Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Predicción de trayectorias de aeronaves empleando algoritmos de Deep Learning

---

Alumno: Irene Peñas Pérez

Tutores: Aníbal Bregón Bregón
Jorge Silvestre Vilches

Fecha: 16 de septiembre de 2022

Predicción de trayectorias de aeronaves empleando algoritmos de Deep Learning

Irene Peñas Pérez

16 de septiembre de 2022

A mis padres, mi hermano, mis abuelos,
mis tíos y mis amigos,
que me han apoyado, animado,
y ayudado en todo momento. Que han creído en mí y
a los que siempre estaré agradecida.

No te rindas, por favor no cedas,
aunque el frio queme,
aunque el miedo muerda,
aunque el sol se esconda y se calle el viento,
aun hay fuego en tu alma,
aun hay vida en tus sueños,
porque la vida es tuya y tuyo tambien el deseo,
porque lo has querido y porque te quiero.

"No te rindas", Mario Benedetti

Usa tu sonrisa para cambiar el mundo,
no dejes que el mundo cambie tu sonrisa.

La magie consiste à rêver et à croire
que rien n´est impossible.

Tu mettici il tuo meglio,
e allora darai luce anche a tutto il resto.

Working hard is important.
But there's something that matters even more.
Believing in yourself.

"Harry Potter", J.K Rowling

"Gib niemals auf.

Agradecimientos

Querría agradecer en primer lugar a mis tutores, Aníbal Bregón Bregón, Jorge Silvestre Vilches y al profesor Miguel Ángel Martínez Prieto por la oportunidad de poder hacer este TFG con ellos, y por su incansable ayuda.

También me gustaría agradecer a mis padres, Mariano y Alicia, a mi hermano Mariano, a mis abuelos, a mis tíos y a mis amigos, porque han estado a mi lado en todo momento, apoyándome y animándome siempre, porque siempre han pensado que sería capaz de lo que me propusiera, incluso las veces que yo lo veía más complicado.

Sin todos ellos no habría sido posible llegar hasta donde he llegado, muchas gracias de corazón.

Resumen

A día de hoy, el avión es el medio de transporte más utilizado del mundo. Miles de aviones sobrevuelan cada día los cielos de todo el mundo. Es por esto, que pueden surgir conflictos entre las diferentes aeronaves, cuando están en sus respectivas rutas hacia sus aeropuertos de destino.

Sin embargo, es posible predecir y anticipar estos posibles problemas, gracias a la aplicación de algoritmos de Aprendizaje Profundo. Los datos que se han tomado para realizar el estudio pertenecen a un cuadrante hecho alrededor del Aeropuerto Adolfo Suárez Madrid-Barajas. El conjunto de datos pertenece a un total de 100 vuelos, en un periodo de tiempo de 10 días, del 1 de enero de 2020 al 10 de enero de 2020. Mediante dicha serie de datos de posición (latitud, longitud, y tiempo), que son recogidos mediante el sistema ADS-B (Automatic Dependent Surveillance-Broadcast), es posible crear unas matrices en las cuales se representa la trayectoria de la aeronave.

Después, se puede entrenar un modelo, que estará formado por Redes Neuronales Convolucionales y LSTM (Long Short-Term Memory), el cual será capaz de predecir la próxima posición de la aeronave. En consecuencia, se podrán evitar posibles conflictos entre aeronaves, mejorando la seguridad de los aviones, además de mejorar los tiempos de llegada, ya que los aviones no tendrían cambios significativos en sus respectivos rumbos. Por otro lado se ahorraría combustible, y a su vez, sería mejor para el medio ambiente. Después de haber entrenado el modelo, se comprueba que éste predice satisfactoriamente las trayectorias de las aeronaves.

Por otro lado, la metodología que se ha utilizado para llevar a cabo el proyecto ha sido UVagile, una metodología ágil, desarrollada para el ámbito académico. El proyecto se ha dividido en 5 sprints de aproximadamente la misma duración cada uno. Al final de cada sprint se ha entregado un incremento. Al final del último sprint, se ha entregado el incremento final.

Palabras clave: predicción de trayectorias, gestión del tráfico aéreo, aprendizaje profundo, redes neuronales convolucionales, LSTM, UVagile.

Abstract

Nowadays, the aeroplane is the most widely used means of transport in the world. Thousands of aircraft fly over the skies around the world every day. This is why conflicts can arise between different aircraft when they are on their respective routes to their destination airports.

However, it is possible to predict and anticipate these potential problems, thanks to the application of Deep Learning algorithms. The data taken to carry out the study belong to a quadrant made around the Adolfo Suárez Madrid-Barajas Airport. The dataset belongs to a total of 100 flights, in a time period of 10 days, from 1st January 2020 to 10th January 2020. Using this series of position data (latitude, longitude and time), which are collected using the ADS-B (Automatic Dependent Surveillance-Broadcast) system, it is possible to create matrix in which the aircraft trajectory is represented.

Afterwards, a model can be trained, which will consist of Convolutional Neural Networks and LSTM (Long Short-Term Memory), which will be able to predict the next position of the aircraft. As a result, possible conflicts between aircraft can be avoided, improving aircraft safety, as well as improving arrival times, since aeroplanes would not have significant changes in their respective courses. On the other hand, it would save fuel, which would be better for the environment. After training the model, it is checked that aircraft trajectories match satisfactorily the model's prediction.

On the other hand, the methodology used to carry out the project was UVagile, an agile methodology developed for the academic environment. The project was divided into 5 sprints of approximately the same duration each. At the end of each sprint an increment was delivered. At the end of the last sprint, the entire increment was delivered.

Keywords: trajectory prediction, air traffic management, deep learning, convolutional neural networks, LSTM, UVagile.

Índice general

Índice de figuras

Índice de tablas

Parte I

Descripción del proyecto

Capítulo 1

Introducción

La aeronáutica es una disciplina que se dedica al estudio, diseño y manufactura de aparatos mecánicos capaces de volar, además del conjunto de técnicas que permiten el control de aeronaves [77]. Por otro lado, la gestión del tráfico aéreo (*Air Traffic Management*, ATM) integra la interacción entre todos los sistemas que ayudan a las aeronaves a partir de un aeropuerto de origen, incorporarse a un determinado espacio aéreo de tránsito, y la toma de tierra en un aeropuerto de destino. Esto incluye el control del tránsito aéreo, el personal encargado de la seguridad durante el mismo, la meteorología aeronáutica, las ayudas a la navegación, etc. En la Figura 1.1, se pueden observar todas las etapas dentro de ATM [9].

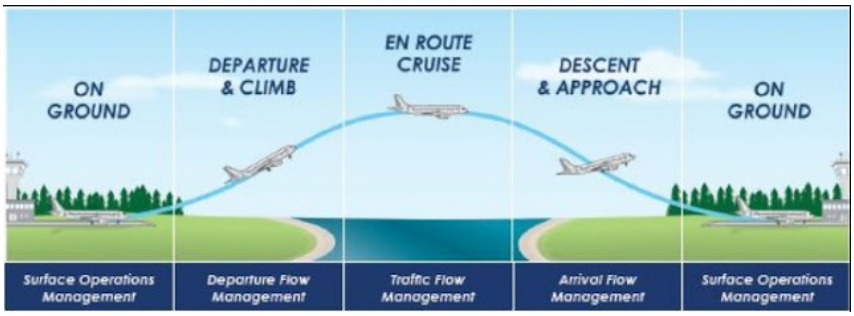


Figura 1.1: Gestión del tráfico aéreo

En el sector de la gestión del tráfico aéreo se producen todos los días una gran cantidad de vuelos de aeronaves. La ruta que la aeronave realiza desde un lugar de origen para llegar a uno de destino, se conoce como trayectoria de vuelo. Estas trayectorias, a veces pueden causar conflictos entre aeronaves, haciendo que una de las aeronaves tenga que desviarse de forma no planificada. Sin embargo, este inconveniente podría ser salvado prediciendo las trayectorias.

El término predecir se define como "anunciar por revelación, conocimiento fundado, intuición o conjetura algo que ha de suceder", según la Real Academia Española [21]. Por lo tanto, la predicción de trayectorias podría definirse como la tarea de anticipar qué rumbo van a tomar las aeronaves, evitando así posibles dificultades antes de que se pudieran producir. En circunstancias normales, los vuelos no pueden sufrir ningún conflicto, debido a que estos son planificados con hasta una semana de antelación, y muchos son periódicos y por tanto repetitivos. Los conflictos surgen cuando se dan circunstancias anómalas, que llevan a una situación no planificada. Algunos de estos problemas son: fenómenos atmosféricos adversos, retrasos en la salida del aero-

puerto de origen, o también exceso de tráfico en el destino. Pueden ocurrir otras circunstancias excepcionales como la guerra de Ucrania, por la cual ha habido que reordenar muchas rutas, o también desviaciones por parte del piloto con respecto a la ruta planificada. Los conflictos suelen producirse en el entorno de los aeropuertos, pues son las zonas de mayor densidad de operaciones de aeronaves. A cielo abierto la probabilidad de que exista un conflicto es baja, debido al espacio que hay. Por consiguiente, en el presente Trabajo Fin de Grado, se abordará el desafío de la predicción de trayectorias en el entorno del aeropuerto, dada su relevancia en el marco general de la gestión del tráfico aéreo.

Para poner en contexto el tema sobre el que se cimenta el Trabajo Fin de Grado, hay que remontarse un siglo atrás. La historia de la aviación comercial comienza su andadura a principios de siglo XX, en 1910, unos años después de que fueran creados los primeros aviones, en 1903, por los hermanos Wright (Figura 1.2).

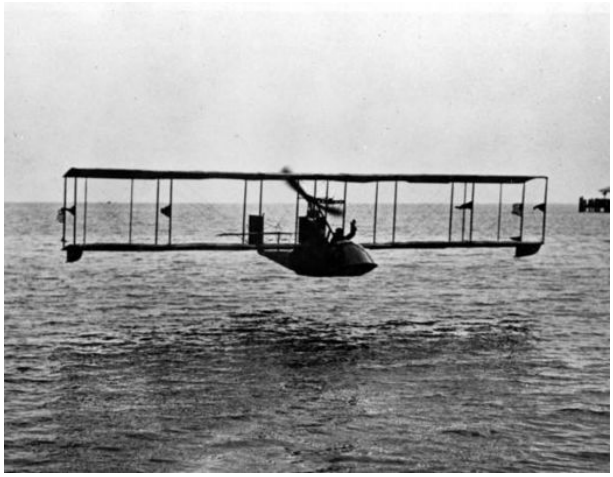


Figura 1.2: Biplano

En 1919, tras finalizar la Primera Guerra Mundial, nacieron las primeras compañías aéreas. Inicialmente, la navegación dependía exclusivamente del piloto, sin embargo, un tiempo más tarde, en el año 1935, se inventó el radar (Figura 1.3), un sistema de ondas electromagnéticas para medir distancias, altitudes, direcciones y velocidades de objetos, como por ejemplo aeronaves.

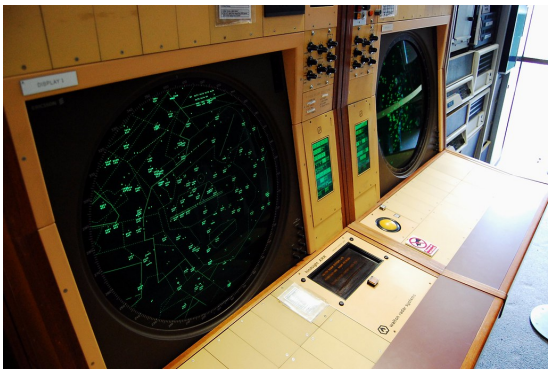


Figura 1.3: Radar

Esta tecnología se siguió desarrollando, sobre todo en Inglaterra durante la Segunda Guerra Mundial. Con la explosión del conflicto bélico, la aviación desempeñó un papel sumamente importante. De esta manera, todo el sector aeronáutico, tanto civil como militar, fue prosperando. Años más tarde, en el sector se empezaron a utilizar motores a reacción, lo que supuso un gran avance en cuanto a tiempos de viaje. Un hecho que revolucionó la aviación, fueron los ordenadores y la informática en todas sus facetas, tanto en sistemas de ciberseguridad para prevenir ataques informáticos, como para ver la posición de una aeronave, o para predecir la trayectoria que un avión puede tomar en un tiempo determinado. Con el paso del tiempo, la aviación comercial fue evolucionando, hasta llegar a nuestros días. A día de hoy, el avión es uno de los medios de transporte más utilizados de todo el mundo, ya que permite salvar distancias entre ciudades en un tiempo muy asequible. En la Figura 1.4. se muestra una gráfica con el número de pasajeros que utilizaron el avión en la última década.

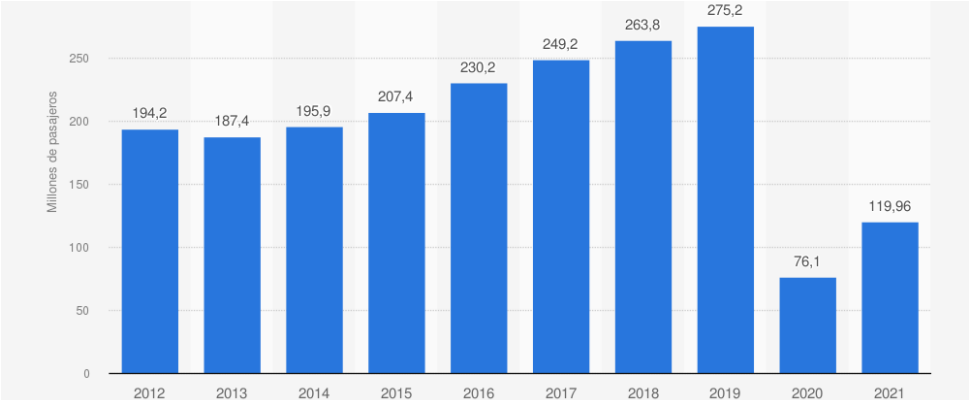


Figura 1.4: Pasajeros en transporte aéreo en España entre 2012 y 2021 (millones) [53]

Se puede ver que el número de usuarios del avión ha ido creciendo a lo largo de los años, exceptuando en estos dos últimos años, 2020 y 2021, debido a la pandemia del COVID-19, en la que el uso del avión ha bajado radicalmente debido a las restricciones a la movilidad aplicadas sobre los desplazamientos internacionales.

Paralelamente, una de las ramas de la informática que más ha crecido es la Inteligencia Artificial (*Artificial Intelligence*, AI), y en particular el Aprendizaje Automático (*Machine Learning*, ML). Dentro del ML, se encuentra el Aprendizaje Profundo (*Deep Learning*, DL) que se muestra muy potente combinado con el *Big Data*, debido a que uno de sus requisitos es disponer de grandes cantidades de datos para que los algoritmos obtengan buenos resultados. En el DL sobresalen 2 técnicas: las Redes Neuronales Convolucionales (*Convolutional Neural Networks*, CNN) y las Redes Neuronales Recurrentes (*Recurrent Neural Networks*, RNN). Las CNN son muy útiles a la hora de encontrar patrones en imágenes, para reconocer objetos, caras y escenas [62]. Por otro lado, las RNN se utilizan para analizar datos de series temporales [72]. Este tipo de redes profundas se pueden aplicar al problema de la predicción de trayectorias. Existen investigaciones en otros campos de estudio, en las que se realizan predicciones basadas en DL mediante fotogramas de vídeo. Se puede observar una cierta similitud entre la predicción de trayectorias con imágenes y la de fotogramas de vídeo. Si se representa la trayectoria de una o varias aeronaves en el espacio aéreo en forma de imagen, y se ve su evolución a lo largo del tiempo (mediante múltiples fotogramas), se podrían utilizar técnicas de DL para predecir imágenes, o fotogramas que describan la posición más probable de dichas aeronaves en el futuro.

