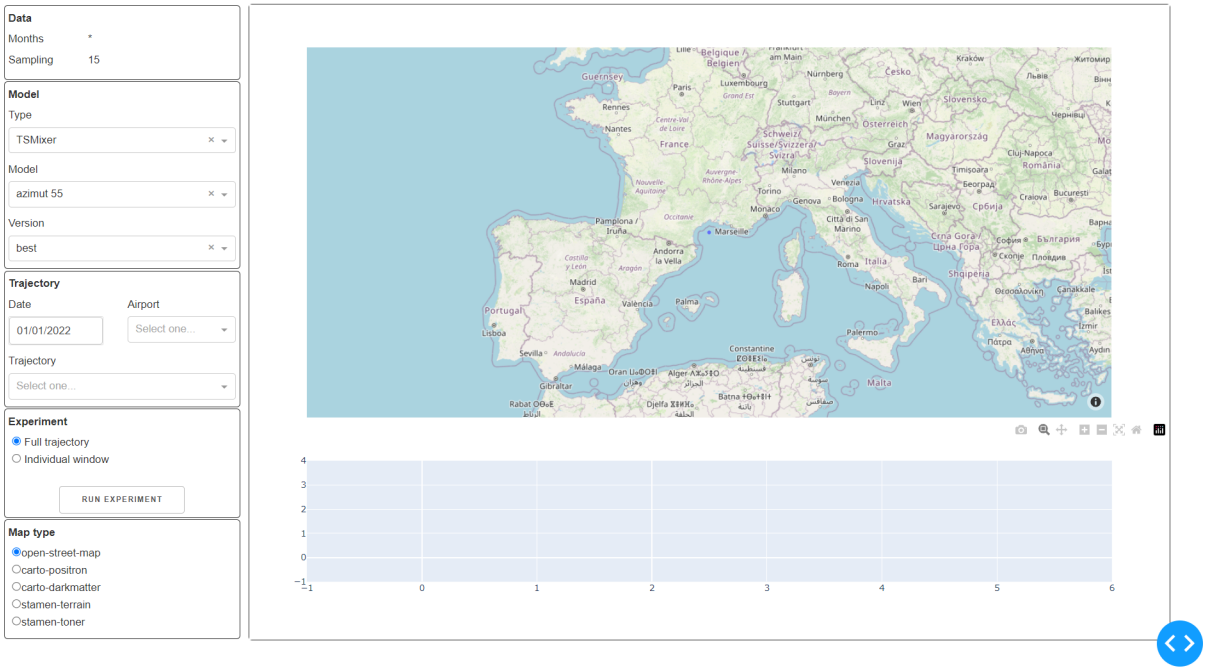

Figura 4.3: Interfaz del visor de trayectorias.

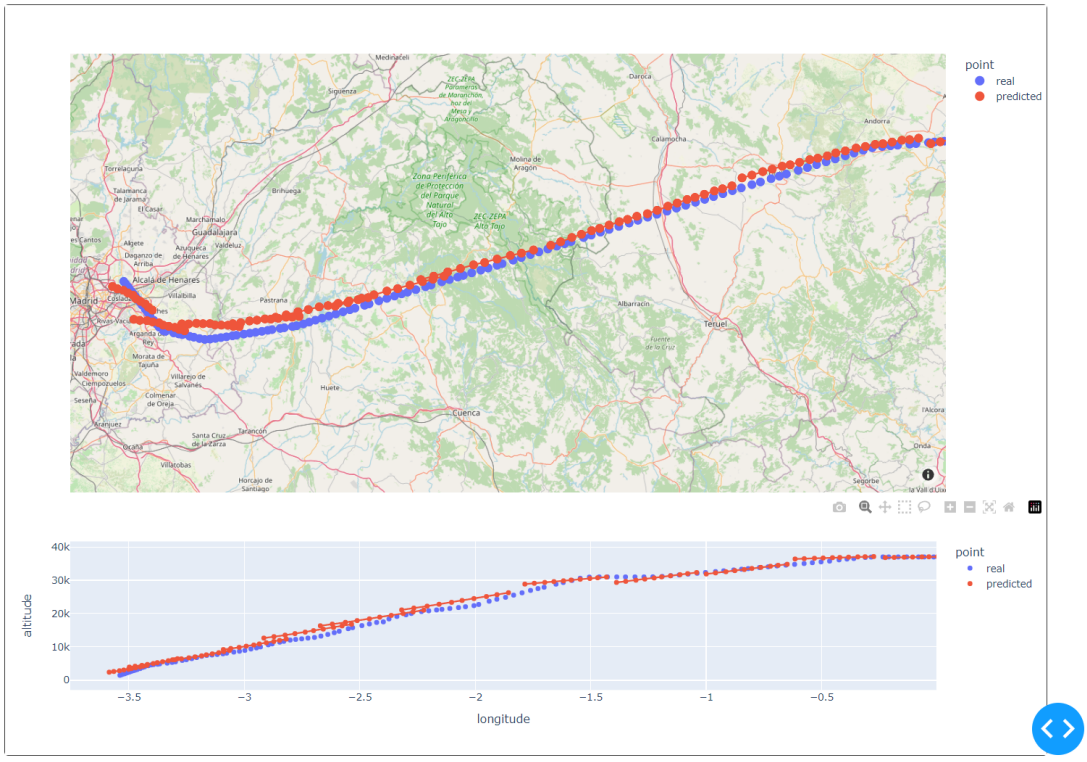

Figura 4.4: Sección de una predicción de trayectoria completa. Los puntos reales se muestran en azul, y los predichos, en rojo.

Figura 4.5: Esquema de colores para los vectores de los gráficos.

Para conseguir el objetivo VIS-02, se ha definido una clase genérica, "ModeloBase", que generaliza la implementación de las arquitecturas. Esto permite desacoplar el visor de la estructura específica de cada modelo, de tal forma que se pueda utilizar en el visor cualquier modelo que herede de esta clase. En la figura 4.6 se muestra el diagrama de clases que describe la estructura utilizada en la implementación de los modelos y el visor. No se han incluido los constructores, los *getters* ni los métodos heredados para facilitar la comprensión del diagrama. Las funcionalidades que describe esta clase son las siguientes:

- Creación de un modelo a partir de un conjunto de hiperparámetros.
- Carga de un punto de control de un modelo. Para ello, se utiliza el fichero de configuración almacenado en el directorio del modelo.
- Cargar datos a partir del mes y el tipo de conjunto de datos (entrenamiento, validación y prueba), entre otros filtros.
- Procesar y reestructurar los datos para proporcionárselos al modelo.
- Entrenar el modelo.
- Realizar una predicción a partir de un conjunto de datos procesado.
- Evaluar el modelo.

4.6. Herramientas y librerías para la implementación

El lenguaje de programación utilizado para la implementación de los modelos es Python, a través de las librerías TensorFlow [28] y Keras [15], que permiten la creación y entrenamiento de redes neuronales profundas. La gestión del entorno y sus librerías se realiza mediante Anaconda [1], una distribución de Python orientada a la computación científica, que facilita la gestión e implementación de librerías. Anaconda incluye Jupyter Notebook [22], una aplicación que permite crear documentos interactivos utilizando diferentes lenguajes de programación, en este caso Python.

Para la búsqueda del mejor conjunto de hiperparámetros de cada arquitectura se utiliza Weights & Biases (Wandb) [31], una plataforma de seguimiento, visualización y automatización para el aprendizaje automático. Esta herramienta facilita la paralelización de los entrenamientos

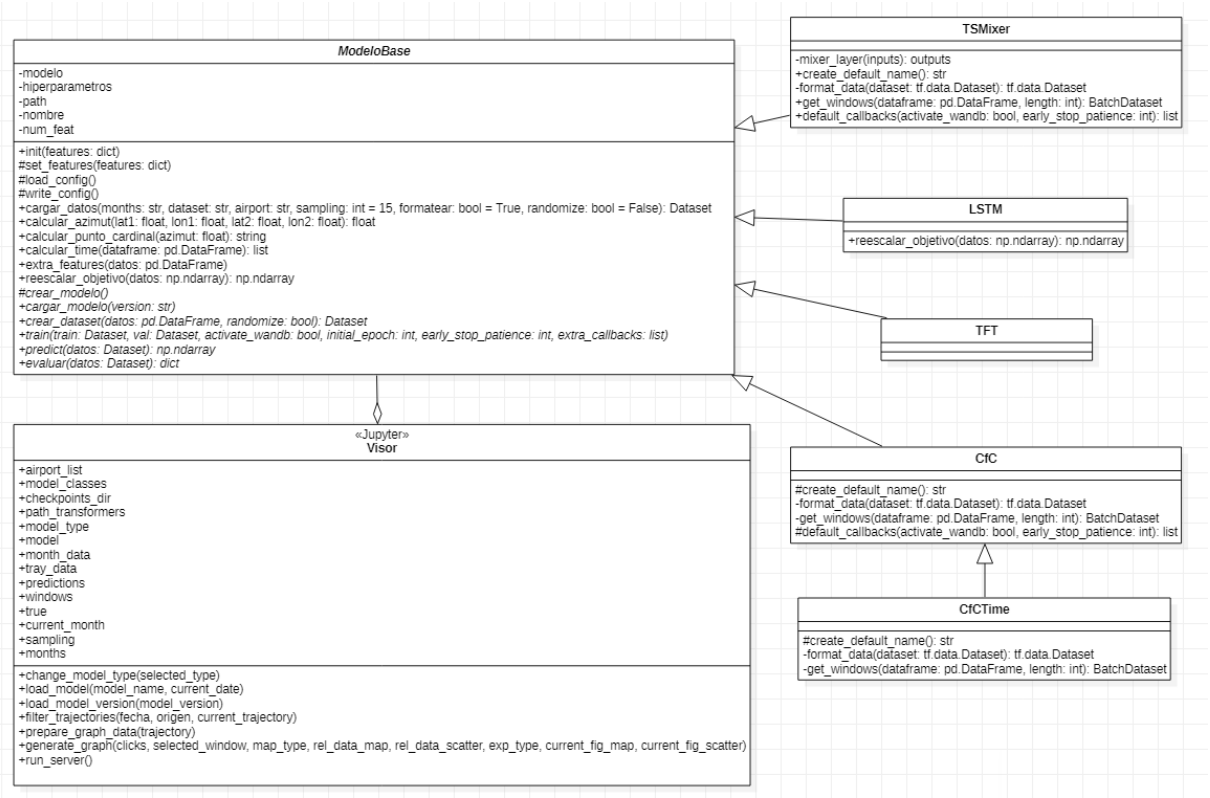

Figura 4.6: Diagrama de clases que describe la estructura de las arquitecturas y el visor.

en la búsqueda de los mejores hiperparámetros, que se realizarán tanto en el ordenador personal como en Google Colab [5], un servicio de Google que permite la ejecución de cuadernos de Jupyter en la nube, de forma gratuita.

Capítulo 5

Experimentación y evaluación

En este capítulo se realiza un estudio del impacto de los hiperparámetros sobre TSMixer, CfC y CfCTime. Para ello, se definen un conjunto de valores para cada hiperparámetro y se realizan entrenamientos con las posibles combinaciones de los mismos, utilizando los datos correspondientes a enero. Tras obtener conclusiones sobre su comportamiento, se procede a realizar la optimización de los hiperparámetros, con el objetivo de conseguir los modelos que reporten mejores resultados. Finalmente, se entrenarán los modelos finales utilizando el conjunto de datos completo, y se compararán los resultados obtenidos con los de los modelos del estado del arte.