1.1. Planteamiento del problema (*Problem Statement*)

- IDEAL: El entorno ideal es un entorno operativo en el que cualquier tipo de desviación con respecto al plan (la planificación inicial no debería incluir posibilidades de conflicto) pueda detectarse y corregirse con suficiente antelación como para que no se pueda producir ninguna situación de riesgo para la seguridad o una divergencia significativa con respecto a lo planificado.
- REALIDAD: Los sistemas que se usan son radares, uno primario y uno secundario. Estos radares tienen una serie de limitaciones, por lo que se están implantando otras tecnologías más modernas. Actualmente, las trayectorias de los aviones pueden dar lugar a conflictos, que pueden llevar a una de las dos aeronaves a cambiar su trayectoria planificada. Si ocurre esto, seguramente haya cambios en la separación horizontal, en la altitud o en los tiempos con respecto a la planificación.
- CONSECUENCIAS: Si este problema no se soluciona, supondrá una serie de pérdidas, ya que al tener que cambiar de trayectoria una de las dos aeronaves, esto interfiere o bien en el tiempo de vuelo, o en la altitud (separación vertical) o en la separación horizontal (diferencia de las posiciones de las dos aeronaves en el plano horizontal, volando al mismo nivel). Al aumentar el tiempo de vuelo, se pueden sufrir pérdidas económicas, más gasto de combustible y una posible reducción de la seguridad
- PROPUESTA: La solución a la que se quiere llegar, es poder predecir las trayectorias de los aviones, de tal manera que se puedan prever con antelación los conflictos y se puedan resolver con cambios menos drásticos. Para poder predecir las citadas trayectorias, se usará información de posición de las aeronaves, obtenida de un *dataset* compuesto de mensajes ADS-B. Mediante la aplicación de algoritmos de Aprendizaje Automático, se podrán anticipar dichos rumbos. De esta manera, los tiempos de llegada de los aviones serán mejores, se ahorrará en combustible al no tener que cambiar trayectorias, y los aviones serán más seguros al predecir dónde van a estar los aviones en cada momento, evitando así en gran medida conflictos que puedan surgir entre las diferentes aeronaves.

1.2. Objetivos del trabajo

- OBJ-1: Se estudiará el entorno de negocio relacionado con la gestión del tráfico aéreo (ATM), analizando un aeropuerto concreto, el Aeropuerto Adolfo Suárez Madrid-Barajas. A su vez se estudiarán las tecnologías que existen actualmente en relación con ATM.
- OBJ-2: Se hará un análisis de las características y el funcionamiento del ML y DL, además de ver los algoritmos principales dentro de cada una de las ramas.
- OBJ-3: Mediante una serie de datos, utilizando algoritmos de Aprendizaje profundo (DL), se creará un modelo para predecir las trayectorias de los vuelos.

1.2.1. Restricciones

- Disponer de suficientes datos para poder entrenar de una forma adecuada a la red neuronal. El *dataset* utilizado incluye datos ADS-B procedentes en su origen de *OpenSky*, procesados

y enriquecidos de tal manera que facilite su utilización [51].

- Tiempo disponible para realizar el TFG (300 horas).

1.3. Estructura de la memoria

En esta sección se va a detallar cómo se ha organizado el documento, para que su posterior lectura sea más fácil.

- **Capítulo 1. Introducción.** En el primer capítulo se describe la motivación y los objetivos del TFG, además de la estructura del documento.
- **Capítulo 2. Planificación.** En el siguiente capítulo se detalla qué metodología se ha usado para realizar el TFG.
- **Capítulo 3. *Background*.** En este capítulo se explica el contexto del tema principal del presente TFG, es decir, todo aquello que rodea a la predicción de trayectorias en el mundo de la aeronáutica comercial.
- **Capítulo 4. Propuesta.** En este capítulo se realiza la propuesta para la predicción de las trayectorias de los aviones, desde el tratamiento de datos para convertirlos en matrices, hasta la final predicción del modelo ya entrenado.
- **Capítulo 5. Evaluación.** En este capítulo se va a estudiar cómo de bien funcionan los diferentes modelos, haciendo una serie de comparaciones entre ellos.
- **Capítulo 6. Conclusiones.** En este capítulo se expondrán los resultados de todo el proyecto, analizando los aspectos más importantes y haciendo una recapitulación del TFG.

Capítulo 2

Planificación

En este capítulo se presenta la metodología utilizada para llevar a cabo el TFG, además de las herramientas utilizadas, la planificación temporal, los presupuestos y el balance temporal y económico. El proyecto se ha realizado siguiendo la metodología ágil UVagile, descrita en la Sección 2.1. El proyecto se ha dividido en 5 sprints de aproximadamente la misma duración cada uno. La organización de cada sprint ha sido descrita en el *backlog* de la aplicación Trello.

2.1. Metodología de trabajo

El presente Trabajo Final de Grado se va a estructurar y desarrollar empleando la metodología de trabajo ágil UVagile [44], adaptada para su uso en Trabajos de Fin de Grado. Esta metodología ágil se basa en los valores y principios del manifiesto ágil, mejorando los proyectos de aprendizaje. El proyecto se desarrollará atendiendo a todas las normas establecidas por la metodología, para lograr un trabajo eficiente y con unos resultados mejores. A continuación se pasa a explicar qué es UVagile.

2.1.1. El marco de trabajo UVagile

UVagile es una metodología de enseñanza y aprendizaje desarrollada como un proyecto de innovación docente de la Universidad de Valladolid. Se quiere implantar una metodología Agile, en concreto Scrum [61] en el ámbito de la universidad. Mediante la implementación de UVagile se pretenden abordar proyectos de aprendizaje y establecer una dinámica de trabajo iterativa e incremental, al tiempo que se motivan el aprendizaje activo, la comunicación entre el profesor y los alumnos, y generar de manera más temprana la retroalimentación.

En este trabajo se usarán los siguientes artefactos:

- Espacio de trabajo compartido en Teams: aquí se guardarán todos los subproductos que se vayan generando durante los sprints. Este espacio será visible tanto para el alumno como para los profesores. Contará también con 2 canales: un canal privado para que la comunicación sea más sencilla entre los profesores y el alumno; y un segundo canal general con el resto de la comunidad. Dicho espacio cuenta con control de versiones y una copia de respaldo en la nube. El espacio de trabajo compartido está formado por un conjunto de directorios: documentación, incrementos, personal y recursos.

- Tablero del proyecto en Trello: el tablero de Trello cuenta con 6 columnas cada una dedicada a un propósito en concreto: *Backlog* del proyecto, Objetivo del sprint, Tareas pendientes, En curso, Bloqueado y Finalizado. En el tablero se ponen las historias de usuario con los objetivos del proyecto. Cada historia corresponde a una serie de tareas que se van a ir realizando durante la consecución del sprint. Las tareas en un principio están en pendientes. Una vez empezadas se mueven a en curso. Si surgen bloqueos, se moverán a dicha columna, En caso de que la tarea se complete, se llevará la ficha a finalizada. En el tablero se especifican los objetivos mediante dichas historias de proyecto, con una descripción de lo que se debe realizar. Las tareas cuentan con una descripción también, con unos puntos de esfuerzo para realizarla, la fecha de finalización prevista y una lista con todos los resultados que tienen que darse para que dicha tarea se marque como completada. Este tablero es accesible tanto al alumno como a los profesores, para que se pueda ir siguiendo el desarrollo del proyecto de manera ordenada y eficaz.

- Incremento: son todos los entregables que se crean durante el periodo de tiempo que dura un sprint y que debe satisfacer el alcance establecido para el sprint. En un incremento se puede entregar código, la memoria, el cuaderno de trabajo, etc.

- Cuaderno de trabajo: artefacto utilizado para anotar y llevar un control sobre las tareas que se llevan a cabo, el tiempo que estas han requerido y la fecha en la que se completaron

- Retroalimentación: al final de cada sprint, al entregar el incremento, los profesores realizan una corrección, y con ello se produce una retroalimentación, es decir, se ponen de manifiesto todas aquellos aspectos que se deben perfeccionar y se aportan ideas para mejorar los distintos aspectos del TFG. Dicha retroalimentación se recibe tanto de profesores como de la comunidad tras la presentación del trabajo, que se realiza al final del sprint.

- Eventos:

  - Sprint de trabajo: periodo de tiempo definido, en el que se establecen una serie de objetivos, que al final del sprint tienen que haber sido realizados. Todos los sprints tienen que tener la misma duración (de 4 a 5 semanas). La carga de trabajo oscilará entre 60 y 75 horas. De esta manera, se proporciona retroalimentación de una manera rápida y frecuente. En cada sprint se tienen que realizar una serie de ceremonias, además de la entrega de un incremento del producto, mediante el cual se materializan los objetivos del sprint.
  - Reunión de inicio: es el punto de partida de cada sprint. Se tiene como propósito consolidar el objetivo del sprint y establecer una cierta planificación. En la reunión de inicio el equipo analiza el alcance y se derivan las tareas. A su vez se fijan las fechas de las reuniones de sincronización. En esta reunión, participan los profesores y el alumno. La reunión se realiza la primera semana del sprint, con una duración de 30 minutos.
  - Reunión de sincronización: reunión semanal que tiene una duración máxima de 15 minutos, para asegurar una comunicación regular y frecuente entre el alumno y los profesores. En esta reunión se tiene que responder a 3 preguntas: ¿qué hice ayer?, ¿qué haré hoy?, ¿en qué me he atascado? En esta reunión participan los profesores y el alumno.

  - Comunicación de progresos: el mayor objetivo es retroalimentar el desarrollo del proyecto y preparar la posterior defensa, en base al incremento que se ha entregado. La presentación es una exposición a alto nivel del proyecto, junto a su planificación, la solución dada, los resultados conseguidos, y un análisis del desempeño durante la duración del sprint. El contenido de la presentación va evolucionando a medida que van realizándose los sprints. En esta reunión, están presentes los profesores, el alumno y la comunidad. Dicha reunión se realiza el último día del sprint, con una duración de 1 hora por alumno (30 minutos de presentación y otros 30 minutos para el debate posterior).
  - Retrospectiva: en esta reunión se revisa el proceso de trabajo que se ha realizado durante el sprint, destacando los aspectos positivos y las mejoras a hacer. La retrospectiva se lleva a cabo en un tablero donde se ubicarán los aspectos positivos, negativos y a mejorar. La herramienta utilizada para la retrospectiva es EasyRetro. En esta reunión los participantes son el alumno, los profesores y la comunidad. La retrospectiva es la última actividad del sprint, con una duración de 30 minutos o menos.

- Roles: papeles que desempeñan las diferentes personas del equipo.

  - Profesor/es: son los tutores. Son los responsables de plantear un proyecto de calidad, para que se alcancen los objetivos deseados en la guía docente del TFG. Además, guiarán al alumno a lo largo del proyecto. El papel de profesor se inspira en el de *Product Owner* de Scrum, aunque también tiene pinceladas de *Scrum Master*. Los profesores tienen una serie de responsabilidades: desarrollarán y comunicarán el objetivo del sprint, detallando las historias de proyecto en el tablero de aprendizaje; facilitarán que se desarrollen los eventos en tiempo y forma; participarán de todos los eventos planificados para ese sprint; ayudarán al alumno a resolver los bloqueos y fomentarán la comunicación a través de los canales previamente acordados; por último, proporcionarán retroalimentación de valor al alumno regularmente y en plazos de tiempo suficientemente cortos.
  - Estudiante: será el alumno. Es el papel principal, debido a que él tiene que construir el producto esperado en el TFG y defenderlo posteriormente ante un tribunal. Este rol está inspirado en el rol *developer* (desarrollador) de Scrum. El estudiante debe establecer un plan de trabajo, es decir, las tareas a realizar, a partir del objetivo del sprint. Dichas tareas deberán ser ejecutadas en tiempo y forma. Además, éste deberá participar en todos los eventos planificados para el sprint. El alumno mantendrá una comunicación fluida con los profesores, actualizará el tablero del proyecto y completará el cuaderno de aprendizaje.
  - Comunidad: responsable de establecer un entorno de trabajo de aprendizaje colectivo. Dicha comunidad acompañará al alumno en todo su camino a través del proyecto. La comunidad está formada por el "grupo de alumnos" que realiza el TFG al mismo tiempo que el alumno, además de otras personas interesadas, tanto profesores como alumnos. La comunidad debe participar de manera activa en la Comunicación de progresos, proporcionando una retroalimentación constructiva. Además, deben participar en la retrospectiva y proponer acciones de mejora del proceso de trabajo. Por último deben colaborar con el alumno en sus bloqueos y fomentar la comunicación a través de los canales establecidos.

- Tribunal: responsable de valorar el trabajo realizado por el alumno, en concordancia con los objetivos de aprendizaje que se esperan para el TFG. Dicho tribunal está formado por un grupo de profesores de la titulación. Dentro de este grupo, no puede haer ningún profesor que haya desempeñado el rol de profesor en el TFG. El tribunal está inspirado en el papel del *cliente*. El tribunal asume una serie de responsabilidades: evaluar objetivamente y atendiendo a los objetivos de aprendizaje del TFG, el producto de aprendizaje realizado por el alumno; evaluar objetivamente la calidad de la documentación técnica generada, que forma parte del producto construido por el alumno; evaluar objetivamente la calidad de la defensa del TFG realizada por el alumno.

2.2. Estimación del esfuerzo

A la hora de realizar la estimación del esfuerzo que se requiere para realizar cada una de las tareas planificadas, se cuenta con varias técnicas: tallas de camisetas y *planning poker* entre otras. Se ha escogido la metodología de *planning poker* porque es bastante simple y efectiva para establecer los puntos de historia de cada tarea. A continuación se explica su funcionamiento:

- Se tiene una baraja de cartas, cuyos valores siguen la secuencia de Fibonacci: 1,2,3,5,8,13 y 21. En cada baraja, además se cuenta con 3 cartas especiales: una con el símbolo de ∞, otra con el de ? y una última con el del café. Las cartas que componen el mazo están repetidas un número determinado de veces, para que puedan usar una baraja todos los miembros del equipo. Cada miembro coge una carta de cada valor, y ya están listos para empezar a planificar.

Figura 2.1: Baraja de cartas de planning poker

- Los valores de cada carta en este caso van a ser puntos de historia, es decir, la dificultad que va a entrañar realizar cada tarea. Los valores van desde el 0, sin dificultad, hasta el 21, la dificultad más alta. En el caso de ∞, quiere decir que hay que subdividir la tarea, porque si no, es imposible de realizar. En caso de ?, quiere decir que se desconoce la dificultad, y el símbolo del café se utiliza para realizar un descanso cuando se está planificando.

- Por último, las tareas se van leyendo en voz alta, y los miembros del equipo, eligen una carta para dicha tarea. A la de 3, los miembros del equipo revelan la carta que han escogido. Si coincide, se incluyen los puntos de historia en el tablero del proyecto, dentro de las descripciones de las tareas. Si las personas no están de acuerdo, habrá que debatir por qué cada persona ha asignado esos puntos. Después se tiene que repetir la estimación, y se repetirán estos 2 últimos pasos hasta que se alcance un consenso.

En este caso, al ser una sola persona, se estimará en función de las consideraciones que la misma haga. Cada tarea tendrá en Trello unos puntos de historia asignados. En las Tablas 2.1 a 2.5 se expone cada una de las tareas y subtareas que se han planificado:

Sprint 1	
Tarea	Subtarea
Caracterización del proyecto	Definir los objetivos y las tareas a realizar en el sprint Reunión de inicio Reunión de sincronización Descripción de la metodología de trabajo Establecer una metodología de trabajo, una planificación y un presupuesto para el proyecto Entregar incremento 1 Generación del *feedback* Retrospectiva Comunicación de progresos
Entorno de negocio	Describir qué es una trayectoria 4D y sus implicaciones en ATM Describir la tecnología ADS-B Descripción de las operaciones ATM en un aeropuerto Describir el caso del aeropuerto de Madrid Introducción al entorno de negocio relacionado con la festión del tráfico aéreo (ATM)
Contexto científico - técnico	Introducción al *Deep Learning* Descripción y funcionamiento de las CNN Descripción y funcionamiento de las RNN Descripción y funcionamiento de una red neuronal Introducción general al *Machine Learning* (ML) Caracterización de las librerías para desarrollo de modelos de DL en Python
Estado del arte	Investigar si existen ya trabajos sobre este campo de estudio Análisis comparativo de los trabajos seleccionados y conclusiones sobre el estado del arte estudiado Propuesta de las dimensiones comparativas del trabajo relacionado en el contexto del proyecto Resumen individual de los diferentes trabajos relacionados Introducción general al *Machine Learning* (ML) Caracterización de las librerías para desarrollo de modelos de DL en Python
Memoria Sprint #1	Elaborar la introducción al proyecto Redacción del Contexto Establecer una metodología de trabajo, una planificación y un presupuesto para el proyecto
Comunicación #1	Presentación Sprint#1 Redacción del Contexto Establecer una metodología de trabajo, una planificación y un presupuesto para el proyecto

Tabla 2.1: Planificación de tareas del Sprint 1

Sprint 2	
Tarea	Subtarea
Caracterización del proyecto	Definir los objetivos y las tareas a realizar en el sprint Reunión de inicio Reunión de sincronización Corrección memoria Sprint 1 Comunicación Oficial de Inicio (COI) Descripción de la metodología de trabajo Entregar incremento 2 Generación del *feedback* Retrospectiva Comunicación de progresos
Desarrollo Técnico	Describir generalmente las tecnologías en la implementación del *pipeline* Diseñar el pipeline de transformación (*logical datamap*) Modelar el *smart data* (diseño conceptual) Transformación del *raw data* en *smart data* Construir los perfiles característicos del *raw data* Caracterizar los parámetros de post-procesamiento del *smart data* Procedimiento parametrizado para el post-procesamiento del *smart data* Construcción del modelo *baseline*
Comunicación #2	Presentación Sprint#2 Redacción del Contexto Realizar presentación Sprint 2

Tabla 2.2: Planificación de tareas del Sprint 2

Sprint 3	
Tarea	Subtarea
Caracterización del proyecto	Definir los objetivos y las tareas a realizar en el sprint Reunión de inicio Reunión de sincronización Corrección memoria Sprint 2 Entregar incremento 3 Generación del *feedback* Retrospectiva Comunicación de progresos
Desarrollo Técnico	Diseñar el *pipeline* de transformación *(logical datamap)* Describir generalmente las tecnologías en la implementación del *pipeline* Procedimiento parametrizado para el post-procesamiento del *smart data* Caracterizar los parámetros de post-procesamiento del *smart data* Construcción del modelo *baseline* Descripción de tecnologías utilizadas para la construcción del modelo *baseline* Procedimiento para la construcción del modelo
Comunicación #3	Presentación Sprint#3 Redacción del Contexto Realizar presentación Sprint 3

Tabla 2.3: Planificación de tareas del Sprint 3

Sprint 4	
Tarea	Subtarea
Caracterización del proyecto	Definir los objetivos y las tareas a realizar en el sprint Reunión de inicio Reunión de sincronización Corrección memoria Sprint 3 Entregar incremento 4 Generación del *feedback* Retrospectiva Comunicación de progresos
Desarrollo Técnico	Diseñar el *pipeline* de transformación *(logical datamap)* Describir generalmente las tecnologías en la implementación del pipeline Procedimiento parametrizado para el post-procesamiento del *smart data* Caracterizar los parámetros de post-procesamiento del *smart data* Construcción del modelo *baseline* Descripción de tecnologías utilizadas para la construcción del modelo *baseline* Procedimiento para la construcción del modelo
Comunicación #4	Presentación Sprint#4 Realizar presentación Sprint 4

Tabla 2.4: Planificación de tareas del Sprint 4

Sprint 5	
Tarea	Subtarea
Caracterización del proyecto	Definir los objetivos y las tareas a realizar en el sprint Reunión de inicio Reunión de sincronización Corrección memoria Sprint 4 Terminar memoria TFG Entregar TFG Generación del feedback Retrospectiva Comunicación de progresos
Desarrollo Técnico	Diseñar el *pipeline* de transformación *(logical datamap)* Describir generalmente las tecnologías en la implementación del *pipeline* Procedimiento parametrizado para el post-procesamiento del *smart data* Caracterizar los parámetros de post-procesamiento del *smart data* Construcción del modelo *baseline* Descripción de tecnologías utilizadas para la construcción del modelo *baseline* Procedimiento para la construcción del modelo Experimentar con distintos modelos
Comunicación #5	Presentación Sprint#5 Realizar presentación Sprint Final

Tabla 2.5: Planificación de tareas del Sprint 5

2.2.1. Herramientas empleadas

Para el desarrollo de este proyecto se han empleado las siguientes herramientas:

- Overleaf: herramienta en línea para la generación de código LaTeX.
- Teams: herramienta de comunicación entre el tutor y el alumno, además de ser el sitio de entrega de los diferentes incrementos.
- Trello: herramienta para dar soporte al tablero de proyecto de UVagile. En la Figura 2.2 se pueden ver las diferentes columnas del tablero de Trello, además de las tareas.

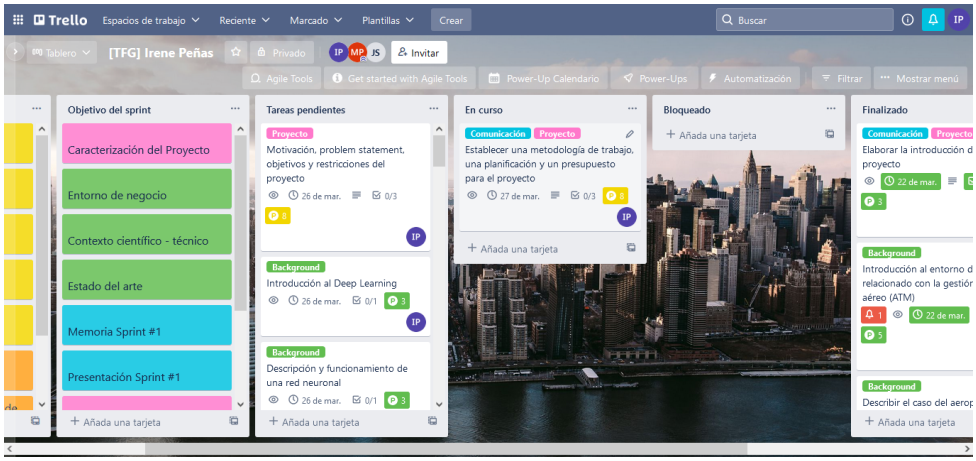


Figura 2.2: Tablero TFG Trello

2.3. Planificación temporal

En la Figuras 2.4 a 2.8 se puede observar la planificación temporal que se ha seguido. Además se ha utilizado una leyenda de colores (ver Figura 2.3) para identificar si la tarea se ha realizado en tiempo o si ha habido problemas.

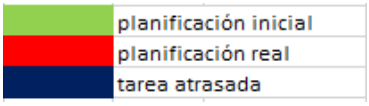


Figura 2.3: Leyenda seguida para la planificación

En la Tabla 2.6 se puede observar la planificación temporal que se ha seguido, con cada uno de los sprints realizados, la duración en horas, los puntos de historia que ha supuesto en total el sprint y los retrasos tanto en horas como en puntos.

SPRINT 1	FECHA (semanas)					
TAREA	14-20 mar	21-27 mar	28 mar-3 a	4-10 abr	11-17 ab	18-24 abr
Descripción de la metodología de trabajo						
Reunión de inicio						
Elaborar introducción del proyecto						
Introducción al entorno de negocio relacionado con ATM						
Descripción de las operaciones ATM en un aeropuerto						
Describir la tecnología ADS-B						
Describir qué es una trayectoria 4D y sus implicaciones en ATM						
Introducción general al ML						
Descripción y funcionamiento de las RNN						
Descripción y funcionamiento de las CNN						
Introducción al DL						
Caracterización de las librerías para desarrollo de modelos de DL en Python						
Descripción y funcionamiento de una red neuronal						
Motivación, problem statement, objetivos y restricciones del proyecto						
Redacción del contexto						
Investigar si existen ya trabajos sobre este campo de estudio						
Análisis comparativo de los trabajos seleccionados y conclusiones sobre el estado del arte estudiado						
Construcción de la bibliografía						
Entregar incremento 1						
Propuesta de las dimensiones comparativas del trabajo relacionado en el contexto del proyecto						
Estimar presupuesto del proyecto						
Planificación temporal del proyecto						
Realizar presentación sprint 1						
Resumen individual de los diferentes trabajos relacionados						
Reunión de sincronización						
Generación de feedback						
Retrospectiva						
Comunicación de progresos						

Figura 2.4: Planificación Sprint 1

SPRINT 2	FECHA (semanas)						
TAREA	11-17 abr	18-24 abr	25 abr-1 m	2-8 may	9-15 may	16-22 may	23-29 may
Corrección memoria sprint 1							
Reunión inicio							
Comunicación de oficial inicio (COI)							
Describir generalmente las tecnologías en la implementación del pipeline							
Diseñar el pipeline de transformación (logical datamap)							
Modelar el smart data (diseño conceptual)							
Transformación del raw data en smart data							
Construir los perfiles característicos del raw data							
Caracterizar los parámetros de post-procesamiento del smart data							
Procedimiento parametrizado para el post-procesamiento del smart data							
Construcción del modelo baseline							
Entregar el incremento 2							
Realizar presentación sprint 2							
Estimar presupuesto del proyecto							
Reunión de sincronización							
Generación de feedback							
Retrospectiva							
Comunicación de progresos							
Planificación temporal del proyecto							

Figura 2.5: Planificación Sprint 2

SPRINT 3	FECHA (semanas)		
TAREA	2-5 jun	6-12 jun	13 -19 jun
Corrección memoria sprint 2			
Reunión inicio			
Diseñar el pipeline de transformación (logical datamap)			
Describir generalmente las tecnologías en la implementación del pipeline			
Procedimiento parametrizado para el post-procesamiento del smart data			
Construcción del modelo baseline			
Descripción general de las tecnologías utilizadas para la construcción del modelo baseline			
Procedimiento para la construcción del modelo			
Realización presentación Sprint 3			
Reunión de sincronización			
Generación de feedback			
Retrospectiva			
Comunicación de progresos			

Figura 2.6: Planificación Sprint 3

SPRINT 3		FECHA (semanas)		
TAREA	PUNTOS HISTORIA	20-26 jun	27 jun-3 jul	4 -10 jul
Corrección memoria sprint 3	5			
Reunión inicio	0			
Diseñar el pipeline de transformación (logical datamap)	2			
Procedimiento parametrizado para el post-procesamiento del smart data	2			
Describir generalmente las tecnologías en la implementación del pipeline	2			
Caracterizar los parámetros del proceso de post-procesamiento del smart data	2			
Construcción del modelo baseline	5			
Descripción general de las tecnologías utilizadas para la construcción del modelo baseline	2			
Procedimiento para la construcción del modelo	2			
Realización presentación Sprint 4	3			
Reunión de sincronización	0			
Generación de feedback	0			
Retrospectiva	0			
Comunicación de progresos	0			

Figura 2.7: Planificación Sprint 4

SPRINT 5		FECHA (semanas)								
TAREA	PUNTOS HISTORIA	11-17 jul	18-24 jul	25 -31 jul	1-7 ago	8-14 ago	15-21 ago	22-28 ago	29 ago-4sept	5-11 sept
Corrección memoria sprint 4	5									
Reunión inicio	0									
Diseñar el pipeline de transformación (logical datamap)	2									
Procedimiento parametrizado para el post-procesamiento del smart data	2									
Describir generalmente las tecnologías en la implementación del pipeline	2									
Caracterizar los parámetros del proceso de post-procesamiento del smart data	2									
Construcción del modelo baseline	5									
Descripción general de las tecnologías utilizadas para la construcción del modelo baseline	2									
Procedimiento para la construcción del modelo	2									
Experimentar con distintos modelos	3									
Terminar memoria TFG	8									
Realización presentación Sprint Final	3									
Reunión de sincronización	0									
Generación de feedback	0									
Retrospectiva	0									
Comunicación de progresos	0									
Entregar TFG	1									
SUMA TOTAL DE PUNTOS DE HISTORIA	37									

Figura 2.8: Planificación Sprint 5

Sprint	Duración estimada	Puntos historia iniciales	Duración final	Puntos historia realizados
1	60:00:00	101	22:05:00	95
2	60:00:00	45	58:00:00	32
3	60:00:00	25	35:30:00	10
4	60:00:00	25	109:00:00	10
5	60:00:00	37	87:05:00	37

Tabla 2.6: Planificación temporal

2.4. Presupuestos

A la hora de realizar los presupuestos se deben tener en cuenta tres conceptos principales: *hardware*, *software* y costes de personal. Para las operaciones se utilizan las siguientes fórmulas:

$$\textit{Coste por mes (€/mes)} = \frac{\textit{Coste total (€)}}{\textit{Vida útil (meses)}} \tag{2.1}$$

$$\textit{Coste real por mes (€/mes)} = \textit{Coste por mes} \times \textit{Porcentaje de uso} \tag{2.2}$$

$$\textit{Coste real (€)} = \textit{Coste real por mes} \times \textit{Meses de uso (meses)} \tag{2.3}$$

Costes de *Hardware*: en la Tabla 2.7 se ven detalladamente.

Herramienta	Coste total (€)	Vida útil	% Uso	Meses uso	Coste real (€)
Ordenador personal	850 €	4 años	80 %	4,5	60 €
Conexión a Internet	30 €/ mes	-	80 %	4,5	108 €
				Coste total	168 €

Tabla 2.7: Presupuesto hardware

Costes de *Software*: en la Tabla 2.8 se ven detalladamente. Todas las aplicaciones utilizadas son gratis o se tiene licencia de estudiante gratuita.