Antes de comenzar, se explicarán los hiperparámetros que definen cada modelo. Todas las propuestas tienen en común los hiperparámetros que se detallan en la tabla 5.1. Estos hiperparámetros influyen de manera diferente en cada tipo de arquitectura propuesta, por lo que se debe estudiar su impacto en cada modelo, junto con los hiperparámetros específicos de cada uno. Por otro lado, todas las propuestas utilizan la configuración de la tabla 5.2.

Hiperparámetro	**Descripción**	**Ejemplo**
Batch Size	Cantidad de muestras del conjunto de entrenamiento que se procesarán antes de actualizar los pesos del modelo.	64
Dropout	Porcentaje de neuronas que no se activarán durante un lote (*batch*). En cada modelo afecta a capas diferentes.	0,1
Learning Rate (lr)	Define el grado en que se actualizarán los pesos de los modelos en función del error cometido.	0,0005
Lookback	Tamaño de ventana utilizado.	55
Función de activación	Función matemática que se aplica a las salidas de una capa, introduciendo así no linealidades al modelo.	ReLu

Tabla 5.1: Hiperparámetros comunes a todas las propuestas.

Se ha utilizado Weights & Biases (Wandb) [31] para la automatización y análisis de los entrenamientos. Esta plataforma permite registrar, analizar y visualizar las métricas resultantes de los entrenamientos. Junto a esto, Wandb permite definir *sweeps*, que consisten en automatizaciones para gestionar y ejecutar los entrenamientos, de cara a optimizar los hiperparámetros y analizar los resultados (figura 5.1).

Hiperparámetro	Descripción	Enero	9 meses
Early Stop Patience	Número de épocas durante las cuales no se ha percibido una mejora en la pérdida sobre el conjunto de validación, deteniendo así el entrenamiento.	10	15
Epochs	Número de épocas máximas que puede durar el entrenamiento.	100	300
Lookforward	Tamaño del horizonte.	10	10

Tabla 5.2: Configuración común a todas las propuestas.

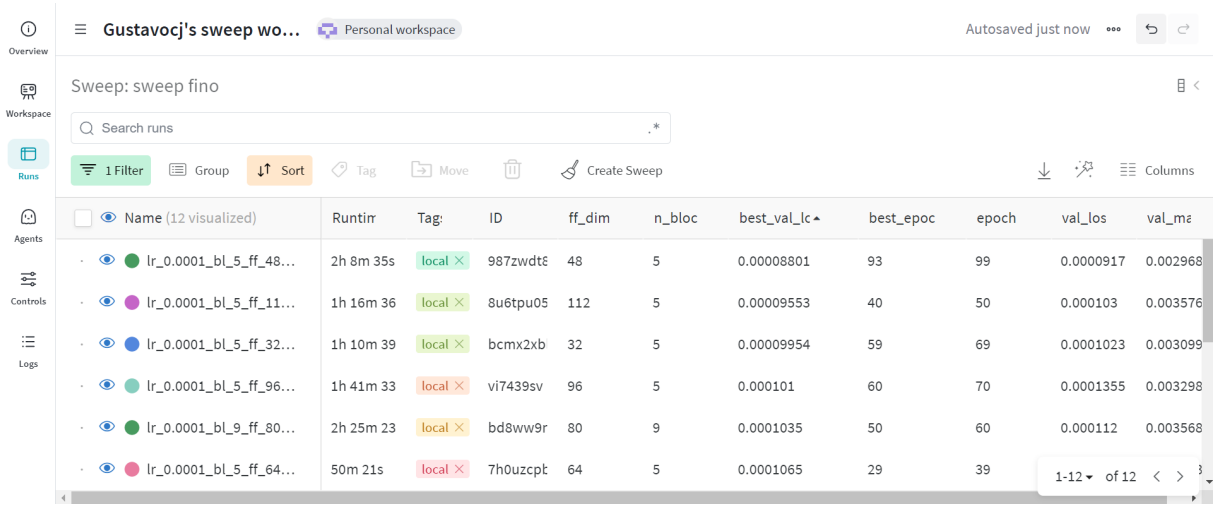

Figura 5.1: Interfaz de un *sweep* en Wandb.

Una de las funciones disponibles en Wandb es el cálculo de la importancia y correlación de cada hiperparámetro sobre la métrica objetivo (MSE). Esto facilita comprender el impacto que tiene cada hiperparámetro sobre la arquitectura, lo que resulta interesante de cara a escoger un conjunto de valores óptimo a partir de las pruebas realizadas.

5.1. Estrategias para el análisis de hiperparámetros

La estrategia más sencilla e intuitiva es el método de la "malla" (*grid*), que consiste en entrenar un modelo para cada combinación de valores de hiperparámetros, a partir de una serie de valores elegidos dentro de un determinado rango para cada uno de estos hiperparámetros. El modelo resultante se evalúa con respecto al conjunto de validación, con el fin de identificar aquel que arroja un mejor resultado. Sin embargo, esta técnica resulta demasiado exhaustiva: por ejemplo, si queremos probar únicamente dos valores distintos para cuatro hiperparámetros, sería necesario entrenar 16 (2^4) modelos para considerar todas las combinaciones. Por tanto, incorporar a la búsqueda valores adicionales de hiperparámetros incrementa el número de entrenamientos considerablemente, lo que conlleva un consumo excesivo de recursos computacionales y temporales.

A causa de esto, se han estudiado otras estrategias para el análisis de hiperparámetros. La primera que se ha considerado es el método "aleatorio", similar al de la malla, pero escogiendo los conjuntos de hiperparámetros de forma aleatoria. Esto permite terminar el experimento cuando se considere que se han evaluado suficientes modelos, pero al ser aleatorio, puede ocurrir que algún valor concreto no se haya utilizado lo necesario para sacar conclusiones. Este método permite definir rangos de valores continuos de hiperparámetros (frente a la malla, que sólo permite valores discretos).

La tercera estrategia evaluada es la Optimización Bayesiana [24]. Como su nombre indica, esta técnica se basa en el Teorema de Bayes, y busca predecir cuál será la pérdida en el conjunto de validación de un modelo, a partir de los resultados del resto de modelos entrenados. Esto permite descartar configuraciones que pudiesen generar peores resultados, y de esta forma reducir considerablemente el número de modelos entrenados hasta conseguir uno cuyos resultados sean suficientemente buenos (esto se debe a que la Optimización Bayesiana no finaliza por sí misma).

En las siguientes secciones de este capítulo se concluye que la Optimización Bayesiana no es una estrategia adecuada para el análisis del impacto de los hiperparámetros, puesto que no se obtienen suficientes resultados de todos los valores a estudiar. Sin embargo, esta técnica puede resultar especialmente útil de cara a la optimización de hiperparámetros utilizando el conjunto de datos completo, debido a que se reduce considerablemente la cantidad de modelos que hay que entrenar hasta obtener el mejor.

Finalmente, la técnica Hyperband [16] permite descartar modelos que no muestren resultados satisfactorios durante el entrenamiento, al compararlos con la evolución del resto de modelos. Esta evaluación se realiza en épocas concretas establecidas al inicio del experimento. El problema es que Hyperband está diseñado para el entrenamiento de múltiples modelos en paralelo, por lo que no es viable en este proyecto, en el que se dispone únicamente de dos entornos de entrenamiento. Hyperband se puede combinar con cualquiera de las tres técnicas anteriores, mostrando una buena sinergia con la Optimización Bayesiana.

5.2. TSMixer

Como se ha descrito en el apartado 3.2.2.5, TSMixer se compone de varias capas de mezcla y la proyección temporal. Por ello, los hiperparámetros que definen a esta arquitectura (además de los anteriormente descritos) son el número de bloques de mezcla y el tamaño de la capa densa de la fase de mezcla de características (se utiliza el mismo número de neuronas en esta capa en cada bloque de mezclas) (tabla 5.3). Según los resultados obtenidos en el artículo de TSMixer [4], la función de activación que mejor funciona en general para esta arquitectura es la ReLu, por lo que se utilizará la misma por defecto.

Hiperparámetro	**Descripción**	**Ejemplo**
n_blocks	Número de bloques de mezcla.	5
fc_dim	Número de neuronas en la capa densa de la fase de mezcla de características.	16

Tabla 5.3: Hiperparámetros específicos de la propuesta TSMixer.

5.2.1. Primer experimento

La estrategia inicial que se siguió para evaluar el impacto de los hiperparámetros sobre TSMixer consistió en definir, para cada hiperparámetro, tres posibles valores, con el objetivo de analizar y comprender las tendencias, a grandes rasgos, que muestran los resultados. La tasa de aprendizaje (*lr*) se fija en 0,0001 (este valor se utiliza en [4]), con el objetivo de reducir el número de combinaciones en este experimento. El método utilizado fue el de la malla, que consiste en probar todas las combinaciones posibles de hiperparámetros según los valores establecidos. La tabla 5.4 describe la configuración de este experimento.

Hiperparámetro	**Valores**
Método	malla
batch_size	32, 64, 128
dropout	0,1, 0,3, 0,5
fc_dim	1, 8, 16
lookback	8, 32, 64
lr	0,0001
n_blocks	1, 3, 5

Tabla 5.4: Valores de hiperparámetros evaluados en el primer experimento de TSMixer.

El número de combinaciones de hiperparámetros con esta configuración es de 243, mientras que el tiempo de entrenamiento de cada modelo varía entre 10 minutos y 3,5 horas por lo que, debido a las restricciones de tiempo, se llevaron a cabo 186 entrenamientos, y se evaluó la posibilidad de utilizar otras estrategias diferentes a la de la malla. Pese a esto, se entrenaron suficientes modelos como para representar el impacto a grandes rasgos de los hiperparámetros analizados.

En la tabla 5.5 se muestran los resultados de los cinco mejores modelos, cuya pérdida en el conjunto de validación (*val_loss*) es del orden de $1{,}1e^{-4}$. Todos estos modelos se componen de 5 bloques de mezclas, por lo que puede resultar interesante evaluar modelos con más bloques; junto a esto, se puede ver una tendencia en que aumentar el tamaño de los lotes, el *dropout* y el tamaño de la ventana mejora los resultados. El tamaño de la capa densa no parece seguir un patrón claro, por lo que es posible que su valor se vea fuertemente afectado por el resto de hiperparámetros; aún así, se estudiará un rango de valores mayor.

5.2.2. Segundo experimento

En base a los resultados del experimento anterior, el análisis se centró en explorar rangos de valores más amplios para el *dropout*, la dimensión de la capa densa y el número de bloques. Puesto que aumentar el tamaño de la ventana parecía mostrar mejores resultados, se fijó en 55 vectores, como las redes LSTM y TFT de [26]. En el siguiente apartado se realizará un análisis más preciso de este hiperparámetro. Por otro lado, el tamaño de los lotes se fijó en 64, ya que este valor muestra un buen equilibrio entre los resultados obtenidos y el tiempo de ejecución.