Herramienta	Coste total (€)	Vida útil	% Uso	Meses uso	Coste real (€)
Microsoft Teams	-	-	50 %	4,5	0 €
Overleaf	-	-	60 %	4,5	0 €
Jupyter Notebook (Python)	-	-	60 %	4,5	0 €
Trello	-	-	30 %	4,5	0 €
EasyRetro	-	-	5 %	4,5	0 €
Excel	-	-	5 %	4,5	0 €
				Coste total	0 €

Tabla 2.8: Presupuesto software

Costes de personal: dichos costes van a ser reflejados mediante dos tablas. Durante el proyecto una misma persona va pasando por los diferentes puestos, por lo que hay varios roles desempeñados (Tabla 2.9). A su vez, esta persona tiene que estar dada de alta en la Seguridad Social, por lo que habrá que calcular los costes de cotización para la empresa derivados de su contratación (Tabla 2.10). Dichos costes se corresponden con el 32,1 % del sueldo bruto de la persona.

Por lo tanto, los costes totales previstos de todo el proyecto se recogen en la Tabla 2.11.

2.5. Balance temporal y económico

En esta sección se va a describir el balance temporal y económico del proyecto.

Puesto	Salario (€/hora)	Horas	Coste total (€)
Desarrollador de Python	15,50 €/ hora	180	2790,00 €
Científico de datos	16,60 €/ hora	120	1992,00 €

Tabla 2.9: Presupuesto para personal

Causa	Coste real (€)
Sueldo Desarrollador de Python	2790,00 €
Sueldo Científico de datos	1992,00 €
Seguridad Social	1535,02 €
Total	6317,02 €

Tabla 2.10: Presupuesto para personal (con gastos de Seguridad Social)

Presupuesto	Coste (€)
Hardware	168,00 €
Software	0 €
Personal	6317,02 €
Total	6485,02 €

Tabla 2.11: Presupuesto total del proyecto

2.5.1. Balance temporal

Al principio del proyecto se estableció una planificación temporal, la cual, debido a algunos contratiempos, ha supuesto una serie de modificaciones en la dicha temporal.

- Hubo una serie de problemas al instalar algunas de las librerías de Python en el entorno Anaconda, por lo que se retrasó el inicio de la parte práctica, es decir, la transformación de los datos a matrices y el posterior entrenamiento del modelo. A su vez, la construcción de las matrices fue más complejo de lo que en un principio se estimó, por lo que el proyecto se retrasó casi un mes.
- Por otro lado, el proyecto se inició a la vez que las prácticas curriculares de empresa, y con el curso de la escuela de idiomas, por lo que no se pudieron dedicar todas las horas que se quería. Además, en junio con los exámenes de la escuela de idiomas hubo varios días que no se pudo dedicar el tiempo requerido.

Debido a estos inconvenientes, finalmente se decidió presentar el proyecto en Septiembre. En la Figura 2.9 se puede ver el resumen de los puntos de historia y duración de cada uno de los sprints.

2.5.2. Balance económico

Los costes del proyecto han sufrido modificaciones debido al incremento en el tiempo que ha sufrido el proyecto. Dichos costes recalculados se detallan en las Tablas 2.12 a 2.15.
Costes de *Hardware*:

RESUMEN SPRINTS		FECHA (meses)						
TAREA	PUNTOS HISTORIA	marzo	abril	mayo	junio	julio	agosto	septiembre
Sprint 1	101							
Sprint 2	45							
Sprint 3	25							
Sprint 4	25							
Sprint 5	37							
SUMA TOTAL DE PUNTOS DE HISTORIA	233							

Figura 2.9: Resumen sprints

Herramienta	Coste total (€)	Vida útil	% Uso	Meses uso	Coste real (€)
Ordenador personal	850 €	4 años	80 %	6,5	92 €
Conexión a Internet	30 €mes	-	80 %	6,5	156 €
				Coste total	248 €

Tabla 2.12: Presupuesto hardware

Costes de *Software*: no han sufrido ninguna desviación debido a que las herramientas son libres o se tiene licencia de estudiante.
Costes de personal:

Puesto	Salario (€/hora)	Horas	Coste total (€)
Desarrollador de Python	15,50 €/ hora	180	2790,00 €
Científico de datos	16,60 €/ hora	120	1992,00 €

Tabla 2.13: Presupuesto para personal

Causa	Coste real (€)
Sueldo Desarrollador de Python	2790,00 €
Sueldo Científico de datos	1992,00 €
Seguridad Social	1535,02 €
Total	6317,02 €

Tabla 2.14: Presupuesto para personal (con gastos de Seguridad Social)

Por lo tanto, el balance de los costes totales de todo el proyecto serán los reflejados en la Tabla 2.15.

Presupuesto	Coste (€)
Hardware	248,00 €
Software	0 €
Personal	6317,02 €
Total	6565,02 €

Tabla 2.15: Presupuesto total del proyecto

Capítulo 3

Contexto del Trabajo

3.1. Entorno de negocio

La gestión del tráfico aéreo (ATM, por sus siglas en inglés *Air Traffic Management*) se entiende como la interacción del avión con todos los sistemas que le son de ayuda para despegar desde un aeropuerto de origen, incorporándose así a un espacio aéreo de tránsito y aterrizar en un aeropuerto de destino. ATM incluye los siguientes aspectos: control del tránsito aéreo, personal de seguridad del mismo, meteorología aeronáutica, sistemas de navegación, gestión del espacio, servicios y control de flujo [80].

En la gestión de tráfico aéreo, se usan sistemas interoperables que permiten que un avión opere con un cambio mínimo de rendimiento para transitar de un espacio aéreo a otro. Para entender esto más profundamente, hay que comprender qué es un espacio aéreo.

Un espacio aéreo esta formado por zonas de gestión ATC (*Air Traffic Control*). Cada zona de gestión del ATC tiene que conocer los detalles generales de cada vuelo en el que esté implicado antes de que este ocurra. Cada zona de gestión ATC informará a la siguiente de cuándo tiene que esperar que el vuelo pase por su espacio aéreo, y comunicar cualquier cambio. Además, las zonas de gestión se comunican con los pilotos por radio a diferentes frecuencias. El ATC debe ser informado del vuelo al menos 30 minutos antes del despegue.

A su vez, cada zona de gestión del ATC puede subdividirse en sectores. Estos son operados por 1 o 2 controladores, y pueden controlar unas 15 aeronaves.

Durante el vuelo, el procesamiento de información es necesario para conocer la posición del avión en cada momento. Es necesario tener la latitud, la longitud y la altitud a la que vuela la aeronave para evadir obstáculos que puedan resultar peligrosos. Esto es conocido como la navegación 3D.

Sin embargo, la navegación 3D no es suficiente, y surgen algunos problemas, como el congestionamiento del espacio aéreo en muchas partes del mundo. Este cuello de botella afectó a más de 25 millones de pasajeros de las aerolíneas que conforman *Airlines For Europe* [74]. Debido a este inconveniente, se vuelve de vital importancia agregar otra variable más a las ya existentes: el tiempo, originando el concepto de navegación 4D [32].

$$4D = latitud + longitud + altitud + tiempo \tag{3.1}$$

La trayectoria 4D, con las 4 dimensiones citadas anteriormente, significa que cualquier retraso que ocurra es una distorsión de la trayectoria tanto como un cambio de nivel o un cambio de la

posición horizontal.

Si se usa la navegación 4D, al tener disponible un sistema de navegación que permita mantener sincronizadas las operaciones de las aeronaves, facilita la introducción de más aviones en el mismo sector del espacio aéreo sin incrementar el riesgo, por lo que el cuello de botella del congestionamiento del espacio aéreo se reduce.

El problema que hay que resolver ahora, es encontrar la trayectoria óptima entre un aeropuerto de origen y uno de destino, minimizando el consumo.

Actualmente, la organización del espacio aéreo gira en torno al concepto de aerovía fija, que es una ruta designada en el espacio aéreo, fijada mediante varios elementos de ayuda a la navegación. Es un componente básico de los sistemas de vuelo.

Ahora bien, este concepto se está quedando atrás. El futuro está en lo que se conoce como rutas libres. Las rutas libres son una solución con la que los pilotos podrán escoger sus propias rutas directas apoyados en una tecnología avanzada que evita potenciales conflictos en las trayectorias de los aviones [49, 52]. Esto no quiere decir que haya más aerovías, sino más vuelos por trayectoria. Dichas rutas libres se pueden lograr mediante la aplicación de la trayectoria 4D.

En las Figuras 3.1 y 3.2 se puede observar gráficamente los distintos tipos de ruta que existen: Servicio de tránsito aéreo (*Air Traffic Services*, ATS), que son las aerovías fijas y las rutas libres, explicadas anteriormente.

En la Figura 3.1 entra en juego el concepto de TMA (*Terminal Control Area*), en español Área de Control Terminal, que es un área designada de espacio aéreo controlado que rodea un aeropuerto principal donde hay un alto volumen de tráfico [90].

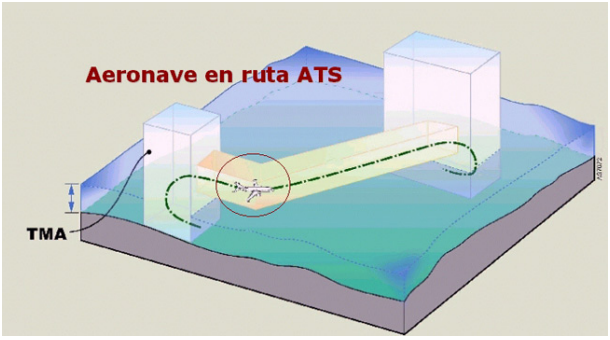


Figura 3.1: Ruta servicio de tránsito aéreo (ATS)

El objetivo de la trayectoria 4D es garantizar un vuelo libre de posibles inconvenientes, y la trayectoria óptima (a día de hoy la aeronave debe cumplir con precisión un tiempo de llegada a un punto dado). El mayor avance del sistema de control consiste en su capacidad de predicción de las trayectorias de las aeronaves bajo control mediante la tecnología 4D.

Según la DFS *(Deutsche Flugsicherung)*, incorporar esta tecnología dejaría obsoleto el sistema antiguo con fichas de papel del progreso del vuelo [32]. Este nuevo sistema proporcionaría datos en cuatro dimensiones de la ruta del vuelo para aquellos vuelos que se consideren relevantes para el control aéreo. Esto ayudaría a los controladores aéreos a anticipar y resolver posibles conflictos entre rutas de aeronaves en una fase temprana. Otra ventaja es que habría una mayor precisión en la planificación de los vuelos, por lo que habría una mayor puntualidad y se reducirían las situaciones en las que se debe alterar una ruta. A su vez habría una reducción de las emisiones, un aumento de las capacidades (en ruta y aeropuerto), es decir, los controladores podrían manejar

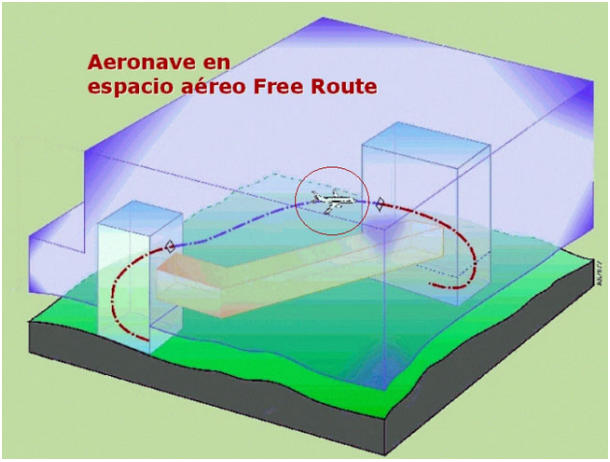


Figura 3.2: Ruta libre

con seguridad más tráfico, puesto que la información llegaría con mucha antelación.

Con las trayectorias 4D se reduciría la incertidumbre en la predicción de las trayectorias en los espacios aéreos. Ahora bien, después de investigar sobre las trayectorias 4D, no se debe olvidar otra tecnología que revolucionó toda la aviación, el radar. Esta tecnología no debe ser omitida, pues es todavía utilizada en la aviación y juega un papel fundamental en la reconstrucción de las trayectorias 4D.

3.1.1. Radar

El radar sirve para detectar objetos que se puedan aproximar a la aeronave. Esto se consigue gracias a la propiedad de reflexión de las ondas de radio. El principio del funcionamiento de un radar consiste en lo siguiente: un sistema electrónico provisto de una antena giratoria emite un haz de ondas electromagnéticas UHF (ultra alta frecuencia) que, cuando chocan con un objeto, son reflejadas y capturadas por la misma antena del sistema. Estas señales son procesadas y aparecen en forma de impulsos luminosos en la pantalla del controlador del tráfico aéreo, permitiéndole detectar objetos en vuelo en las cercanías del sistema de radar.

En las aeronaves existen 2 tipos de radares: los radares primarios y los secundarios.

- Radar primario: identifica objetos detectando reflexiones de señales de radiofrecuencia en dichos objetos. El problema es que solo detecta objetos en el espacio aéreo, pero no puede identificarlos: es decir, no sabe qué tipo de objeto ha detectado. Una desventaja del radar primario es que si hay cambios de altitud del objetivo, o atenuación de la señal debido a lluvia intensa por ejemplo, puede hacer que el objetivo visualizado se desvanezca.

- Radar secundario: codifica mensajes en forma de pulsos modulados en amplitud, realizando "interrogaciones" mediante trenes de pulsos, dirigidos a transpondedores. Los transpondedores son equipos a bordo de la aeronave. Un transpondedor es un receptor y transmisor de radio que opera en la frecuencia del radar. Estos detectan y codifican la interrogación de la estación base radar y responden en consecuencia. Por lo tanto, el radar secundario permite la identificación y seguimiento de blancos específicos en el espacio. Está asociado a un radar primario.

El radar secundario es más preciso que el radar primario.

Sin embargo, los radares tienen un problema, que radica en que una vez un avión sale al océano y está lejos de tierra, el radar pasa a ser de poca utilidad, principalmente debido a que entraña una mayor dificultad construir antenas receptoras que en tierra. La cobertura de los radares civiles se acaban a unos 370 kilómetros (200 millas náuticas) de la costa. Entonces, para un vuelo de Nueva York a Londres, se perdería el contacto con la aeronave durante aproximadamente la mitad del viaje.

En los últimos años se han producido casos en los que, de repente, el avión dejó de ser visto en el radar y desapareció sin dejar rastro alguno. Los casos más notorios fueron mientras se cruzaba el océano. Desde hace mucho tiempo se han intentado probar otros métodos, como el GPS, pero fue descartado debido a su coste. En este punto, es donde entra en juego una tecnología llamada ADS-B, que determina la posición del avión por satélite y emite dicha información a los receptores situados en tierra o en otras aeronaves.

3.1.2. ADS-B

Las siglas de la tecnología ADS-B significan lo siguiente:

- *Automatic*: Proceso automático sin intervención alguna.
- *Dependent*: Depende de la información recibida por los datos de navegación y señal GPS.
- *Surveillance*: Con el objetivo de dar servicio de identificación de tráfico y vigilancia.
- *Broadcasting*: Los datos se radiodifunden a todas las estaciones con capacidad de recepción.