El segundo experimento se realizó en varias etapas puesto que, según los resultados obtenidos en una etapa, se definía el objetivo de la siguiente. Por ejemplo, en la primera etapa se utilizó

	modelo 1	**modelo 2**	**modelo 3**	**modelo 4**	**modelo 5**
batch_size	32	64	32	64	64
dropout	0,3	0,5	0,3	0,3	0,3
fc_dim	1	8	1	16	1
lookback	64	32	32	64	64
n_blocks	5	5	5	5	5
mejor época	25	29	13	27	24
Duración	2h 46m 47s	35m 42s	1h 9m 25s	53m 47s	38m 55s
mejor val_loss	**0,000112**	**0,0001134**	**0,0001174**	**0,000122**	**0,0001269**

Tabla 5.5: Mejores modelos del primer experimento de TSMixer, ordenados de mejor a peor.

la Optimización Bayesiana con Hyperband (BOHB), pero se observó que los entrenamientos terminaban demasiado rápido, sin dar margen de que los modelos pudiesen converger (esto se debe al funcionamiento de Hyperband, apartado 5.1). En la tabla 5.6 se especifican las configuraciones de cada etapa.

Hiperparámetro	**Etapa 1**	**Etapa 2**	**Etapa 3**	**Etapa 4**	**Etapa 5**
Método	BOHB	malla	malla	malla	malla
batch_size	64	64	64	64	64
lookback	55	55	55	55	55
lr	0,0001	0,0001	0,0001	0,0001	0,0001
dropout	0,1, 0,3, 0,5	0,1, 0,3, 0,5	0,1	0,1	0,1
fc_dim	1, 16, 32	1, 16, 32	32, 64, 128	256	128
n_blocks	1, 3, 5	1, 3, 5	3, 5, 7, 9	5, 9, 11	11

Tabla 5.6: Configuraciones de las etapas del segundo experimento de TSMixer.

En la tabla 5.7 se muestran las configuraciones de hiperparámetros y los resultados de los cinco mejores modelos obtenidos entre todas las etapas del segundo experimento. La pérdida en el conjunto de validación es del orden de $9e^{-5}$, por lo que existe una mejora respecto a los modelos del primer experimento. Se puede comprobar que el tiempo de entrenamiento se ha incrementado de forma general, alcanzando el modelo 1 las cinco horas y media. Respecto a los hiperparámetros, un *dropout* de 0,1 resulta el más adecuado para todas las configuraciones, mientras que el tamaño de la capa densa sigue sin mostrar una tendencia aparente. Aumentar el número de bloques beneficia al modelo, siendo los mejores valores 5 y 9 bloques. En general, los modelos de 7 bloques ofrecen peores resultados.

	modelo 1	**modelo 2**	**modelo 3**	**modelo 4**	**modelo 5**
batch_size	64	64	64	64	64
dropout	0,1	0,1	0,1	0,1	0,1
fc_dim	128	32	32	128	256
lookback	55	55	55	55	55
lr	0,0001	0,0001	0,0001	0,0001	0,0001
n_blocks	9	5	3	5	5
mejor época	70	51	27	31	26
Duración	5h 27m 34s	3h 3m 25s	1h 30m 34s	2h 4m 14s	2h 21m 38s
mejor val_loss	**0,00009004**	**0,00009229**	**0,00009238**	**0,00009373**	**0,00009449**

Tabla 5.7: Mejores modelos del segundo experimento de TSMixer, ordenados de mejor a peor.

5.2.3. Optimización de hiperparámetros

Antes de comenzar con la optimización de hiperparámetros, resulta interesante entrenar un modelo con los nueve meses de datos utilizando la mejor configuración hasta el momento. De esta forma, se puede comprobar si los resultados que se han obtenido sobre el mes de enero se corresponden con los del conjunto de datos completo. Así, se llevó a cabo el entrenamiento de un modelo con el conjunto de hiperparámetros del mejor modelo obtenido en el segundo experimento, sobre el conjunto de nueve meses. El resultado fue que la pérdida en el conjunto de validación era de 0,0002164, frente a los 0,00009 del modelo sobre un mes.

Se sospechaba que el modelo, al ser tan profundo (9 bloques y 128 neuronas en las capas densas), pudiese estar sobreajustándose a los datos. Por ello, se entrenaron modelos sobre el conjunto de nueve meses aumentando el *dropout*, cuyos resultados se muestran en la figura 5.2.

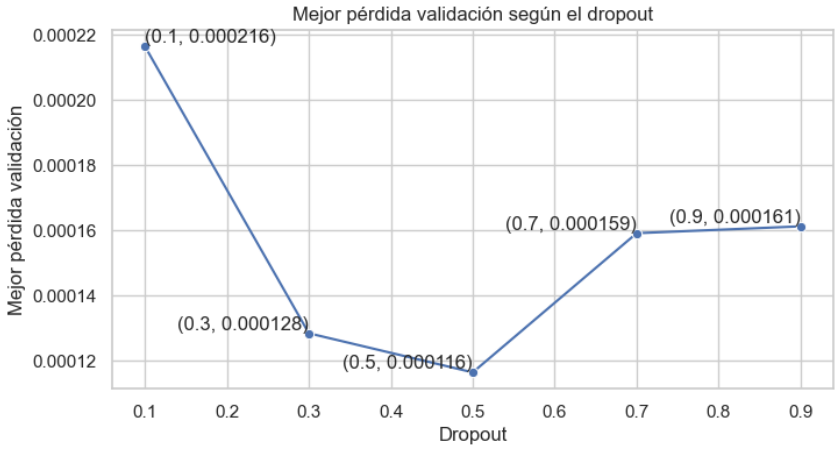

Figura 5.2: Evolución del mejor valor de la pérdida en el conjunto de validación según el *dropout*.

En efecto, aumentar el *dropout* evita que el modelo se sobreajuste, mejorando los resultados. Un valor del 0,5 ofrece los mejores resultados, de 0,000116, que se aproxima más al obtenido sobre un mes de datos. Entonces, se continuará utilizando la configuración de hiperparámetros de este modelo para la optimización de los siguientes hiperparámetros, excepto el *dropout*, que se mantendrá en 0,5.

5.2.3.1. Optimización del tamaño de ventana

En el segundo experimento del apartado anterior se fijó el tamaño de la ventana en 55 vectores, para poder estudiar otros hiperparámetros con más detalle. Ahora, resulta interesante afinar este hiperparámetro, para lo cual se entrenarán modelos sobre un mes de datos (enero) con tamaños de ventana desde 10 vectores a 40, en intervalos de diez. En la figura 5.3 se muestra un diagrama de los resultados, donde se puede comprobar que el error mínimo se obtiene con un tamaño de ventana de 20 vectores. Entonces, el siguiente paso es estudiar los valores en torno a 20, con el objetivo de obtener el mejor resultado posible. En la figura 5.4 se muestra la evolución de la pérdida en el conjunto de validación respecto a tamaños de ventana próximos a 20 vectores, donde se puede comprobar que el mejor tamaño de ventana se mantiene en 20 vectores, por lo que se utilizará este valor en los siguientes modelos.

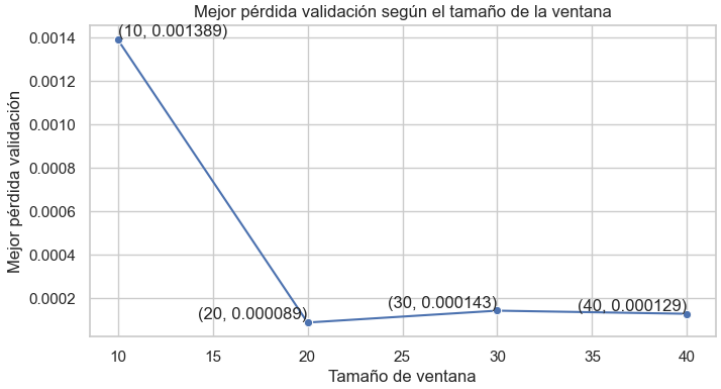

Figura 5.3: Evolución del mejor val_loss según el tamaño de la ventana.

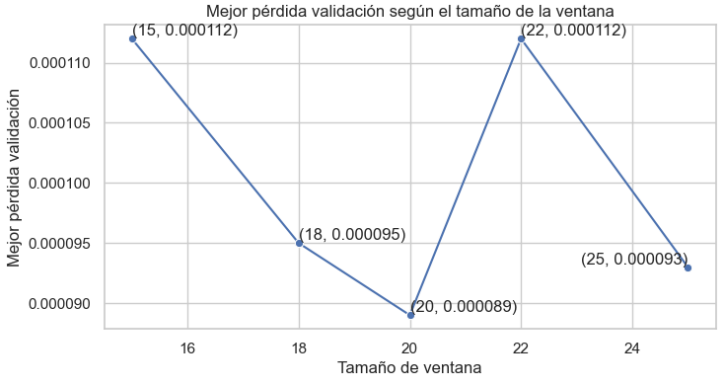

Figura 5.4: Evolución del mejor val_loss según el tamaño de la ventana.

Si comparamos en el visor las predicciones del modelo que utiliza un tamaño de ventana de 55 vectores (figura 5.5) con el que utiliza 20 (figura 5.6) se puede observar una mejora evidente. El primer modelo genera secuencias de predicciones ligeramente desviadas respecto a la trayectoria original, y la distancia entre los puntos dentro de la secuencia predicha es inferior a la de los valores reales. En la curva final de la trayectoria, la secuencia predicha se desvía de la original, pero se percibe la curvatura. Sin embargo, el modelo con ventanas de 20 vectores realiza predicciones mucho más suaves y cercanas a la realidad. Al final de la trayectoria, la predicción se ajusta mejor a la curva realizada antes de la llegada al aeropuerto.

La trayectoria escogida para la comparación comienza en Roma y termina en Madrid. Se ha utilizado el tramo final de esta trayectoria para mostrar las predicciones tanto en el descenso como en el tramo de altitud constante (altitud "crucero"), además de presentar algunos vectores faltantes y disponer de dos curvas: la primera suave y la segunda más pronunciada. Esto permite visualizar en una misma imagen cómo se comporta cada modelo en diferentes escenarios.

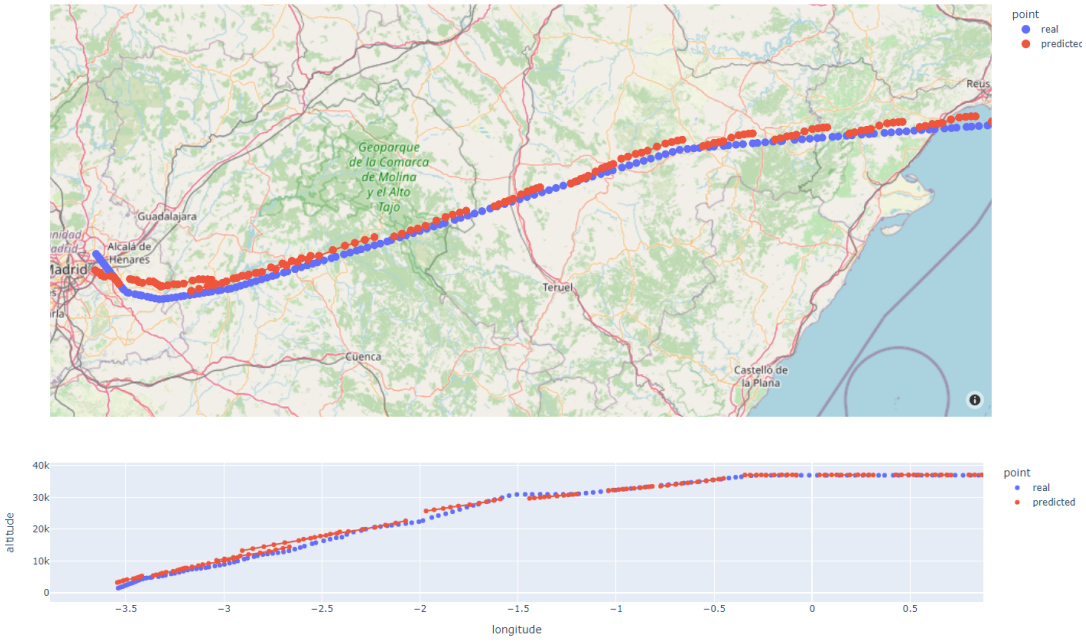

Figura 5.5: Visualización de la predicción utilizando 55 vectores como ventana.