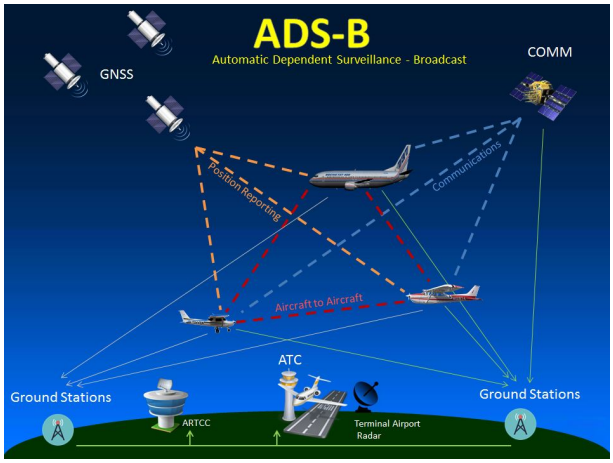


Figura 3.3: Estructura de la tecnología ADS-B

Habiendo visto el significado del nombre, es hora de ver cómo funciona el sistema y sus capacidades. La base del funcionamiento de ADS-B son los mensajes ADS-B. Actualmente, la mayoría de aeronaves mandan constantemente mensajes ADS-B. Desde el año 2020 en la aviación civil de Europa y Estados Unidos, es obligatorio que sean compatibles con ADS-B. Aquellas aeronaves que no cumplan dichos requisitos, tendrán que ser readaptadas o retiradas en un plazo concreto de tiempo [71]. La estructura de los mensajes se presenta en las Tablas 3.1 y 3.2:

La tecnología ADS-B tiene dos capacidades: OUT e IN

DF (5)	CA (3)	ICAO (24)	ME (56)	PI (24)

Tabla 3.1: Estructura principal de un mensaje ADS-B

Nº bits	Abreviatura	Información	Explicación
5	DF	*Downlink Format*	Determina la estructura del *payload*
3	CA	*Transponder capability*	Indica el nivel del transpondedor
24	ICAO	*ICAO aircraft address*	Dirección ICAO única a cada transpondedor en modo S de una aeronave. Identificador único para cada aeronave.
56	ME	*Message, extended squitter*	Tipo del mensaje (*payload*)
5	(TC)	*(Type code)*	Identifica qué información está contenida en un mensaje ADS-B: identificación de la aeronave, posición en superficie, etc
24	PI	*Parity/Interrogator ID*	Bit de paridad

Tabla 3.2: Significado de la estructura principal de un mensaje ADS-B

- *OUT*: capacidad de emitir información ADS-B.
  - Latitud y longitud, altitud barométrica y geométrica, y *ground speed* (velocidad horizontal de un avión en relación con la superficie de la Tierra [86]). Todos estos campos se obtienen de la señal GPS.
  - Rumbo y velocidad.
  - Número de vuelo ATC introducido en la fase de prevuelo de una aeronave (revisión del avión antes del despegue).
  - Indicador de situación de emergencia.
- *IN*: capacidad de recibir información que emiten otras estaciones ADS-B OUT. Para que esta tecnología pueda ser usada, es preciso tener un equipo a bordo que trabaje por enlace de datos *(Datalink)* en la banda de VHF.

Ventajas de ADS-B:

- Más fiable, debido a que los datos son enviados desde los equipos de navegación de la aeronave.
- Mayor velocidad de transmisión, puesto que esta tecnología emite 2 veces por segundo automáticamente, sin que ningún equipo le interrogue. Sólo es necesaria una antena con capacidad ADS-B *IN* para recibir los datos difundidos por la aeronave.
- Sustitución de las antenas de radar por otras receptoras de ADS-B, menos complejas, con menor coste y consumo eléctrico.
- Mejora en el espacio aéreo oceánico: hace posible que los aviones elijan mejores niveles sin verse "bloqueados" por otros aviones que se encuentren a una distancia no superior a la "distancia ITP" (*In trail Procedure*) [1]. La distancia ITP es una aplicación de ADS-B

que permite vuelos menos turbulentos durante viajes transoceánicos, además de ahorrar combustible y de reducir las emisiones.

Todo esto quiere decir, que si una aeronave quiere ascender o descender y cruzar el nivel de otra aeronave, y ambas están equipadas con la tecnología ADS-B *IN* y *OUT*, al enviar la solicitud al ATC, aparecerán las millas náuticas, el nivel y el identificador de las aeronaves. Así, el ATC dispondrá de información más precisa, valorando si se reúnen los requisitos necesarios para dar por bueno el cambio de nivel.

Aún así, habiendo hecho todos estos avances y habiendo visto todas las ventajas que estos suponen, sigue existiendo un inconveniente: el océano y las zonas remotas. Actualmente se está trabajando en la recepción de señales ADS-B *OUT* de los aviones mediante una constelación de satélites que vuelan a baja altura. A día de hoy, existen varias constelaciones de satélites de comunicaciones (Iridium, Inmarsat, Thuraya, Globalstar). Se componen de una gran cantidad de satélites de comunicaciones que orbitan alrededor de la Tierra en órbitas generalmente bajas, a una altura determinada de la Tierra, 780 km en el caso de Iridium [87].

3.1.3. Descripción de las operaciones ATM en un aeropuerto

La gestión del tráfico aéreo aúna todos los sistemas que asisten a la aeronave desde que despega del aeropuerto de origen, transita por el espacio aéreo, hasta que aterriza en el aeropuerto de destino. Las operaciones ATM en un aeropuerto son 4 principalmente [9, 24, 83]:

- Control del tráfico aéreo (ATC): servicio ofrecido por los controladores aéreos desde las torres de control, que dirigen la aeronave a través de una sección del espacio aéreo controlada. El propósito principal del control del tráfico aéreo es evitar colisiones, organizar y regular el tráfico de las aeronaves, y proporcionar información y apoyo a los pilotos.

- Servicios de tráfico aéreo (ATS): su principal objetivo es asegurar un flujo de tráfico ordenado y seguro, facilitado por los servicios ATC. A su vez, proveen información a la tripulación y en caso de emergencia activan las alertas necesarias. Los servicios ATS son facilitados en mayor medida por los controladores de vuelo. Las funciones más importantes de los controladores de vuelo son evitar colisiones, haciendo que haya un orden en el tráfico de los aviones. Otra de sus funciones es comunicarse con la tripulación durante todo el trayecto.

- Gestión del flujo del tráfico de vuelo (ATFM): su principal objetivo es regular el flujo de las aeronaves, tan eficientemente como se pueda, para así evitar la congestión de ciertos sectores de control.

- Gestión del espacio aéreo (ASM): el principal propósito es gestionar el espacio aéreo tan eficazmente como se pueda para así satisfacer a tantos usuarios como sea posible, debido a que es un recurso escaso. Hay que gestionar tanto las rutas, zonas, los niveles de altura de los vuelos, como la forma en que se estructuran todos estos, para así proporcionar servicios de tráfico aéreo.

Estos componentes interactúan de la siguiente manera cuando un avión se dispone a realizar una ruta: [4].

- Despegue: cuando el avión despega, el piloto activa el transpondedor de la aeronave. Este detecta las señales de radar entrantes, y lanza una señal amplificada y codificada en la

dirección de la onda de radar detectada. La señal del transpondedor da al controlador el número de vuelo, altitud, velocidad y destino. Así, en la pantalla del radar del controlador aparecerá una silueta de un avión, pudiendo seguir al mismo. Todo este proceso se desarrolla entre la aeronave y la torre de control del aeropuerto

- Durante la ruta: el avión va pasando por los distintos sectores de cada espacio aéreo. Allí va siendo monitorizado por varios controladores aéreos.
- Aproximación: el controlador de aproximación dirige al piloto (altitud, velocidad...) y le prepara para que pueda aterrizar en una pista.
- Aterrizaje: el controlador del aeropuerto de destino comprueba las pistas y el cielo. En el momento en el que considera que el aterrizaje es seguro, da paso al piloto para que pueda tomar tierra. El controlador también actualiza la información de las condiciones climatológicas y se las comunica al piloto. Por último, monitoriza el espacio entre el avión entrante y las demás aeronaves. Una vez en tierra el avión, el controlador dirige el avión por la pista, le da al piloto la nueva frecuencia de radio para establecer una comunicación con el controlador de tierra y empieza a comunicarse con él. El controlador de tierra se asegura de que cuando el avión va por la pista, no interfiera en el recorrido de otros aviones. Dirige el avión a una puerta de embarque, y proporciona apoyo para el estacionamiento en la misma.

3.1.4. Descripción del Aeropuerto Adolfo Suárez Madrid-Barajas

El aeropuerto Internacional Adolfo Suárez Madrid-Barajas (código IATA: MAD, código OACI: LEMD) es un aeropuerto español situado a 12 kilómetros del centro de la ciudad de Madrid. Es el aeropuerto español que más pasajeros tiene (en 2020 hubo un total de 17.112.389 pasajeros), y el quinto de toda Europa [78]. Barajas se construyó en 1931. Madrid-Barajas es un aeropuerto abierto 24 horas, pero tiene algunas excepciones, como por ejemplo la prohibición de su uso a aeronaves sin radiocomunicación y helicópteros. A continuación se indican las principales estructuras que conforman el aeropuerto, representadas a su vez en la Figura 3.4.

- Edificios: tiene 3 edificios terminales, uno satélite y 2 diques, además de una terminal dedicada a carga. Desde 2013, cuenta con una terminal de aviación ejecutiva.
- Terminales:
  - T1: cuenta con 43 puertas de embarque. Está formada por la Terminal Internacional y el nuevo Dique Sur.
  - T2: cuenta con 20 puertas de embarque. Está formada por la antigua Terminal Nacional y parte de la antigua Terminal Internacional.
  - T3: cuenta con 21 puertas de embarque. Está formada por el Dique Norte y en la actualidad se usa como una extensión de la T2.
  - T4: cuenta con 76 puertas de embarque.
  - T4-S: cuenta con 19 puertas de embarque en la zona *Schengen*, y otras 48 puertas de embarque de salidas internacionales..
- Hangares: hay 2 zonas principales de hangares:

Figura 3.4: Plano del aeropuerto Adolfo Suárez Madrid-Barajas

- Antigua Área Industrial, entre la T3 y la T4
- Área Industrial de La Muñoza

- Pistas: Barajas tiene 4 pistas físicas paralelas dos a dos: las 18L/36R - 18R/36L y las 14L/36L - 14R/32L. Estos códigos significan lo siguiente: en el caso del número, indica su orientación con respecto al norte, dividida entre 10 y redondeada [76]. En el caso de la letra (L o R), indica la posición de la pista respecto a las otras.

  En términos aeronáuticos, son 8 pistas diferentes, ya que sólo se usan simultáneamente 4 de ellas según las configuraciones de operación. Dependiendo de la meteorología, el ATC elige una u otra.

  - Configuración norte:
    - Durante el día (07-23h):
      - Pistas 36L y 36R: despegues.
      - Pistas 32L y 32R: aterrizajes.
    - Durante la noche (23-07h):
      - Pistas 36L: despegues.
      - Pistas 32R: aterrizajes.
    - No se autorizan despegues por las pistas 14L/14R.
  - Configuración sur:
    - Durante el día (07-23h):
      - Pistas 14L y 14R: despegues.
      - Pistas 18L y 18R: aterrizajes.
    - Durante la noche (23-07h):
      - Pistas 14L: despegues.
      - Pistas 18R: aterrizajes.

    - No se autorizan despegues por las pistas 32L/32R.

- Navegación aérea.

  - Torres de control:
    - Torre Norte: junto al edificio satélite. Es la torre de control principal y la más grande.
    - Torre Oeste: junto a la terminal T4. Gestiona el movimiento de rodadura de las aeronaves en tierra en torno a la terminal T4.
    - Torre Sur: junto a la terminal T2. Está situada en la terminal T2. Cuenta con capacidades operativas plenas, aunque ahora sólo gestiona las operaciones de rodadura de las aeronaves en torno a las terminales T1, T2 Y T3.

    Cualquiera de las tres torres tiene capacidad para controlar cualquier aspecto del tráfico aéreo y el movimiento de las aeronaves en el aeropuerto. Sin embargo, las tareas suelen estar repartidas entre las 3 torres como queda explicado anteriormente.
  - Ayudas a la navegación.
  - Sistemas de vigilancia:
    - 2 radares de superficie (SMR): vigilan los movimientos de las aeronaves y de cualquier vehículo en el área de maniobras. Se localizan en el campo de antenas de la torre Norte y en el de la torre Sur.
    - Sistema de multilateración: detecta blancos cooperativos (dotados de un transpondedor) en el área de maniobras.
  - Energía: en Barajas hay una central térmica de ciclo combinado propia. Abastece de energía eléctrica, calefacción y otras necesidades.

3.2. Contexto científico-técnico

En esta sección se van a exponer los conceptos básicos referentes al Aprendizaje Automático (ML), Aprendizaje Profundo (DL), Redes Neuronales Convolucionales (CNN) y Redes Neuronales Recurrentes (RNN). En la Figura 3.5 se muestra la relación entre estos conceptos: el ML es un área que se enmarca en el campo de estudio de la Inteligencia Artificial, siendo el DL un conjunto de técnicas y aproximaciones concretas que dentro del ML. Finalmente, se verán las librerías de Python utilizadas para la implementación de procesos relacionados con ML.

3.2.1. Introducción general al Aprendizaje Automático

El *Machine Learning* (ML), es una rama de la Inteligencia Artificial que se preocupa del diseño y desarrollo de algoritmos que permiten a los ordenadores mejorar su desempeño en la realización de una tarea a partir de la experiencia. Algunos usos del ML son el reconocimiento de habla, la medicina personalizada, el análisis del genoma, etc.

A la hora de aplicar ML, uno de los aspectos más complicados es escribir los algoritmos para resolver los problemas. Entonces, como solución al citado inconveniente, se pensó en que los ordenadores se programaran a sí mismos. Sin embargo, esto no es posible sin la experiencia, que en este caso, viene dada en forma de datos. De forma resumida, el funcionamiento del ML es el siguiente: se necesita un algoritmo, al cual se le proporciona una serie de datos de entrada,

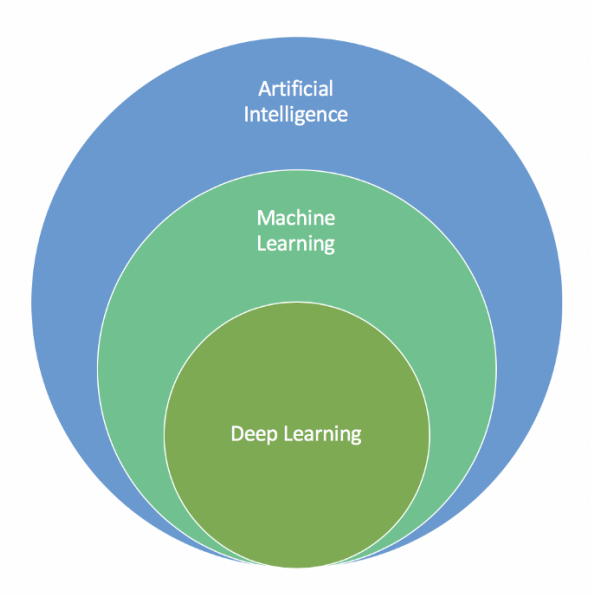


Figura 3.5: Esquema general IA, ML y DL

con las salidas esperadas correspondientes. El algoritmo ajustará sus parámetros internos, de manera que se "auto-programa" para ofrecer la salida correcta para cada entrada. Al pasar los datos por dicho algoritmo, se obtiene como salida un modelo entrenado, es decir, un "programa" especializado en resolver un problema o en realizar una tarea, descrito por las clases de entrada.