5.2.3.2. Optimización del tamaño de las capas densas y el número de bloques

Como se ha visto en el apartado 5.2.2, el tamaño de las capas densas no muestra un patrón claro. Por ello, es necesario estudiar una mayor cantidad de valores para dilucidar su comportamiento. Entonces, se han entrenado modelos con valores para el tamaño de las capas densas desde 32 neuronas hasta 128, en intervalos de 16 neuronas. Además, se han utilizado modelos de 5 y 9 bloques de mezclas, puesto que son los que mejores resultados ofrecen.

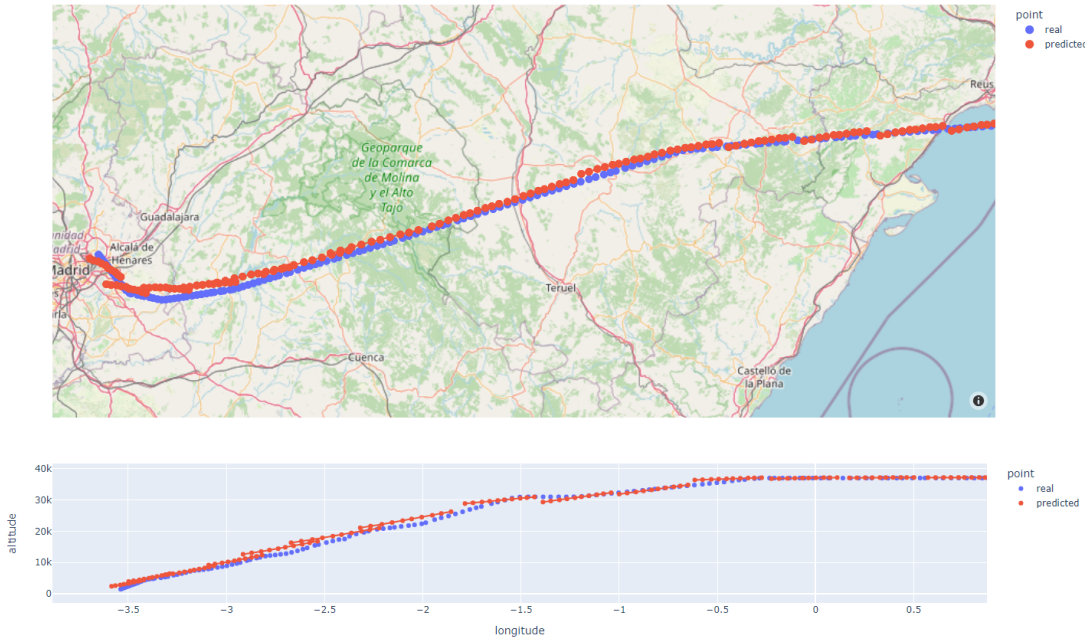

Figura 5.6: Visualización de la predicción utilizando 20 vectores como ventana.

En la tabla 5.8 se muestran los cinco mejores modelos obtenidos, donde se puede comprobar que el modelo con 5 bloques de mezclas y 48 neuronas para las capas densas es el que mejor resultados ofrece, ligeramente superior al modelo base actual (9 bloques de mezclas y 128 neuronas en las capas densas).

	modelo 1	**modelo 2**	**modelo 3**	**modelo 4**	**modelo 5**
fc_dim	48	112	32	96	80
n_blocks	5	5	5	5	9
mejor época	93	40	59	60	50
Duración	2h 8m 35s	1h 16m 36s	1h 10m 39s	1h 41m 33s	2h 25m 23s
mejor val_loss	**0,00008801**	**0,00009553**	**0,00009954**	**0,000101**	**0,0001035**

Tabla 5.8: Mejores modelos de la optimización de fc_dim en TSMixer, ordenados de mejor a peor.

Otro aspecto a destacar es que los modelos de 5 bloques ofrecen, en general, mejores resultados que los de 9, por lo que se han entrenado modelos con entre 3 y 7 bloques de mezclas para optimizar este hiperparámetro. En la figura 5.7 se muestra un diagrama con la evolución de los resultados, donde se puede comprobar que el mejor número de bloques se mantiene en 5.

Finalmente, falta optimizar el tamaño de las capas densas para lo cual, una vez más, se han entrenado modelos con valores cercanos a 48 neuronas: entre 42 y 54 en intervalos de tres. Los resultados se muestran en el diagrama de la figura 5.8, en el cual se aprecia que el número óptimo

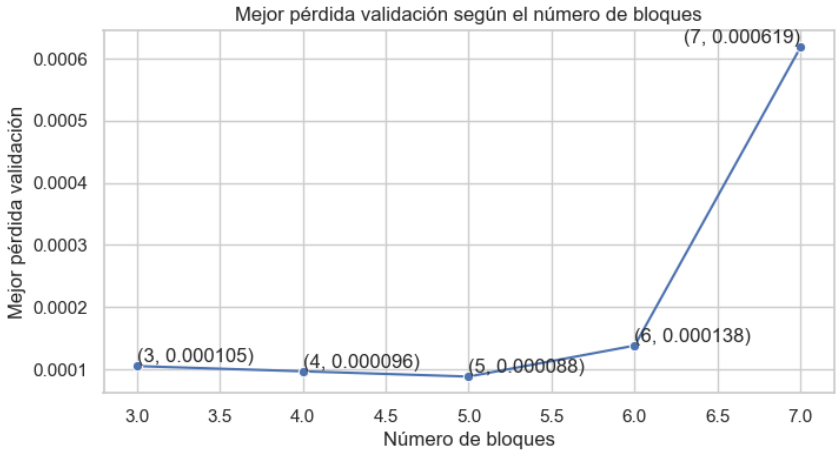

Figura 5.7: Evolución del mejor valor de la pérdida en el conjunto de validación según el número de bloques.

de neuronas en las capas densas es de 48.

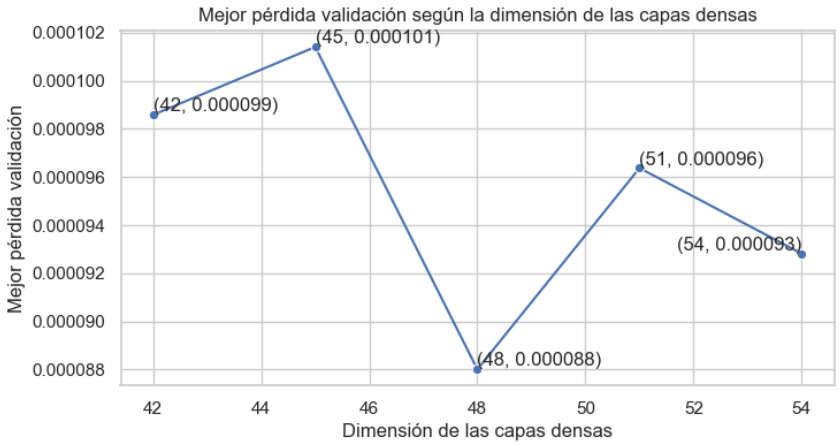

Figura 5.8: Evolución del mejor valor de la pérdida en el conjunto de validación según el tamaño de las capas densas.

5.2.3.3. Modelo final

Una vez optimizados los hiperparámetros de TSMixer, sólo resta entrenar sobre el conjunto de datos de nueve meses un modelo con esta configuración de hiperparámetros. Tras el entrenamiento, el modelo presenta una pérdida sobre el conjunto de validación de 0,0001156, ligeramente más alto que el obtenido con el modelo de un mes. Tras evaluarlo sobre el conjunto de prueba, se obtienen los resultados de la tabla 5.9. Se puede observar que el RMSE es notablemente superior al MAE, lo que significa que este modelo comete algunos errores grandes; además, el modelo predice mejor la latitud en comparación a la longitud.

En la figura 5.9 se muestran las predicciones de este modelo sobre la trayectoria desde Roma

descrita anteriormente. En ella se puede visualizar que las predicciones se corresponden en general con los vectores reales, pero al final de las secuencias la predicción se desvía (especialmente en la zona próxima al aeropuerto), lo que justifica el elevado valor del RMSE.

	RMSE	**MAE**
Latitud	0,0883	0,02710
Longitud	0,1481	0,0485
Altitud	847,4	290,9

Tabla 5.9: RMSE y MAE cometidos sobre el conjunto de prueba en las variables objetivo, por parte del mejor modelo de TSMixer. La latitud y longitud se miden en grados, y la altitud, en pies.

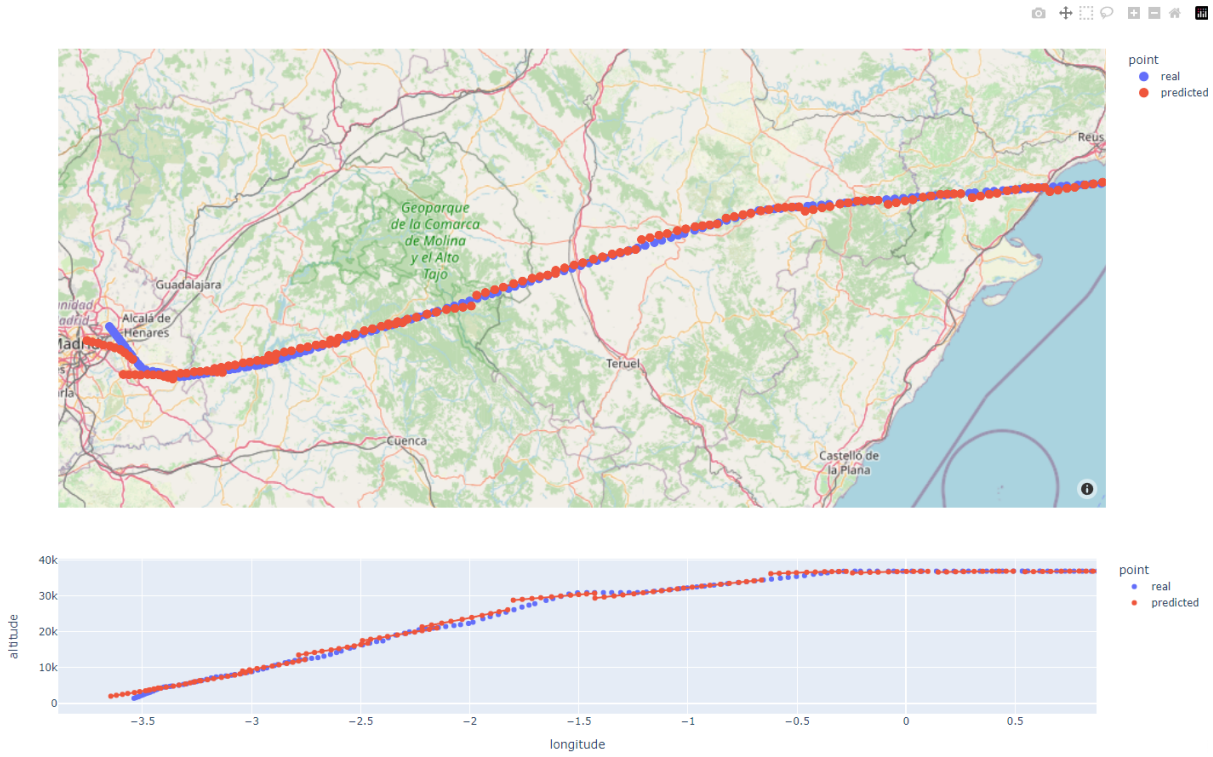

Figura 5.9: Visualización de las predicciones del mejor modelo de TSMixer.