Mediante el ML, se pueden realizar las siguientes tareas:

- Clasificación: al llegar una nueva instancia, esta es asignada a una categoría.
- Regresión: relacionar una variable objetivo (continua) y cierto número de características.
- *Clustering*: a partir de instancias dadas, se forman grupos de elementos similares.
- Reconocimiento de patrones: se identifican estructuras frecuentes en los datos.
- Predicción: a partir de unas instancias dadas, estimar comportamientos.
- Minería de datos: descubrimiento de patrones en grandes volúmenes de conjuntos de datos
- Identificar patrones: reconocimiento de tendencias que puedan seguir los datos.

En el *Machine Learning* existen diferentes tipos de aprendizaje:

- Aprendizaje supervisado: se poseen los datos de ejemplo y los resultados de esos ejemplos (datos etiquetados). El objetivo es predecir la respuesta correcta para los nuevos ejemplos desconocidos que puedan llegar [11]. El esquema básico del aprendizaje supervisado consiste en un conjunto de datos de entrenamiento *(train)* y un conjunto de prueba *(test)*. Con el conjunto de datos de entrenamiento se aplica el algoritmo de aprendizaje, para que el modelo aprenda. Como resultado se obtiene un modelo, el cual se aplica con los datos de prueba para ver cómo predice. El problema que se presenta es encontrar una hipótesis (descripción del concepto objetivo), que concuerde con los ejemplos presentados y que

generalice a los casos nuevos. A continuación se presentan una serie de conceptos clave a la hora de aplicar el aprendizaje supervisado. Lo más sencillo es aprender una función a partir de los ejemplos.

- f es la función objetivo.
- Los ejemplos vienen dados por pares $(x, f(x))$.

El problema que existe es encontrar una hipótesis h, de tal manera que $h \approx f$, para un conjunto de ejemplos de entrenamiento dado. Por lo general, las hipótesis más simples tienden a generalizar mejor los datos futuros. Existen algunos problemas referentes al entrenamiento. A continuación quedan detallados:

- Sobreajuste: una hipótesis $h \in H$ se sobreajusta al conjunto de ejemplos de entrenamiento, si $\exists h' \in H$, que se ajusta peor que h a los ejemplos de entrenamiento, pero que actúa mejor sobre la distribución completa de instancias.
- Además del sobreajuste, se puede dar también el infraajuste, en el cual el modelo es muy simple para explicar las diferencias. Lo óptimo es tener un ajuste apropiado.

El sobreajuste puede ser consecuencia de ruido, es decir, atributos incorrectamente clasificados; de atributos no relevantes pero en un principio presentan una aparente regularidad, también llamados *outliers*; o también por conjuntos pequeños de entrenamiento (sesgados). Lo que se quiere hacer es aprender el concepto objetivo $f : X \longrightarrow Y$. Para ello existen dos tipos de aprendizaje supervisado:

- Regresión: el resultado es un valor numérico, dentro de un conjunto infinito de posibles resultados. Y es continuo. Lo que hace la regresión es aprender la forma de la función. Un ejemplo de regresión sería la regresión lineal.
- Clasificación: el resultado es una clase, entre un número limitado de clases. En este caso Y es discreto. Lo que hace la aplicación es aprender las características de cada clase. Algunos ejemplos de clasificación son SVM, K-NN y Redes neuronales.

- Aprendizaje no supervisado: se tienen los datos de ejemplo, pero no están etiquetados (resultado desconocido). En este caso, el objetivo es encontrar patrones en los datos (agrupamientos "naturales" de los datos). Existen dos tipos de aprendizaje no supervisado:
  - *Clustering*: se trata de buscar agrupaciones entre los datos. Algunos de los algoritmos principales son: k-medias, k-medianas, Cobweb, mapas autoorganizados (SOM)...
  - Reducción de la dimensionalidad: se trata de reducir el número de características o atributos de cada ejemplo sin perder información. Suele ser utilizada antes del aprendizaje supervisado para visualizar o pre-procesar los datos. Uno de los algoritmos principales es el Análisis de Componentes Principales (PCA).
- Aprendizaje por refuerzo: se realizan acciones y se obtienen recompensas. El objetivo es aprender comportamientos que reporten mayores recompensas.
- Aprendizaje semi-supervisado: se tienen datos etiquetados y sin etiquetar.

- Aprendizaje no balanceado: es un problema de clasificación en el cual la distribución de los datos entre las clases de interés no están balanceadas. El número de ejemplos que representan la clase de interés es mucho menor que el de las otras clases.
- Clasificación multi-etiqueta: cada instancia está asociada a un conjunto de clases, en lugar de una sola.
- Aprendizaje por transferencia: para hallar la solución a un problema, se emplean patrones o modelos previamente entrenados para otro problema.

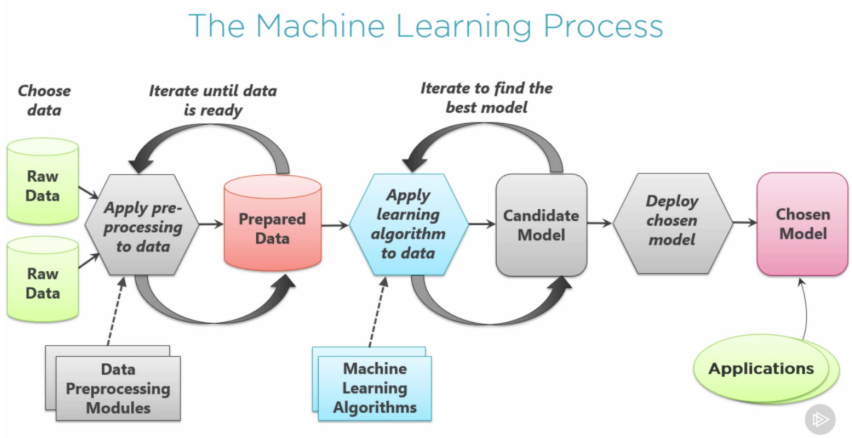


Figura 3.6: Proceso del *Machine Learning* [2]

El proceso de aprendizaje del *Machine Learning* se divide en las siguientes etapas:

- Preprocesado: dentro del preprocesado de datos se incluye la limpieza, transformación, integración y normalización de los datos. Además se incluye la imputación de datos perdidos y la identificación de ruido.
- Aprendizaje: en esta etapa se utiliza uno de los algoritmos de aprendizaje, de entre todos los tipos de aprendizajes vistos anteriormente. En el aprendizaje supervisado destacan los siguientes algoritmos de aprendizaje:
  - K-NN por su simplicidad y por no requerir un modelo ni ajustar parámetros. Aunque uno de sus defectos es que es demasiado simple y lento.
  - Árboles de decisión: tienen forma de árbol, en el que los nodos representan una condición, o bien una categoría o valor (nodos hoja). Se pueden usar para resolver problemas de clasificación y regresión.
  - *Ensembles*: combinan varios modelos de aprendizaje o usan diferentes conjuntos de entrenamiento dependiendo del tipo de *Ensemble*. El principal objetivo es mejorar la precisión del clasificador final. Tienen una función de combinación, que aúna todos los resultados. Se hace una votación, que puede ser ponderada, por mayoría u otras de más complejidad.
  - Regresión: usados sobre todo para predicción. Los métodos más habituales son la regresión lineal y la logística, siendo la primera para datos continuos y la segunda para datos discretos.

- Máquinas de Vectores Soporte (SVM): funcionan muy bien cuando se dispone de datos linealmente separables. El funcionamiento de los SVM es encontrar la mejor separación entre clases, es decir, aquella que maximice la anchura de la "calle" entre las clases. En caso de que las clases no fueran linealmente separables, se mapearía el *dataset* a una dimensión superior.
- Aprendizaje Bayesiano: es una aproximación probabilística para construir sistemas de aprendizaje bajo incertidumbre. De un conjunto de hipótesis, permite determinar cuál es la más probable para un conjunto de entrenamiento.
- Redes Neuronales: se inspiran en la neurona biológica. Es una herramienta muy potente, que funciona muy bien para la gran mayoría de algoritmos de ML. Son utilizadas para regresión y clasificación. Estos funcionan como una caja negra. Son robustas ante ruido y *outliers*. Algunas de las RRNN más conocidas son el Perceptrón y el Perceptrón Multicapa.
- Aprendizaje profundo: red neuronal compleja, con muchos niveles jerárquicos. En las primeras capas se aprenden cosas simples, las cuales van combinándose en las siguientes capas, y así sucesivamente. Algunos ejemplos de DL son las Redes Neuronales Convolucionales (CNN), las Redes Neuronales Recurrentes (RNN), y las Redes Neuronales de corto y largo plazo (LSTM).

En cuanto al aprendizaje no supervisado destacan los siguientes algoritmos de aprendizaje:

- K-medias: algoritmo básico de *clustering* (encontrar el particionado óptimo de los datos en N clústers exclusivos).
- Análisis de Componentes Principales (PCA): utilizada para reducir la dimensionalidad de los datos. Es decir, si entre varias variables existe una relación fuerte, será posible sustituirlas por un menor número de variables, y sin perder mucha información.

- Análisis del error: se evalúa el el resultado del aprendizaje. Suelen existir 3 conjuntos: entrenamiento, validación y test, aunque el de validación puede no existir. Para cada tarea pueden ser usadas varias técnicas, entre las que destacan:

  - División por porcentajes: se selecciona un porcentaje para entrenamiento y otro para pruebas.
  - Validación cruzada: se divide aleatoriamente el conjunto de datos en k subconjuntos disjuntos, y se realizan k aprendizajes. k suele tener un valor de 10. En cada iteración i, se usa el subconjunto i como el conjunto de prueba, y los $k-1$ restantes como subconjunto de entrenamiento. Para terminar, como medida de evaluación, se coge la media aritmética de las k iteraciones que fueron hechas.

  En cuanto a las clasificaciones realizadas, se catalogan en las categorías que se muestran en la Figura 3.7, en función del tipo de acierto o error cometido para cada una de ellas. En el caso de la clasificación, tal y como se puede ver en la imagen, existen dos tipos de errores: los falsos positivos y los falsos negativos. Dependiendo del problema, será más conveniente evitar uno u otro.

  Las métricas que existen para evaluar en clasificación son las siguientes:

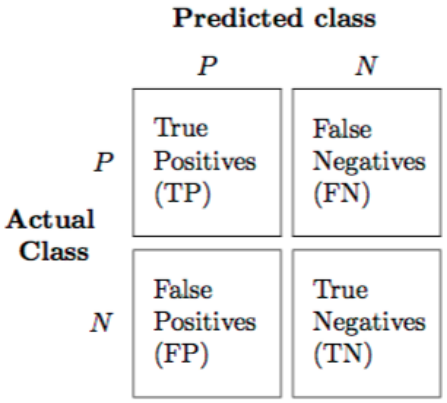


Figura 3.7: Métricas de catalogación del conjunto de datos

- Acierto (*Accuracy*):

$$Acierto = \frac{Clasificados\ Correctamente}{Todos\ Los\ Ejemplos}$$

- Precisión (*Precision*):

$$Precisión = \frac{Clasificados\ Correctamente\ Como\ Positivos}{Todos\ Los\ Ejemplos\ Clasificados\ Como\ Positivos}$$

- *Recall*:

$$Recall = \frac{Clasificados\ Correctamente\ Como\ Positivos}{Todos\ Los\ Verdaderamente\ Positivos}$$

- F1:

$$F1 = 2 \times \frac{Precisión \times Recall}{Precisión + Recall}$$

- Curva ROC (*Receiver Operating Characteristic*) y AUR (*Area Under ROC curve*).

En cuanto a problemas de regresión, las métricas de error más frecuentes son:

- MAE: *Mean Absolute Error* o Error Absoluto Medio.
- RMSE: *Root Mean Squared Error* o Error Cuadrático Medio.
- MAPE: *Mean Absolute Percentage Error* o Error Porcentual Absoluto Medio.

3.2.2. Concepto y funcionamiento de una red neuronal

Las Redes Neuronales (RRNN) son modelos matemáticos que están inspirados en la neurona biológica. Básicamente la neurona biológica es una célula receptora de señales electromagnéticas, de las cuales un 10 % provienen del exterior y un 90 % de otras neuronas, por medio de la sinapsis de las dendritas. La neurona, como se puede ver en la Figura 3.8, tiene una serie de elementos que en cierta manera serán replicados en un modelo para crear una neurona artificial. La neurona es la unidad de procesamiento básica de una red neuronal. Ésta tiene un conjunto de dendritas, que reciben información del exterior o de otras neuronas. Cada conexión tiene un valor de entrada y

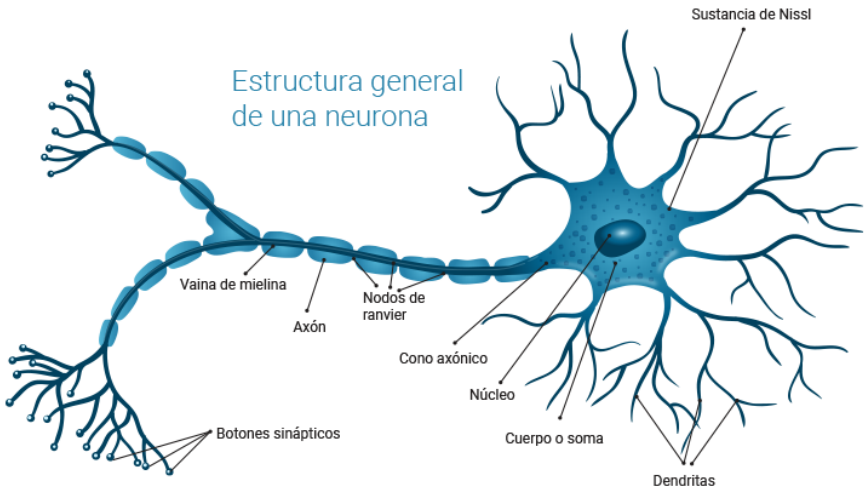


Figura 3.8: Neurona biológica

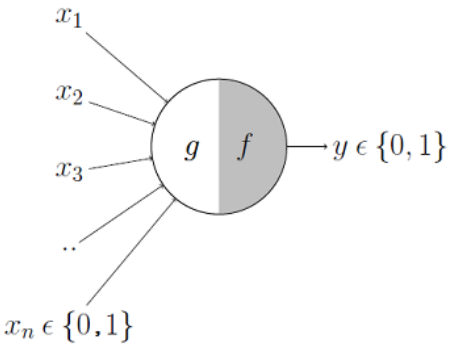


Figura 3.9: Neurona de *McCulloch-Pitts*

un peso. El cuerpo de la neurona sería una función sumatoria, que calcula la suma de todas las entradas. El disparo de la neurona sería la función de activación, la cual limita la amplitud de la salida de la neurona.

Los inicios de las RRNN datan de 1943, con el modelo de McCulloch-Pitts. Esta fue la base sobre la cual se empezaron a cimentar las redes neuronales. En el modelo de McCulloch-Pitts, se aproxima el comportamiento de las neuronas biológicas. Como puede observarse en la Figura 3.9, el modelo tenía una serie de entradas (x_n), que son recogidas por g (actúa como dendrita). g realiza una agregación basada en el valor de agregación de f, que toma una decisión. Esta neurona permitía simular el comportamiento de algunas funciones booleanas, como el caso del AND y del OR.

Diecinueve años más tarde, en 1962, Frank Rosenblatt inventó el perceptrón, el cual estaba basado en la neurona de McCulloch-Pitts. La diferencia con dicha neurona primigenia era que el perceptrón tenía unas entradas, a las cuales se las asociaba un peso. Al tener cada entrada un peso asociado, unas entradas tendrían una mayor relevancia que otras. Además de tener entradas y pesos, tenía un *bias* o sesgo, una función de activación y una salida.

Sin embargo, el problema reside en que una neurona no es suficiente para realizar buenos cálculos, sino que se necesitan múltiples neuronas en paralelo. Esto motivó la aparición de las redes neuronales (RRNN). Una red neuronal está compuesta por múltiples neuronas artificiales, cuyo mayor objetivo es clasificar (transformar las entradas en salidas significativas). De esta manera, varios perceptrones pueden conformar una red neuronal simple para clasificar (Figura 3.10).

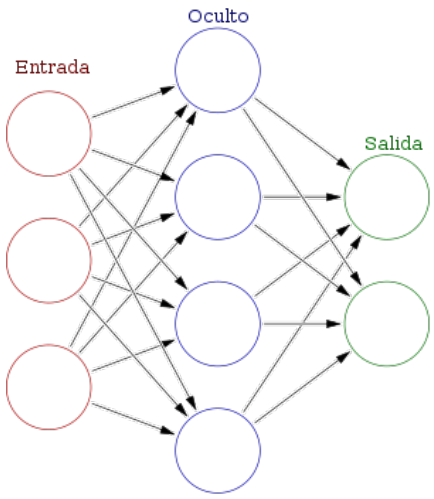


Figura 3.10: Red Neuronal Simple

El comportamiento de la función de una red neuronal, está determinado por los pesos asociados a las conexiones y los sesgos de las neuronas. Para determinar dichos pesos y el sesgo, lo que se hace es entrenar la red.

Proceso de aprendizaje de un perceptrón:

1. Inicialización: se fijan los pesos iniciales y el sesgo.
2. Para el conjunto de datos de entrenamiento:
   a) Se clasifican las entradas con la red actual. De esta manera se obtiene el valor de clasificación actual de la red y_{tred}.
   b) Comparar el resultado obtenido con el deseado, es decir, se calcula el error:
   $$e(t) = y_{subt} - y_{tred}$$
   c) Calcular la corrección de pesos, en caso de que entre ambos valores existan diferencias. Se realiza de la siguiente manera:
      - Para el sesgo: $aw_0 = \alpha \cdot e(t)$
      - Para las entradas: $\alpha \cdot x_{ti} \cdot e(t)$, donde α es el factor de aprendizaje, encargado de moderar las actualizaciones de los pesos.
   d) Por último se actualizan los pesos: $w_i = w_i + aw_i$.
3. Se repite el segundo punto hasta que se llegue a un criterio de convergencia. Uno de los más habituales es que el número de errores de los ejemplos sea menor que un determinado umbral.

En este punto, se debe resaltar un concepto, época (*epoch*), que es el paso del proceso de entrenamiento por todos los ejemplos de entrenamiento.

El perceptrón presenta muchas ventajas, pero también un gran inconveniente. El teorema de Minsky y Papert [11] sostiene que el algoritmo de aprendizaje de perceptrones converge en un número finito de pasos a un vector de pesos W, que clasifica correctamente todos los ejemplos de entrenamiento, siempre que estos sean linealmente separables y α suficientemente pequeño.

La clave que motiva este inconveniente son las palabras "linealmente separables". Algunas funciones booleanas, como XOR no se podían implementar, puesto que no son linealmente separables. Y las funciones que el perceptrón sí podía implementar, podían realizarse mediante otros métodos estadísticos más eficientes, por lo que se dejó de lado todo aquello relacionado con la IA. A partir de aquí hubo una época llamada "el invierno de la Inteligencia Artificial", en la que las investigaciones estaban casi paradas y no había apenas inversión en ellas, por lo que la sequía en la investigación de este campo se prolongó por más de 10 años.

No obstante, en 1986, se inventó el concepto de redes multicapa, que consiste en tener múltiples capas internas, con neuronas conectadas una detrás de otra. Y lo más relevante de todo, es que las redes multicapa permitían resolver problemas de clasificación con clases que no eran linealmente separables.

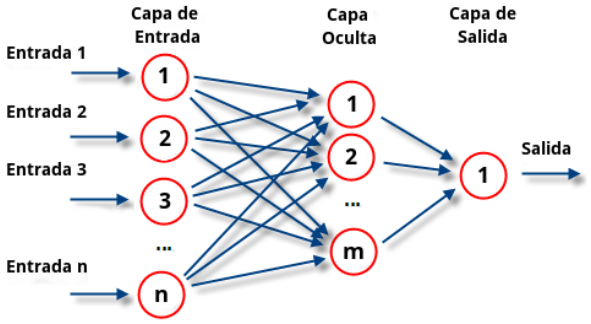


Figura 3.11: Estructura de una red neuronal multicapa

Funcionamiento de las redes neuronales multicapa:

En una red multicapa, las unidades se estructuran en capas, como se puede ver en la Figura 3.11. Las unidades de cada capa reciben su entrada de la salida de las unidades de la capa anterior.

Existen tres tipos de capas:

- Capa de entrada: capa en la cual se sitúan las unidades de entrada.
- Capas ocultas: contienen unidades no observables. El valor de cada unidad oculta es alguna función de los predictores que depende en parte del tipo de red. Los perceptrones multicapa pueden tener 1 ó 2 capas ocultas.
- Capa de salida: capa de las unidades cuya salida da al exterior.

Cuando se combinan las unidades en diferentes capas, se aumenta la capacidad expresiva de la red.

Además de lo anterior, que supuso un gran cambio, otro de los aspectos que dio un giro en relación al perceptrón, fue el uso de funciones de activación distintas. En un principio se utilizó

la función escalón, pero esta tenía una serie de limitaciones: en la función escalón solo había 2 valores, 0 o 1, es decir, confianza total o nula en la predicción. Es por esas desventajas que se empezó a usar la función sigmoidea. Esta función de activación es más suave, es continua y además diferenciable. Se pueden ver las distintas funciones de activación en la Figura 3.12.

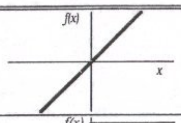
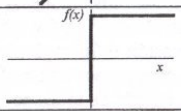
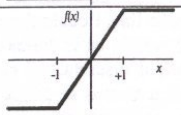
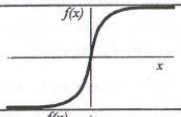
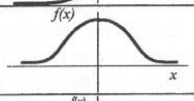
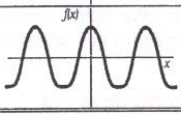
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 3.12: Funciones de activación

En este punto, al igual que con el perceptrón, hay que entrenar la red multicapa. Lo que se debe hacer es minimizar la pérdida del error del conjunto de datos y que la red se ajuste de la mejor manera posible a los ejemplos. Para ello existen varios métodos, entre los que destacan:

- *Hill Climbing* es una técnica de optimización matemática, que está incluida dentro de los algoritmos de búsqueda local.
- Descenso por gradiente: se eligen una serie de pesos al azar, y se van modificando progresivamente, mediante el factor de aprendizaje (haciendo pequeños desplazamientos en dirección opuesta al gradiente). Este algoritmo es de búsqueda local.
- Regla Delta: variación del descenso por gradiente. También es un algoritmo de búsqueda local.
- Algoritmo de retropropagación del error: basado en los mismos conceptos que el descenso por gradiente. Consiste en un proceso de actualizaciones sucesivas de los pesos. Este algoritmo parte de una estructura de red fija. Se intenta decidir cuántas capas ocultas y neuronas en cada capa hay que incluir.

En la actualidad hay múltiples tipos de redes neuronales, entre las que destacan:

- Redes Neuronales prealimentadas: sin estado interno. Aquí se engloba el perceptrón básico y el multicapa.

- Redes Neuronales Convolucionales (CNN)
- Redes Neuronales Recurrentes (RNN): con estado interno. En este apartado se incluyen las Redes de Hopfield y las LSTM.

3.2.3. Introducción general al *Deep Learning*

El aprendizaje profundo (*Deep Learning*, DL) es un campo que forma parte del *Machine Learning*, siendo este muy efectivo en el aprendizaje de patrones. Algunos de los campos en los que se emplea, son el reconocimiento del habla, la visión computacional y el procesamiento del lenguaje natural.

El DL trabaja con algoritmos que funcionan extrayendo conocimiento de los datos, usando una jerarquía de capas que podrían asemejarse a las redes neuronales del cerebro humano. A medida que se van avanzando en dichas capas, los datos van siendo más abstractos. Estas representaciones se van combinando a medida que se profundiza en la red.

Ahora bien, para tratar con el DL, una de las claves es proporcionar una gran masa de datos. Cuanta más información tengamos y más tiempo de computación para procesarla, más rica e interesante será la información que se extraiga al aplicar DL. Otro aspecto sumamente importante, es que las técnicas de aprendizaje profundo son capaces de crear modelos flexibles, que explotan la información que en principio está enterrada en grandes conjuntos de datos, haciendo que esta extracción sea mucho más eficiente que con otras técnicas habituales de ML, que extraen características de manera manual.

Fundamentos del *Deep Learning*

Para entrenar en *Deep Learning*, se necesita:

- La función de activación: función que devuelve una salida generada por la neurona dada una entrada o un conjunto de ellas. Hay varias funciones de activación. Algunas de ellas se muestran en la Figura 3.12.
- *Softmax* (capa de salida): suele ser usada en la salida de la red. La salida suele ser un valor numérico para cada una de las clases posibles, que es difícil de interpretar. Esta función transforma dichos valores numéricos al rango $[0, 1]$ que es más fácil de interpretar. La salida que se obtiene será la probabilidad de ocurrencia de cada una de las salidas posibles.
- Función de pérdida: es una de las piezas clave en el DL. Evalúa cómo de bien la red modela el conjunto de datos de entrenamiento. La función describe la diferencia entre las estimaciones hechas por la red para cada uno de los casos de clasificación y el valor real de ésta. Esto ya se realizaba con el perceptrón, pero esta es una versión más avanzada:

  $$e(t) = y_t - y_{tred}$$

  A la hora de interpretar los resultados, cuanto mejor clasifica la red, menor será el valor de la función de pérdida, y viceversa. El entrenamiento se realiza minimizando de manera progresiva la función de pérdida.

  Hay varios tipos de funciones de pérdida, entre las que destacan:

  - Error cuadrático medio (RMSE): si la red predice un escalar.

  - Semejanza del coseno: suele aplicarse en procesamiento del lenguaje natural.
  - Entropía cruzada: muy habitual a la hora de clasificar. Calcula la distancia entre dos distribuciones de probabilidad. Es aplicable sólo sobre distribuciones de probabilidad.

- Optimizador: es la parte encargada de minimizar la función de pérdida iterativamente. Los pesos se van actualizando iterativamente en la dirección en la que se minimice la función de pérdida. Hay muchos optimizadores, entre los que destacan AdaGrad (*Adaptative Gradient Algorithm*), Adam (*Adaptative Momentum*), etc.

Aparte de los conceptos tratados anteriormente, existe otro término muy relevante en DL.

- Regularización: encargada de hacer modificaciones pequeñas sobre el algoritmo de aprendizaje para que dicho algoritmo generalice mejor. Hay varios tipos de regularización, entre los que destacan L2 & L1, *Dropout*, parada temprana y el aumento de datos.

3.2.4. Descripción del concepto y funcionamiento de una red neuronal convolucional (CNN)

Las redes neuronales convolucionales (CNN) [11], son redes que tratan de imitar el córtex visual. Las CNN es un tipo de red neuronal de aprendizaje supervisado. Estas redes cuentan con capas ocultas especializadas. Dichas capas trabajan sobre volúmenes, mientras que hasta ahora lo que se hacía era trabajar con vectores.

El funcionamiento de las capas es jerárquico, es decir, que la función de las primeras capas es identificar elementos básicos, que se van combinando para así formar objetos en las capas más profundas. La red neuronal aprenderá a reconocer patrones, objetos, etc, por lo que es imprescindible poseer una gran cantidad de imágenes, para así poder extraer características únicas de cada objeto y de esta manera generalizarlo para poder identificar dicho objeto en diferentes escenarios.

Una imagen puede ser representada como una matriz con los valores de los píxeles de ésta. Dichos valores, pueden ser desde 0 a 255. Ese rango significa que cada color está representado por 8 bits.

$$2^8 = 256$$

En el caso de que la imagen sea en color, ésta tiene 3 canales: R (rojo) G (verde) B (azul). Esto quiere decir que si hay una imagen en color, es como tener 3 matrices apiladas. En el caso de que la imagen sea en escala de grises, sólo tiene un canal (sólo tendría una matriz).

En la Figura 3.13, se puede ver un ejemplo de una CNN básica para clasificación:

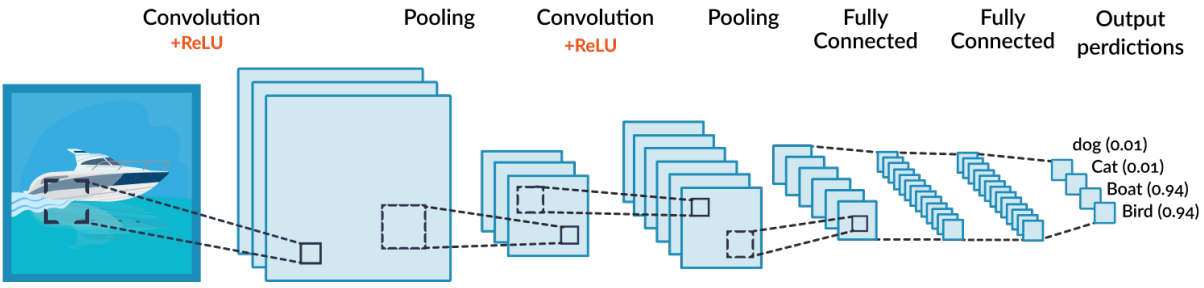


Figura 3.13: Red Neuronal Convolucional para clasificación

Los principales componentes dentro de una CNN son:

- Capa de Convolución: capa que se encarga de extraer las características (*features*) de la imagen. El proceso para extraer dichas características consiste en aplicar un filtro o *kernel* que va barriendo la imagen. Como norma general, dicho filtro suele ser de 3x3 o de 5x5.

  La operación de convolución multiplica los píxeles de la imagen por los pesos que se definieron en el *kernel* anteriormente. Después se suma el resultado total, y se obtiene el mapa de características (*feature map*). El filtro se va aplicando por toda la imagen, moviéndose cada vez un píxel a la derecha, hasta que se llega al final de la imagen (a la derecha del todo). Después, se vuelve a la izquierda pero bajando un píxel, y se repite la operación hasta que se llega al final de la imagen.

  A la hora de realizar la convolución se deben tener en cuenta 3 valores:

  - Profundidad (*Depth*): este parámetro es el número de filtros que se aplican. A medida que se utilicen más filtros, se generan más mapas de características.
  - Zancada (*Stride*): número de píxeles que se saltan al ir moviendo el *kernel*. Cuanto mayor sea el valor de la "zancada", más pequeños son los mapas de características.
  - Relleno (*Padding*): esta característica añade un marco de ceros alrededor de la imagen. De esta manera resulta más fácil aplicar los *kernels* sobre los elementos de los bordes y el mapa de características preserva el tamaño de la imagen original ($stride = 1$).