5.3. CfC

Como se ha descrito en el apartado 4.3, esta propuesta se compone de la capa CfC con *dropout* y una capa de proyección temporal. Además, para definir las conexiones entre las neuronas de CfC se utiliza NCP, y la función de activación empleada será "lecun tanh" (ecuación 5.1), ya que es la recomendada para CfC [11]. En la tabla 5.10 se describen los hiperparámetros específicos de esta propuesta.

$$f(x) = 1{,}7159 \cdot \tanh\left(\frac{2}{3}x\right) \tag{5.1}$$

Hiperparámetro	**Descripción**
Clipnorm	CfC es sensible a la explosión del gradiente. Este hiperparámetro define un límite para la norma L2 del gradiente, lo que permite mitigar este problema.
Mixed Memory	Indica si se utilizará una celda de memoria para ayudar en el aprendizaje de dependencias a largo plazo.
Mode	La versión estándar de CfC aplica dos puertas o *gates* sobre las salidas: una sigmoide y una Hadamard. Este hiperparámetro define si se utilizarán las dos ("default"), ninguna ("pure") o sólo la sigmoide ("no_gate").
Units	Número de neuronas intermedias y de mando que comprenden la capa CfC.
Sparsity Level	Define el número de conexiones totales en la capa CfC, donde 0 implica todas las conexiones posibles, y 1 ninguna conexión (el valor máximo es 0,9).
Seed	Semilla utilizada para definir las conexiones entre las neuronas.

Tabla 5.10: Hiperparámetros específicos de la propuesta de CfC.

5.3.1. Primer experimento

De forma similar al primer experimento de TSMixer (apartado 5.2.1), se han definido hasta tres posibles valores para cada hiperparámetro, con el objetivo de estudiar su impacto a grandes rasgos. Sin embargo, se ha utilizado la Optimización Bayesiana sin Hyperband, puesto que se comprobó en el apartado 5.2.2 que esta técnica no es adecuada en las condiciones de este proyecto. En la tabla 5.11 se muestra la configuración de este experimento.

Tras entrenar 22 modelos, se ha obtenido uno cuya pérdida en la validación es del orden de $8e^{-5}$, el mejor hasta el momento. Sin embargo, en este punto se comprendió que la Optimización Bayesiana no es una técnica adecuada para comprender el impacto de los hiperparámetros sobre el modelo, pues se centra en los valores de hiperparámetros que mejor resultado sugieren. Esto significa que no se realizan entrenamientos con algunos valores de los hiperparámetros, obteniendo como resultado una visión sesgada del impacto de los mismos. Sin embargo, la capacidad de la Optimización Bayesiana para encontrar un modelo suficientemente bueno resulta muy útil de cara a obtener el mejor modelo entrenado sobre el conjunto de datos completo, puesto que se reduce considerablemente el número de entrenamientos necesarios (en este experimento se ha obtenido el mejor modelo en 22 entrenamientos, existiendo 972 combinaciones de hiperparámetros).

Igualmente, en la tabla 5.12 se muestran los cinco mejores modelos obtenidos. En un principio, se puede ver que aumentar el tamaño de los lotes empeora los resultados, pero reduce el tiempo de entrenamiento. Por otro lado, una tasa de aprendizaje menor se relaciona directamente con los mejores resultados. En general, utilizar una celda de memoria (*mixed memory*)

Hiperparámetro	Valores
Método	Optimización Bayesiana
batch_size	64, 128, 256
dropout	0, 0,1, 0,3
lookback	55
lr	0,005, 0,0005
mixed_memory	false, true
mode	default, pure, no_gate
semilla	42
sparsity_level	0,25, 0,5, 0,75
units	32, 64, 128

Tabla 5.11: Valores de hiperparámetros evaluados en el primer experimento de CfC.

resulta beneficioso, así como utilizar el modo *no gate*, aunque el mejor modelo utiliza el modo por defecto. Sobre el *dropout*, el nivel de dispersión y el número de unidades no hay una tendencia evidente, por lo que se necesitarían más pruebas.

	modelo 1	**modelo 2**	**modelo 3**	**modelo 4**	**modelo 5**
batch size	64	64	128	128	256
dropout	0	0	0,3	0,3	0
lr	0,0005	0,0005	0,0005	0,0005	0,0005
mixed memory	true	true	true	true	false
mode	default	no_gate	no_gate	no_gate	no_gate
sparsity level	0,5	0,25	0,25	0,75	0,5
units	128	64	128	128	32
mejor época	32	58	65	43	77
Duración	6h 20m	7h 5m	7h 43m	5h 3m	2h 14m
mejor val loss	**0,00008043**	**0,00008577**	**0,00009706**	**0,0001009**	**0,0001016**

Tabla 5.12: Mejores modelos del primer experimento de CfC, ordenados de mejor a peor.

5.3.2. Segundo experimento

Este experimento consistía en comprender el impacto del nivel de dispersión y el número de unidades sobre el modelo. Para ello, se estudiaron valores entre 0,15 y 0,85 para la dispersión, y entre 32 y 128 unidades. Además, se tuvo en cuenta los modos *default* y *no gate*, con el objetivo

de encontrar alguna conclusión final sobre este hiperparámetro.

Durante los entrenamientos de ambos experimentos, una gran cantidad de modelos sufrían del problema de la explosión del gradiente, obteniendo como resultado una pérdida excesivamente elevada. Cuando esto ocurría, se detenía el entrenamiento, y se mantenía el mejor punto de control hasta el momento. Con el objetivo de evitar esto, se introdujo el hiperparámetro *clipnorm*, que limita a la norma L2 del gradiente durante el ajuste de los pesos. Sin embargo, esta regularización no era capaz de evitar el problema por completo, sólo mitigarlo. Junto a esto, los resultados que se estaban obteniendo en el experimento no fueron satisfactorios, puesto que la pérdida en el conjunto de validación, en general, era del orden de $1e^{-4}$. Por estas razones, y que CfCTime muestra un comportamiento más estable, se decidió descartar esta propuesta y centrar la atención en TSMixer y CfCTime.

5.3.3. Modelo final

Tras terminar el primer experimento, se decidió entrenar un modelo con el conjunto de datos completo, utilizando la configuración de hiperparámetros del mejor modelo de dicho experimento. De esta forma se podría comprobar que los resultados de un mes de datos se corresponden con el *dataset* completo. Puesto que no se continuará profundizando en esta propuesta, se utilizará este modelo como la versión final. El resultado es que este modelo presenta una pérdida sobre el conjunto de validación de 0,0000968 y, tras evaluarlo sobre el conjunto de prueba, se obtienen los resultados de la tabla 5.13. La notable diferencia entre el RMSE y el MAE sugiere que el modelo presenta algunos errores de elevada magnitud. Junto a esto, se observa que el modelo ofrece una menor precisión al predecir la longitud frente a la latitud o la altitud.

	RMSE	**MAE**
Latitud	0,0800	0,0300
Longitud	0,1610	0,0730
Altitud	770,5	269,3

Tabla 5.13: RMSE y MAE cometidos sobre el conjunto de prueba en las variables objetivo, por parte del mejor modelo de CfC. La latitud y longitud se miden en grados, y la altitud, en pies.

En la figura 5.10 se puede visualizar las predicciones realizadas por este modelo sobre la trayectoria de Roma (apartado 5.2.3.3). En el tramo de altitud crucero el modelo es capaz de realizar predicciones precisas, pero al comenzar el despegue y realizar la primera curva las predicciones se desvían de forma evidente, ya que la secuencia apunta directamente hacia el aeropuerto. Sin embargo, según se avanza en la trayectoria las predicciones mejoran. Este comportamiento se debe al uso de ventanas de 55 vectores: al comenzar la curva, el modelo recibe principalmente vectores del tramo de altitud crucero, por lo que la secuencia resultante se ve fuertemente influenciada por estos vectores. Al avanzar en la trayectoria, se incorporan a la ventana una mayor cantidad de vectores tras la curva, por lo que el modelo actúa al respecto.

Figura 5.10: Visualización de las predicciones del mejor modelo de CfC.

5.4. CfCTime

La última propuesta de este TFM es CfCTime. Esta arquitectura, como se desarrolla en el apartado 4.3, se compone de una capa densa con *dropout*, una capa CfC con conexión residual al final, y una proyección temporal. Los hiperparámetros que definen esta arquitectura son los mismos que CfC, junto con el número de neuronas de la capa densa (*fc dim*).

5.4.1. Primer experimento

De forma similar al primer experimento de las dos propuestas anteriores, se comenzará por evaluar hasta cuatro valores de cada hiperparámetro (tabla 5.14), con el objetivo de estudiar las tendencias a grandes rasgos. Se parte con un tamaño de lote de 64, en base a los resultados de las otras arquitecturas; y el tamaño de la ventana inicialmente es de 55 vectores. El método utilizado es la Optimización Bayesiana, puesto que este experimento se llevó a cabo paralelamente al primer experimento de CfC (apartado 5.3.1); sin embargo, la conclusión es la misma: este método no es adecuado para el estudio del impacto de los hiperparámetros, pero sí que lo es para la obtención del mejor modelo.

En la tabla 5.15 se muestran los cinco mejores modelos obtenidos en este experimento. En general, estos resultados no se asemejan con los de CfC (tabla 5.12), puesto que la tendencia en CfCTime es que la celda de memoria no favorece al modelo, y un número de unidades más bajo ofrece los mejores resultados. Lo que sí muestran en común es que el mejor modo es *no gate*, puesto que en este caso todos los modelos lo utilizan. El valor de *clipnorm* más adecuado

Hiperparámetro	**Valores**
Método	Optimización Bayesiana
batch_size	64
clipnorm	1, 5, 10
dropout	0, 0,1, 0,3, 0,5
fc_dim	16, 32, 64, 128
lookback	55
lr	0,05, 0,005, 0,0005
mixed_memory	false, true
mode	default, no_gate
semilla	42
sparsity_level	uniforme (0,15, 0,85)
units	16, 32, 64, 128

Tabla 5.14: Valores de hiperparámetros evaluados en el primer experimento de CfCTime.

en general es de 5, y el *dropout*, aunque hay varios modelos que no lo utilizan (valor cero), sigue siendo relevante. Una tasa de aprendizaje baja es la mejor opción para todos los modelos, y tanto la dimensión de la capa densa como el nivel de dispersión no parecen seguir un patrón claro.

5.4.2. Tamaño de ventana

En base a los resultados obtenidos con TSMixer sobre el tamaño de la ventana (apartado 5.2.3.1), resulta interesante probar en CfCTime si un tamaño de ventana de 20 vectores mejora los resultados. El resultado es que su pérdida en el conjunto de validación es de 0,00008275, una mejora respecto a los 55 vectores, por lo que se utilizará este valor en los siguientes modelos.

5.4.3. Optimización de hiperparámetros

Puesto que los resultados del anterior experimento han resultado satisfactorios, se continuará realizando una búsqueda fina del *dropout*, la celda de memoria, el nivel de dispersión y las unidades (figura 5.16). El método utilizado esta vez es el aleatorio, que es similar al de la malla, pero escogiendo las configuraciones de forma aleatoria. Esto facilita entender las tendencias según avanza el entrenamiento.

En la tabla 5.17 se muestran los resultados de los cinco mejores modelos obtenidos. Todos estos modelos muestran una pérdida sobre el conjunto de validación similar, del orden de $8e^{-5}$, y sus hiperparámetros son similares, por lo que el margen de mejora existente al profundizar más en la optimización de hiperparámetros es mínimo.

En base a esto, se realizará el entrenamiento del modelo definido por el conjunto de hiperparámetros del modelo 1, sobre el conjunto de datos de nueves meses. Una vez finalizado el entrenamiento, el modelo presenta una pérdida sobre el conjunto de validación de 0,0001 y, tras evaluarlo sobre el conjunto de prueba, se obtienen los resultados de la tabla 5.18. Una vez más,

	modelo 1	**modelo 2**	**modelo 3**	**modelo 4**	**modelo 5**
clipnorm	5	5	5	5	1
dropout	0,1	0	0,3	0	0
fc dim	64	32	64	128	64
lr	0,0005	0,0005	0,0005	0,0005	0,0005
mixed memory	false	false	false	true	false
mode	no gate	no gate	no gate	no gate	no gate
sparsity level	0,2369	0,2895	0,8335	0,4492	0,5092
units	32	16	32	16	16
mejor época	38	67	52	27	37
mejor val loss	0,00009213	0,00009315	0,00009456	0,00009482	0,00009542
Duración	2h 31m	3h 46m 47s	3h 15m 5s	2h 39m 55s	2h 32m 41s

Tabla 5.15: Mejores modelos del primer experimento de CfCTime, ordenados de mejor a peor.

Hiperparámetro	**Valores**
Método	aleatorio
batch_size	64
clipnorm	5
dropout	0, 0,1
fc_dim	64
lookback	20
lr	0,0005
mixed_memory	*t*rue, false
mode	no_gate
semilla	42
sparsity_level	De 0,2 a 0,8, saltos de 0,05
units	10, 16, 32

Tabla 5.16: Valores de hiperparámetros evaluados en la optimización de CfCTime.

el hecho de que el RMSE sea considerablemente superior al MAE indica que el modelo presenta algunos errores de gran magnitud, y el modelo muestra una mayor precisión en la predicción de la latitud y altitud en comparación con la longitud.

	modelo 1	**modelo 2**	**modelo 3**	**modelo 4**	**modelo 5**
dropout	0,1	0,1	0	0,1	0
sparsity level	0,6	0,4	0,65	0,65	0,65
units	16	16	16	16	16
mixed memory	false	true	true	true	false
mejor época	68	56	58	45	98
mejor val loss	0,00008202	0,00008262	0,0000829	0,00008336	0,00008364
Duración	2h 15m	2h 2m	17h 35m	7h 26m	2h 45m

Tabla 5.17: Mejores modelos de la optimización de hiperparámetros de CfCTime, ordenados de mejor a peor.

	RMSE	**MAE**
Latitud	0,092	0,030
Longitud	0,151	0,066
Altitud	783,2	271,9

Tabla 5.18: RMSE y MAE cometidos sobre el conjunto de prueba en las variables objetivo, por parte del mejor modelo de CfCTime. La latitud y longitud se miden en grados, y la altitud, en pies.

En la figura 5.11 se muestran las predicciones de este modelo sobre la trayectoria desde Roma (apartado 5.2.3.3). Se puede visualizar que las secuencias siguen la tendencia de la trayectoria respecto a la longitud y altitud, pero la predicción de la latitud muestra patrones inusuales. En general, la precisión en las predicciones se reduce al estimar vectores alejados temporalmente de la ventana, al incrementarse la incertidumbre; sin embargo, las predicciones de este modelo se ajustan mejor, en general, en el centro de la secuencia, desviándose al inicio y al final. Esto se hace especialmente evidente en las últimas secuencias, donde sólo los vectores intermedios de la secuencia se corresponden con los valores reales.

5.5. Resultados

Una vez obtenidos los modelos finales de cada arquitectura propuesta, se compararán sus resultados con los del estado del arte (LSTM y TFT). En la tabla 5.19 se recogen el MAE y RMSE cometidos por cada modelo sobre las variables a predecir (latitud, longitud y altitud) en el conjunto de prueba. Como se puede comprobar, los modelos del estado del arte siguen mostrando los mejores resultados, destacando TFT, cuyos resultados en general son considerablemente mejores que los del resto de modelos.

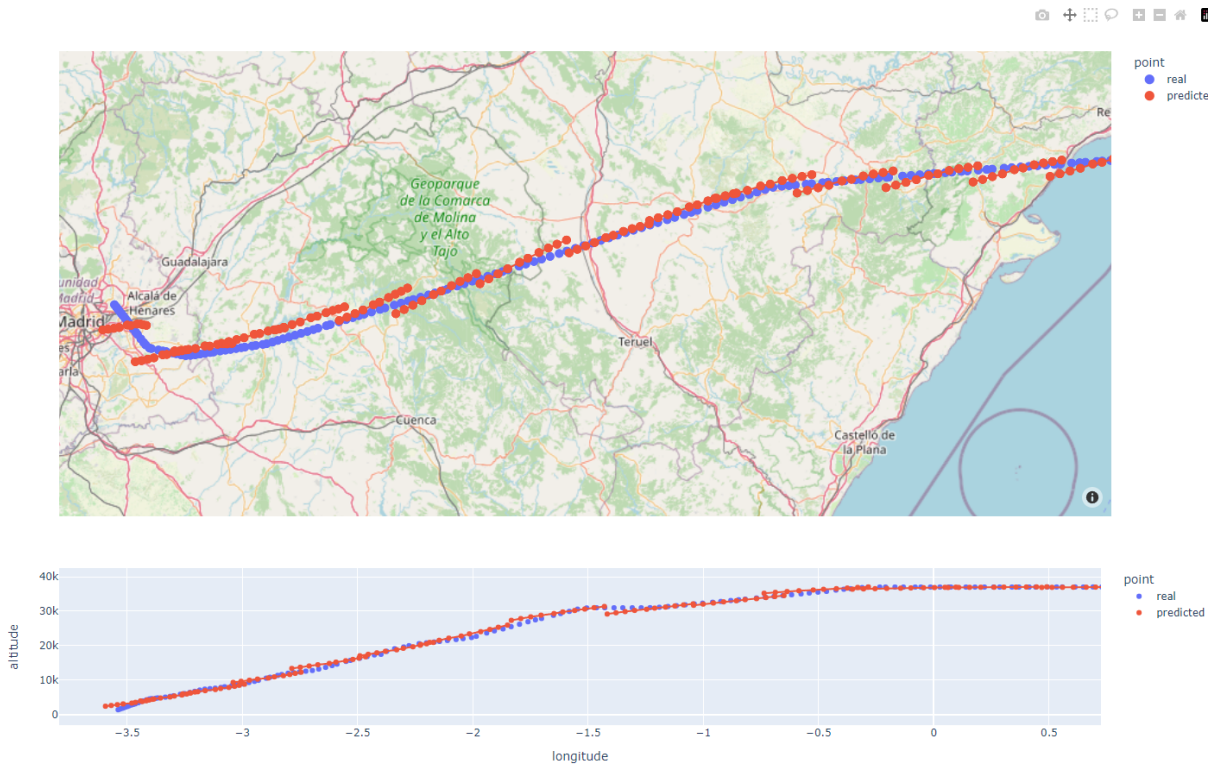

Figura 5.11: Visualización de las predicciones del mejor modelo de CfCTime.

Modelo	MAE			RMSE		
	Latitud	Longitud	Altitud	Latitud	Longitud	Altitud
LSTM	0,0216	**0,0364**	242,8	0,091	0,147	820,2
LSTM-FC	**0,0208**	0,0373	259,2	0,091	0,149	820,2
TFT	0,0216	0,0574	**189,8**	**0,0403**	**0,0834**	**465,9**
TSMixer	0,0271	0,0485	290,9	0,0883	0,1481	847,4
CfC	0,0300	0,0730	269,3	0,0800	0,1610	770,5
CfCTime	0,0300	0,0660	271,9	0,0920	0,1510	783,2

Tabla 5.19: Resultados sobre el conjunto de prueba. La latitud y la longitud se expresan en grados, y la altitud, en pies.

TSMixer, CfC y CfCTime muestran resultados competitivos, especialmente en la latitud y altitud, pero no consiguen superar a los modelos del estado del arte. Para obtener mejores resultados, sería preciso profundizar en la optimización de hiperparámetros sobre el conjunto de datos de nueve meses, con el objetivo de encontrar el mejor modelo posible. Esto no ha sido posible en este TFM debido a limitaciones de *hardware*, ya que estos entrenamientos son temporalmente costosos en las condiciones actuales, lo que impide considerar este tipo de optimización dentro

del alcance del trabajo. Como consecuencia, se han utilizado los conjuntos de hiperparámetros que han presentado los mejores resultados sobre un mes de datos.

Aunque utilizar un subconjunto representativo de los datos es útil para analizar el impacto de los hiperparámetros sobre los modelos, resulta necesario hacer una optimización sobre el conjunto de datos completo, debido a que el mejor modelo conseguido en este caso diferirá del obtenido sobre los datos de enero. Un ejemplo de esto es el *dropout* en TSMixer, donde un valor de 0,1 era el más adecuado en un mes de datos, pero sobre el conjunto de datos complejo funcionaba mejor 0,5.

Al comparar los resultados de CfC y CfCTime, se puede comprobar que son muy similares, siendo los de CfC ligeramente mejores. Esto indica que CfC sigue siendo un candidato a tener en cuenta, pese a su inestabilidad, por lo que puede resultar interesante estudiar otras técnicas de regularización diferentes o profundizar en qué valor de *clipnorm* sería capaz de evitar la explosión del gradiente, sin empeorar los resultados.

Sin embargo, al comparar las predicciones de estos modelos sobre la trayectoria desde Roma (apartado 5.2.3.3) en el visor (figuras 5.10 y 5.11), se puede comprobar que los resultados son diferentes: CfC se desvía de la trayectoria a causa de utilizar 55 vectores en la ventana, mientras que CfCTime muestra un patrón de predicciones peculiar, donde los vectores centrales de la secuencias son más precisos que los del inicio y final.

Un aspecto a destacar es que tanto TSMixer como CfCTime utilizan un tamaño de ventana de 20 vectores, frente al resto de modelos analizados, que utilizan 55 vectores (60 en el caso de TFT). Esto significa que TSMixer y CfCTime son capaces de realizar predicciones con un error similar al resto de propuestas, pero utilizando menos de la mitad de la información. Esto, a su vez, permite realizar predicciones minutos después de despegar, lo cual resulta especialmente beneficioso en trayectorias cortas.

Finalmente, se compararán las predicciones de los modelos del estado del arte a través del visor, sobre la trayectoria desde Roma. La figura 5.12 muestra las predicciones del modelo LSTM, donde se puede comprobar que las estimaciones se ajustan a la trayectoria, pero los vectores siguen un patrón errático. Como se ha indicado en el apartado 3.3, el modelo LSTM-FC incorpora dos capas LSTM adicionales con el objetivo de interpolar los vectores, y obtener una curva más suave que se asemeje a una trayectoria. Este resultado se puede observar en la figura 5.13, puesto que las secuencias predichas son idénticas a las del modelo LSTM, pero suavizadas.

Sin embargo, las predicciones de TFT no resultan satisfactorias. En la figura 5.14 se muestran las secuencias predichas por TFT, donde se puede comprobar que no se adecúan con los resultados obtenidos. Es posible que este comportamiento sea consecuencia de un fallo en la adaptación de TFT utilizando la interfaz común, pero las predicciones en altitud crucero siguen la tendencia de la trayectoria, lo cual indica que este modelo es incapaz de predecir correctamente los vectores del tramo de descenso.

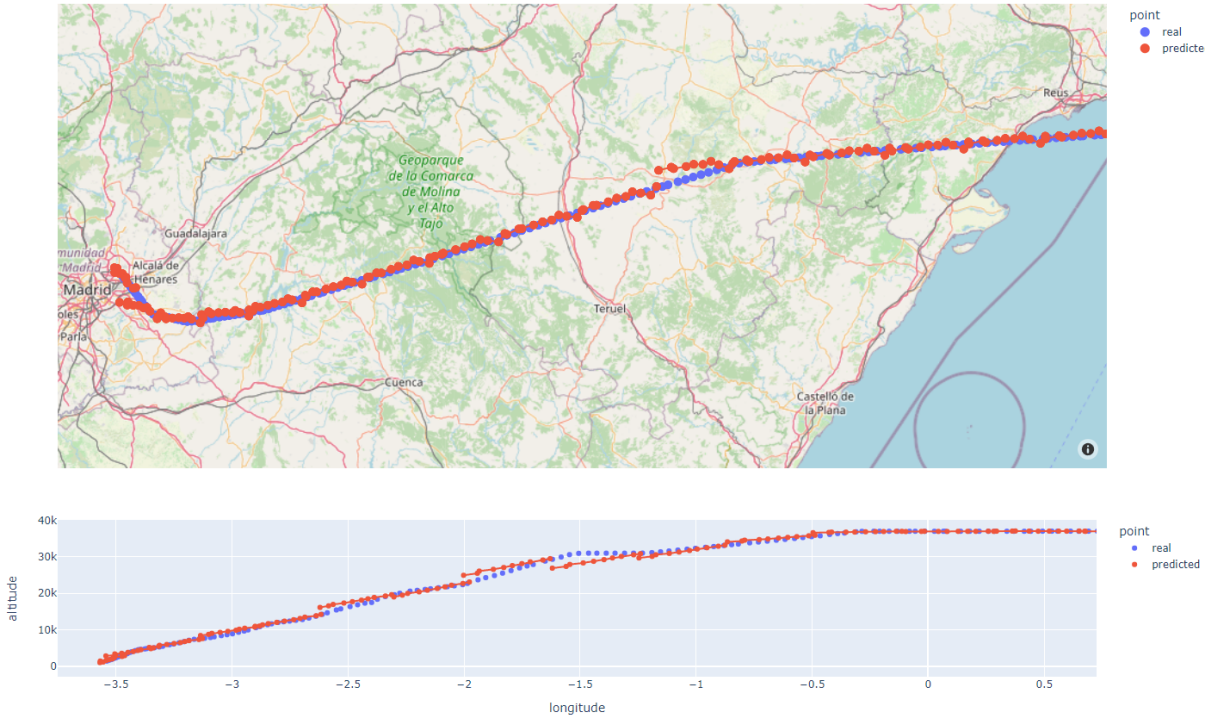

Figura 5.12: Visualización de las predicciones del modelo LSTM.

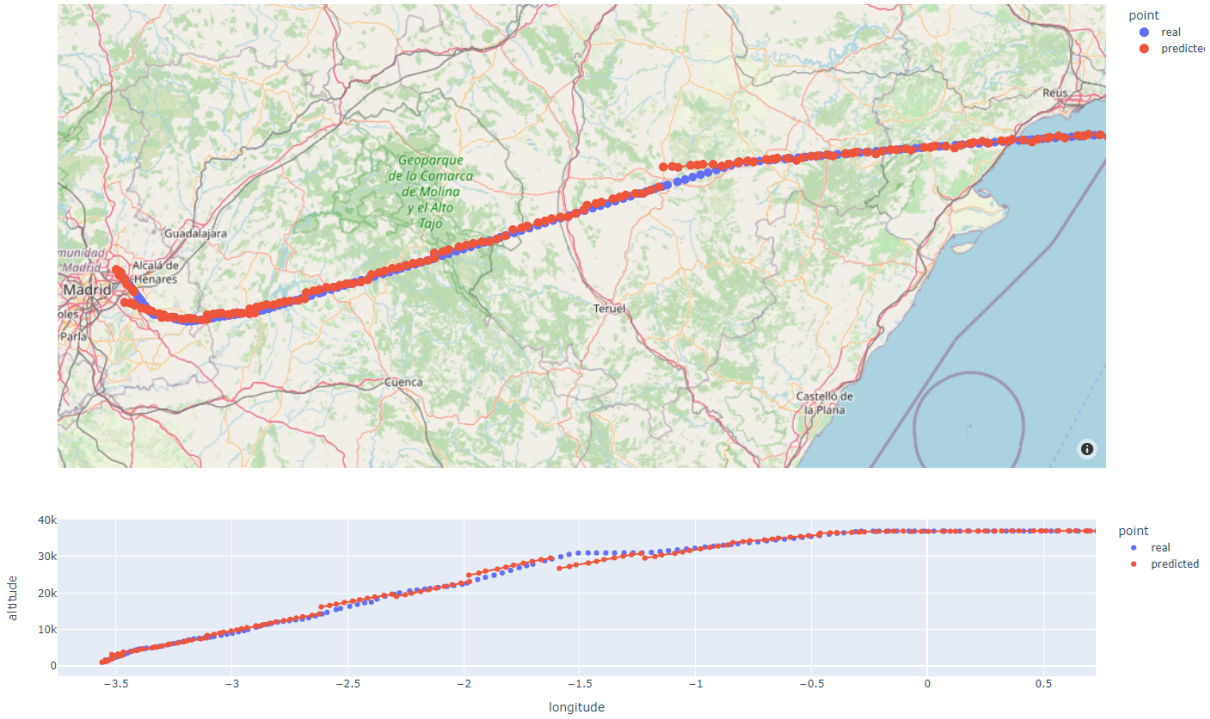

Figura 5.13: Visualización de las predicciones del modelo LSTM-FC.

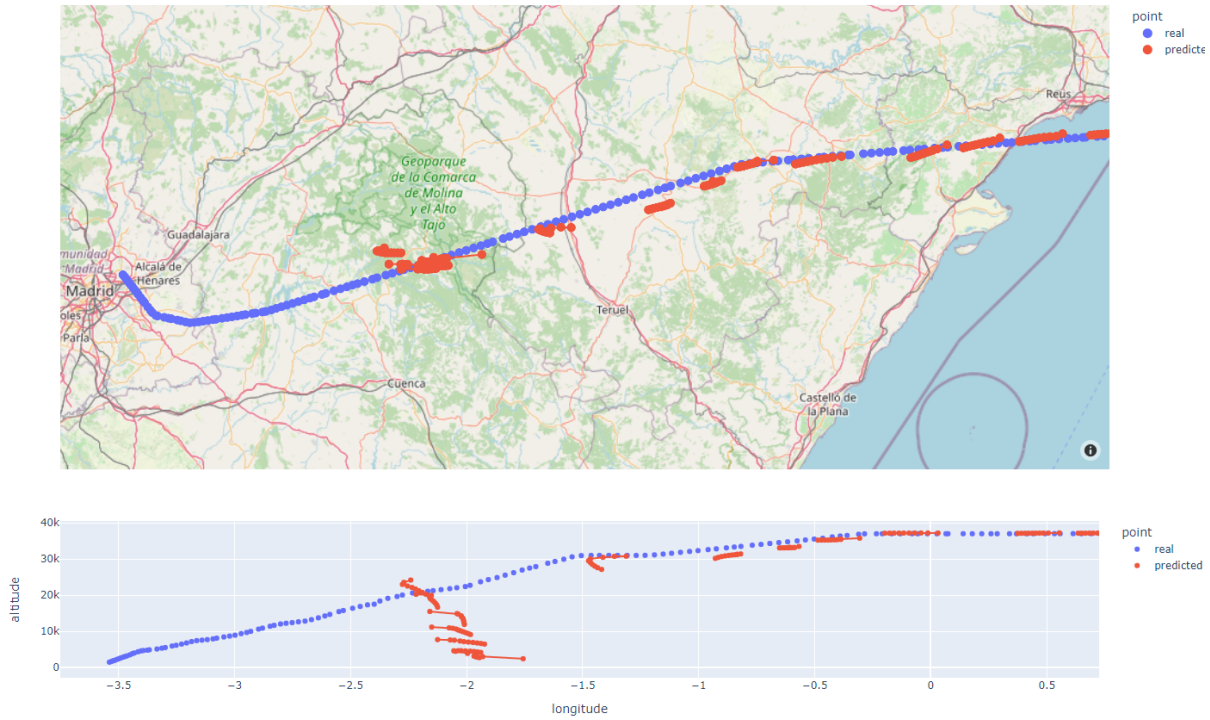

Figura 5.14: Visualización de las predicciones del modelo TFT.

Capítulo 6

Conclusiones y trabajo futuro

En este apartado se recapitulan los principales aspectos del proyecto. Se interpretan los resultados obtenidos y el grado de cumplimiento con los objetivos, estudiando las limitaciones del proyecto y proponiendo posibles mejoras. Junto a esto se incluye la perspectiva personal del estudiante que desarrolla este proyecto.

6.1. Conclusiones

A lo largo de este proyecto, se han estudiado, implementado y evaluado distintas arquitecturas de *deep learning* novedosas. TSMixer es una arquitectura ligera que se basa en concatenar capas densas, pero a pesar de su simpleza, es capaz de obtener resultados similares a otras arquitecturas más complejas.

Las LNN presentan un nuevo paradigma a la hora de realizar las conexiones entre las neuronas, modificando sus pesos en función de los valores de las entradas. Las redes CfC surgen de la necesidad de prescindir del *solver* utilizado en las redes LTC para resolver las ecuaciones diferenciales ordinarias que definen el estado interno de la red. Para ello, realizan aproximaciones de las mismas, obteniendo los mismos resultados con un error despreciable, pero mejorando el tiempo de ejecución de forma abrumadora.

Con el objetivo de aprovechar la capacidad de CfC a adaptarse a entradas que estén espaciadas en intervalos irregulares, se ha propuesto CfCTime, que recibe como entrada adicional los segundos transcurridos desde el vector anterior, además de utilizar conexiones residuales, de cara a facilitar el entrenamiento.

Comparando los resultados obtenidos, los modelos del estado del arte muestran mejores resultados que TSMixer, CfC y CfCTime. Sin embargo, las propuestas de este proyecto ofrecen resultados competitivos, por lo que es posible mejorar su capacidad predictiva al profundizar en la optimización de sus hiperparámetros

Debido a la naturaleza inestable de CfC, a causa del problema del gradiente que explota, se ha centrado la atención en CfCTime, que muestra una mayor estabilidad en los entrenamientos. Sin embargo, observando los resultados, CfC sigue siendo una propuesta competitiva, por lo que se deben estudiar otras estrategias de regulación para mitigar estabilizar al modelo, sin empeorar los resultados.

El desarrollo de la herramienta de visualización ha facilitado la comprensión del desempeño de los modelos, pudiendo distinguir las diferencias que se producen al modificar los hiperparámetros

de los modelos.

6.1.1. Perspectiva del proyecto

Este proyecto se ha llevado a cabo a lo largo de dos asignaturas: I+D+I y el TFM. En la primera etapa (I+D+I) se llevó a cabo el estudio e implementación de las propuestas, y se comenzó con el análisis del impacto de los hiperparámetros sobre los modelos.

En la segunda etapa (TFM) se ha retomado el analisis de los hiperparámetros, para a continuación realizar una optimización de los mismos utilizando los datos de un mes, debido a limitaciones de tiempo. En base a esto, se llevó a cabo los entrenamientos con el conjunto de datos de nueve meses, obteniendo así los modelos finales. Junto a esto, se ha desarrollado una herramienta para la visualización de trayectorias 4D, la cual permite visionar las predicciones que realizan los modelos. Además, esta herramienta es capaz de utilizar cualquier modelo que se haya implementado utilizando una estructura definida (establecida por una clase abstracta).

De esta forma, la primera etapa tiene un carácter más técnico, en el sentido de que se centra más en la implementación de las arquitecturas. Sin embargo, la segunda etapa se enfoca más en el análisis e interpretación de resultados, aunque ha tenido un fuerte lado técnico con la implementación del visor.

6.1.2. Perspectiva personal

Este proyecto ha resultado un reto para mí, puesto que las arquitecturas utilizadas son más complejas que las estudiadas en las asignaturas del Grado y el Máster. Sin embargo, he disfrutado mucho comprendiéndolas e implementándolas, así como procurando encontrar mejores estrategias para mejorar y reducir el tiempo de los entrenamientos, puesto que los modelos requieren en general de varias horas para entrenar, y sólo dispongo de mi ordenador personal (con el que sólo puedo entrenar cuando no lo uso) y Google Colab (que es considerablemente más lento).

El análisis e interpretación de los resultados es complejo. Una gran parte del esfuerzo en este proyecto ha consistido en entender el comportamiento de los modelos en base a sus hiperparámetros y los resultados que se obtenían de su entrenamiento, de tal forma que se pudiesen sacar conclusiones de cara a definir el siguiente experimento. El uso de Wandb ha facilitado este proceso considerablemente, al poder organizar y visualizar los entrenamientos realizados y sus resultados.

Aunque es una pena que las propuestas de este proyecto no mejoren los resultados del estado del arte, siguen siendo competitivos con los mismos. Como se ha indicado anteriormente, no se ha podido llevar a cabo una optimización de los modelos utilizando el conjunto de datos completo, ya que esto conllevaría una cantidad de tiempo abrumadora (hasta 80 horas en los modelos de mayores dimensiones), por lo que existe un margen de mejora.

La guía y apoyo de Miguel Ángel, Aníbal y Jorge ha sido fundamental para el correcto desarrollo del proyecto. Este ha sido un factor clave para que un proyecto de esta complejidad me haya resultado interesante y abordable durante todo el desarrollo del mismo. Siguiendo la metodología ASAP, hemos realizado una reunión todas las semanas, para la cual acudía al campus de Segovia y trabaja en el proyecto a lo largo de la mañana. Esta experiencia me ha resultado muy valiosa y agradable, tanto desde el punto de vista profesional como el personal.

6.2. Trabajo futuro

Una línea de trabajo directa a partir de este proyecto consiste en realizar la optimización de hiperparámetros sobre el conjunto de datos de nueve meses. Como se ha indicado en el apartado 5.5, los modelos específicos que ofrecen buenos resultados en el mes de enero no son los mismos que los del conjunto de datos completo, por lo que es necesario entrenar modelos con diferentes configuraciones de hiperparámetros. Para ello, resulta muy interesante utilizar la Optimización Bayesiana, puesto que esta técnica reduce el número de modelos a entrenar hasta conseguir uno suficientemente bueno.

Otra línea de trabajo sería implementar "TSMixer extendido" (TSMixer-Ext), una versión de TSMixer que procesa los datos dependiendo de su categoría (estáticos, variables temporales conocidas, y variables temporales desconocidas, ver apartado 3.2.2.4). Esto puede mejorar los resultados, al aprovechar mejor la información que ofrecen los datos.

Respecto a CfC, se puede valorar la implementación de otras arquitecturas que incluyan esta capa. En [11] se utiliza CfC para procesar los resultados de una red convolucional, en el procesamiento de series temporales de imágenes. Podría resultar interesante utilizar otras arquitecturas junto con CfC para la predicción de trayectorias 4D.

Finalmente, se puede utilizar los datos meteorológicos que proporciona OpenSky en sus mensajes con el objetivo de afinar las predicciones, puesto que en condiciones meteorológicas adversas pueden surgir desvíos o se deben realizar maniobras diferentes a las habituales. Sin embargo, el procesamiento de estos datos resulta complejo, y puede resultar difícil conseguir que los modelos sean capaces de extraer información valiosa de los mismos.

Bibliografía

[1] *Anaconda - The World's Most Popular Data Science Platform*. URL: https://www.anaconda.com/ (visitado 10-05-2024).

[2] *Búsqueda de empleo en Glassdoor*. Glassdoor. URL: https://www.glassdoor.es/index.htm (visitado 18-07-2024).

[3] *Calculadora coste empresa contratar un trabajador | Factorial*. URL: https://factorialhr.es/calculadora-coste-trabajador (visitado 17-07-2024).

[4] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik y Tomas Pfister. *TSMixer: An All-MLP Architecture for Time Series Forecasting*. 2023. arXiv: 2303.06053 [cs.LG].

[5] *colab.google*. colab.google. URL: https://colab.google/ (visitado 25-04-2024).

[6] RTCA Special Committee et al. *Minimum aviation system performance standards for automatic dependent surveillance broadcast (ADS-B)*. Inf. téc. Technical report, January, 1998.

[7] *Esquema de una red neuronal artificial del tipo feed forward*. ResearchGate. URL: https://www.researchgate.net/figure/Figura-2-Esquema-de-una-red-neuronal-artificial-del-tipo-feed-forward-Optimizadores_fig1_323268573 (visitado 18-07-2024).

[8] *Free Online Gantt Chart Software*. URL: https://www.onlinegantt.com (visitado 17-07-2024).

[9] *Gestiona los proyectos de tu equipo desde cualquier lugar | Trello*. URL: https://trello.com/es (visitado 10-05-2024).

[10] *Hacienda actualiza la compensación por gastos de kilometraje para quienes trabajen con vehículo propio*. URL: https://www.lamoncloa.gob.es/serviciosdeprensa/notasprensa/hacienda/Paginas/2023/170723-hacienda-actualiza-compensacion-kilometraje.aspx (visitado 17-07-2024).

[11] Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl y Daniela Rus. «Closed-form Continuous-time Neural Models». En: *Nature Machine Intelligence* 4.11 (15 de nov. de 2022), págs. 992-1003. ISSN: 2522-5839. DOI: 10.1038/s42256-022-00556-7. arXiv: 2106.13898[cs,math]. URL: http://arxiv.org/abs/2106.13898 (visitado 03-05-2024).

[12] Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus y Radu Grosu. *Liquid Time-constant Networks*. 2020. arXiv: 2006.04439 [cs.LG].

[13] Sepp Hochreiter y Jürgen Schmidhuber. «Long Short-Term Memory». En: *Neural Computation* 9.8 (1997), págs. 1735-1780. DOI: 10.1162/neco.1997.9.8.1735.

[14] *Informes anuales*. URL: https://www.aena.es/es/estadisticas/informes-anuales.html (visitado 18-07-2024).

[15] *Keras: Deep Learning for humans*. URL: https://keras.io/ (visitado 25-04-2024).

[16] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh y Ameet Talwalkar. «Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization». En: *Journal of Machine Learning Research* 18.185 (2018), págs. 1-52. URL: http://jmlr.org/papers/v18/16-558.html.

[17] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff y Tomas Pfister. «Temporal Fusion Transformers for interpretable multi-horizon time series forecasting». En: *International Journal of Forecasting* 37.4 (2021), págs. 1748-1764. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0169207021000637.

[18] Miguel A. Martínez-Prieto, Jorge Silvestre, Aníbal Bregón, Patricia Baz, Clara Gándara-González, Paula Mielgo e Irene Peñas. «Una metodología basada en prácticas ágiles para la realización de Trabajos Fin de Grado». En: *Actas de las JENUI* 8 (2023).

[19] *Microsoft Teams*. URL: https://www.microsoft.com/es-es/microsoft-teams/free (visitado 10-05-2024).

[20] Ministerio de Trabajo y Economía Social. *Resolución de 24 de enero de 2024, de la Dirección General de Trabajo, por la que se registra y publica el Convenio colectivo de Ediservicios Madrid 2000, SL*. 2 de feb. de 2024. URL: https://www.boe.es/eli/es/res/2024/01/24/(2) (visitado 17-07-2024).

[21] *Overleaf*. URL: https://es.overleaf.com (visitado 10-05-2024).

[22] *Project Jupyter*. URL: https://jupyter.org (visitado 25-04-2024).

[23] *Quickstart - Neural Circuit Policies 0.0.1 documentation*. URL: https://ncps.readthedocs.io/en/latest/quickstart.html (visitado 03-05-2024).

[24] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams y Nando de Freitas. «Taking the Human Out of the Loop: A Review of Bayesian Optimization». En: *Proceedings of the IEEE* 104.1 (2016), págs. 148-175. DOI: 10.1109/JPROC.2015.2494218.

[25] Jorge Silvestre, Paula Mielgo, Anibal Bregon, Miguel A. Martínez-Prieto y Pedro C. Álvarez Esteban. «Towards aircraft trajectory prediction using LSTM networks». En: *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*. SAC '24. Association for Computing Machinery, 2024, 1059–1060. ISBN: 9798400702433. DOI: 10.1145/3605098.3636195. URL: https://doi.org/10.1145/3605098.3636195.

[26] Jorge Silvestre, Paula Mielgo, Anibal Bregon, Miguel A. Martínez-Prieto y Pedro C. Álvarez Esteban. «Multi-route aircraft trajectory prediction using Temporal Fusion Transformers». En: *IEEE Access* (2024), págs. 1-1. DOI: 10.1109/ACCESS.2024.3415419.

[27] Souhaib Ben Taieb, Gianluca Bontempi, Amir Atiya y Antti Sorjamaa. *A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition*. 2011. arXiv: 1108.3259 [stat.ML]. URL: https://arxiv.org/abs/1108.3259.

[28] *TensorFlow*. TensorFlow. URL: https://www.tensorflow.org/?hl=es-419 (visitado 25-04-2024).

[29] *The OpenSky Network - Free ADS-B and Mode S data for Research*. URL: https://opensky-network.org/ (visitado 04-07-2024).

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser e Illia Polosukhin. «Attention is All you Need». En: *Advances in Neural Information Processing Systems*. Ed. por I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan y R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[31] *Weights & Biases: The AI Developer Platform*. Weights & Biases. URL: https://wandb.ai/site (visitado 25-04-2024).