  Existen múltiples tipos de filtros, que tendrán diferentes efectos en las imágenes, algunos de estos son: identidad, detección de bordes, destacar los bordes, desenfoque y desenfoque Gaussiano [11].

- Función de activación: su función es limitar la amplitud de la salida de la neurona. Existen diferentes funciones de activación (ver Figura 3.12), entre las que destacan: identidad, paso binario, sigmoide, tangente hiperbólica (tanh) y unidad lineal rectificada (ReLU).

  Una de las funciones que más se usa actualmente es la ReLU. Esta función de activación tiene muchas ventajas, como reducir los problemas de *vanishing gradient*, en comparación con la función de activación sigmoide, que se satura en ambas direcciones.

- Capa de *Pooling*: utilizada para reducir el tamaño de la matriz. El funcionamiento consiste en aplicar un filtro sobre la salida de la capa anterior, y escoge un único número como salida. Como el tamaño de la matriz se reduce, la red podrá entrenar mucho más rápido. A continuación se muestran los diferentes tipos de *pooling* que existen:

  - *Max Pooling*: selecciona el valor máximo.
  - *Average Pooling*: selecciona el valor medio.
  - *Sum Pooling*: selecciona la suma de los valores.

  Una de las mayores ventajas del *pooling* es que reduce el número de los pesos de las capas, y en consecuencia, habrá menos neuronas pero sin perder información.

- Capa *Fully Connected*: es una capa neuronal multicapa normal, cuyo objetivo es combinar las características que fueron extraídas anteriormente. Su función principal es transformar la matriz tridimensional de características (generada en las convoluciones), en un vector

unidimensional. A este vector resultante, se le aplica *Softmax* (sólo en clasificación). De esta manera, se obtendrá la probabilidad de que la salida sea de una clase o de otra.

3.2.5. Concepto y funcionamiento de una red neuronal recurrente (RNN)

Las redes neuronales recurrentes [11], no tienen la estructura de capas definida estrictamente "hacia adelante", sino que permiten conexiones arbitrarias entre las neuronas, es decir, se pueden crear ciclos entre ellas, haciendo que tengan memoria. Este es un estado interno que mantiene la información, creando así la temporalidad, permitiendo que la red recuerde. Cada neurona de la capa oculta, puede establecer conexiones consigo misma y con otras neuronas de la capa oculta. Es por todo esto, que las RNN puedan ser utilizadas para solucionar problemas que requieren secuencias de datos, debido a que la red neuronal se puede utilizar en distintos instantes de tiempo, pasando la salida del instante anterior al actual.

En la Figura 3.14 se puede observar una red neuronal recurrente representada tanto en su forma compacta como en su forma desenrollada.

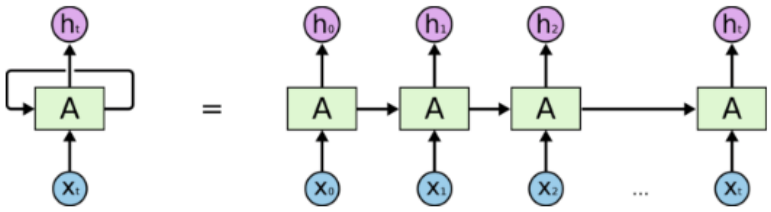


Figura 3.14: Red neuronal recurrente

Existe un concepto importante que se aplica a las RNN, *unrolling*, que significa que desenrollando la RNN un número n de veces, cada activación de las neuronas dentro de la red, son replicadas un número n de veces, siendo la misma red que se repite en los diferentes espacios de tiempo, manteniendo también los pesos de las capas ocultas para que sea posible memorizar. En cuanto a las entradas y salidas de las RNN, se pueden combinar como sea necesario, dependiendo de qué se quiera clasificar.

Las RNN tienen un gran potencial a la hora de analizar secuencias, como por ejemplo en el análisis de textos, sonido, vídeo, series temporales, reconocimiento del habla o procesamiento de lenguaje natural. No obstante, las RNN tienen algunos inconvenientes detallados a continuación:

- Las RNN son muy profundas, por lo que suelen ser muy difíciles de entrenar. Es por esto que la manera de entrenar las RNN es mediante el Mecanismo de Retropropagación a través del tiempo, *Backpropagation Trough Time (BPTT)*. Esta es una técnica basada en el gradiente para entrenar ciertos tipos de RNN.

- *Vanishing gradient*: las RNN no pueden recordar relaciones temporales lejanas. En cuanto a este inconveniente, existe una solución, las redes con memoria a corto y largo plazo, *Long Short-Term Memory networks (LSTM)*. Estas redes están diseñadas explícitamente para evitar el problema del *Vanishing gradient*.

3.2.6. Librerías para el desarrollo de modelos de DL en Python

Existen varias librerías para el desarrollo de modelos de *Deep Learning* en Python, entre las que destacan:

- TensorFlow: librería de Python desarrollada por Google, para realizar cálculos numéricos mediante grafos de flujo de datos. Se codifica un grafo, cuyos nodos son operaciones matemáticas, con cualquier número de entradas y salidas (variables y constantes también), y cuyas aristas son los tensores (matrices de datos multidimensionales), que definen el flujo de datos entre los nodos. TensorFlow es muy potente y flexible, escalable y portable.
- Keras: destaca por su facilidad de uso. Keras es un *framework* de alto nivel para aprendizaje profundo. Esta diseñado para facilitar la creación de redes neuronales profundas de manera sencilla y en pocos pasos. Keras usa otras librerías de DL (TensorFlow, CNTK o Theano).
- PyTorch: es una librería de Python, que permite el cálculo numérico eficiente en CPU y GPUs, implementando NumPy en una GPU, es decir, si se posee un procesador gráfico, se acelerará el tiempo de ejecución.
- Fast.ai: es una librería para DL de código abierto. El mayor propósito de Fast.ai es hacer el entrenamiento de las redes neuronales profundas lo más fácil, rápido y preciso posible usando buenas prácticas modernas. Fast.ai está escrito en Python. [16, 93].

3.3. Estado del arte

3.3.1. Descripción de trabajos relacionados

***1. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation. Network Speed Prediction* [95]**

En este documento, los autores proponen un método novedoso basado en CNN, que consiste en aprender del tráfico en forma de imágenes, y en base a ellas predecir la velocidad en toda la red de tráfico con una alta eficacia. Para ello, el tráfico pasa a convertirse en imágenes que describen el tiempo y el espacio del tráfico, usando matrices. El método propuesto se compara con 4 algoritmos: mínimos cuadrados ordinarios, k-vecinos, red neuronal artificial y *random forests*; y con 3 arquitecturas de *Deep learning*: *stacked autoencoder*, RNN y LSTM. Para probar el algoritmo CNN se realizaron 4 predicciones, que se diferenciaban unas de otras en los tiempos de predicción y en el tamaño de la información que se proporcionaba a la red. Los resultados muestran que el método propuesto mejora los otros algoritmos en un 42,91 % de media.

***2. Aircraft Prediction using LSTM Neural Network with Embedded Convolutional Layer* [55]**

En este artículo, los autores proponen una nueva arquitectura de red que incorpora capas convolucionales en células LSTM para predecir la trayectoria, basándose en las condiciones meteorológicas convectivas, junto con el plan de vuelo antes de que el avión despegue. En el experimento se usan datos históricos de las trayectorias de vuelo, el último plan de vuelo registrado y el mapa meteorológico convectivo dependiente del tiempo. Se toman datos de 3 meses con

vuelos entre 2 aeropuertos americanos. En este proyecto se desarrolla una interpolación sobre los planes de vuelo y las trayectorias históricas para fijar el número de células LSTM y reducir la complejidad computacional. La función de pérdida es la *Mean Squared Error* de las trayectorias predichas y las trayectorias históricas reales. El algoritmo de retropropagación del error utilizado es Adam. Al aprender de los datos históricos de vuelo, la prueba muestra que el 47 % de las trayectorias de vuelo predichas, pueden reducir la desviación comparada con el último plan de vuelo. La varianza global se reduce en un 12,3 %.

***3. One-Step Time-Dependent Future Video Frame Prediction with a Convolutional Encoder-Decoder Neural Network* [75]**

En este trabajo los autores plantean anticiparse a los cambios en el futuro a partir de un fotograma actual de un vídeo. Este trabajo amplía la capacidad de las CNN de predecir una anticipación de un suceso en un momento arbitrario en el futuro, no teniendo que ser necesariamente en el siguiente fotograma. En este proyecto, se explora la predicción de fotogramas, partiendo de un fotograma de entrada. Esto reduce la propagación y la acumulación de los errores de predicción. La salida es una imagen que se asemeja al futuro anticipado. En este método los autores proponen una serie de términos: una CNN de codificación, una CNN de decodificación, y una rama separada paralela al codificador, que modela el tiempo y permite generar predicciones a un tiempo vista en el futuro. A la hora de realizar los experimentos, se evalúa el método generando imágenes de los eventos del futuro anticipado a diferentes tiempos vista, comparándolos mediante *Mean Squared Error* con los verdaderos fotogramas futuros. Los parámetros de entrenamiento son: optimizador Adam con L2 y un 80 % del *dataset* para entrenamiento. Se usan 500.000 épocas con *minibatches*. El conjunto de datos tiene 6 acciones humanas diferentes, hechas por 25 actores diferentes. Como resultado, los autores constatan que mejoraron la línea de base en un 13,41 %, tanto en términos de similaridad visual de los fotogramas futuros, como en términos de *Mean Squared Error*.

***4. A Hybrid CNN-LSTM Model for Aircraft 4D Trajectory Prediction* [41]**

En este trabajo, los autores proponen usar una novedosa arquitectura híbrida de predicción de las trayectorias 4D basada en *Deep Learning*, combinando CNN y LSTM. La convolución 1D se usa para extraer la funcionalidad de dimensión espacial de la trayectoria, y LSTM se usa para extraer la funcionalidad de la dimensión de la trayectoria. Por lo que la predicción de la trayectoria 4D se hace basándose en la fusión de las funcionalidades arriba mencionadas. A su vez, los autores usan datos históricos de la trayectoria con ADS-B y comparan dicho método con un modelo LSTM con el mismo conjunto de datos. Como resultado, los autores constatan que usando el modelo híbrido de CNN-LSTM, la eficacia de la predicción de las trayectorias es superior a usar un modelo único. El error se reduce en un 21,62 % de media, en comparación con el modelo LSTM, y en un 52,45 % comparándolo con el modelo de retropropagación del error.

3.3.2. Discusión

En este apartado se van a comparar los diferentes trabajos sobre los que se ha investigado y realizado el posterior resumen. Las dimensiones comparativas escogidas van a ser: el tipo de estructura de red que se ha escogido, el tipo de entrada, las tecnologías utilizadas, las funciones de pérdida y los optimizadores.

Nº	Estudio	Técnica	Tipo de entrada	Tecnologías usadas	Función de perdida	Optimizador
1	*Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation. Network Speed Prediction*	CNN	Imagen en color	-	MSE	-
2	*Aircraft Prediction using LSTM Neural Network with Embedded Convolutional Layer*	LSTM con capas convolucionales	Imagen binaria	Tensorflow	MSE	Adam
3	*One-Step Time-Dependent Future Video Frame Prediction with a Convolutional Encoder-Decoder Neural Network*	CNN de codificación y CNN de decodificación	Imagen en blanco y negro	-	L2, MSE	Adam
4	*A Hybrid CNN-LSTM Model for Aircraft 4D Trajectory Prediction*	CNN y LSTM	Matriz	Keras, Tensorflow	RMSE, MAE, MAPE	-

Tabla 3.3: Comparativa de trabajos relacionados

Los trabajos estudiados muestran diferentes técnicas y herramientas que pueden ser de gran utilidad a la hora de crear el modelo del presente TFG. Para la realización del mismo, se ha tenido en cuenta toda esta información El modelo de predicción de trayectorias que se quiere crear, será mediante una CNN combinada con LSTM, como en el cuarto trabajo estudiado. El tipo de entrada que se usará en la predicción de trayectorias para este TFG será un conjunto de imágenes en blanco y negro. En cuanto a las tecnologías utilizadas, se está utilizando Keras y Tensorflow para implementar el modelo, junto a otras librerías de Python. La función de pérdida usada es *Binary Cross Entropy*. Por último, el optimizador usado es Adam, al igual que en dos de los trabajos.

Parte II

Desarrollo de la propuesta

Capítulo 4

Descripción y desarrollo de la propuesta

En este capítulo se explicará la propuesta de trabajo de manera detallada. Primeramente se realizará el análisis de requisitos para este proyecto (Sección 4.1), seguido del diseño (Sección 4.2) e implementación (Sección 4.3) del código necesario para satisfacer dicho análisis. Finalmente, se describirán brevemente las pruebas realizadas (Sección 4.4).

4.1. Análisis

Durante esta etapa se van a describir tanto las fuentes de datos como los requisitos del proyecto.

4.1.1. Descripción de fuentes de datos

Para la realización del proyecto se dispone de un *dataset* o conjunto de datos, que más tarde se usarán para implementar el modelo y realizar consecuentemente una serie de pruebas. El *dataset* utilizado incluye datos ADS-B procedentes en su origen de *OpenSky*, procesados y enriquecidos de tal manera que facilite su utilización en base al concepto de trayectoria 4D. Por su parte, *OpenSky* es una fuente de datos ADS-B abierta que proporciona esta información como vectores. En la Tabla 4.1 se van a detallar todos los campos incluidos en cada fila de dicho *dataset*.

En total el *dataset* escogido, *20170109_16vuelos.part*, con un tamaño de 1999 KB, se compone de 60326 registros, pertenecientes a 100 vuelos en un lapso de tiempo de 10 días, del 1 de enero de 2020 al 10 de enero del mismo año. Se tienen 24 atributos.

Las rutas a las que pertenecen las trayectorias de los vuelos son con origen en el aeropuerto Adolfo Suárez Madrid-Barajas, y con destino en distintos aeropuertos de otros países.

4.1.2. Requisitos

Los requisitos previos a poder empezar la fase de diseño son:

- REQ-1: realizar la instalación del entorno adecuado, en este caso se hará uso de los cuadernos de Jupyter.
- REQ-2: cargar y preparar el *dataset* para poder utilizarlo. Haciendo uso de la librería Pandas, se va a cargar el conjunto de datos, *20170109_16vuelos.part*. Este conjunto está en formato Parquet.

Nombre atributo	Tipo	Descripción
fpId	*string*	Identificador del vuelo
icao24	*string*	Dirección usada a bajo nivel de protocolo de radio entre el radar de vigilancia y el transpondedor
callsign	*string*	Identificador único de un avión
latitude	*float*	Latitud de la aeronave en un momento dado del vuelo, en grados
longitude	*float*	Longitud de la aeronave en un momento dado del vuelo, en grados
altitude	*float*	Altitud de la aeronave en un momento dado del vuelo, en metros
speed	*float*	Velocidad horizontal de la aeronave en un momento dado del vuelo, en millas/h
vspeed	*float*	Velocidad vertical de la aeronave en un momento dado del vuelo, en millas/h
ground	*boolean*	Indica si la aeronave está volando o en tierra
track	*float*	Proyección sobre la superficie terrestre de la trayectoria de una aeronave, cuya dirección en cualquier punto suele expresarse en grados respecto al Norte (verdadero, magnético o de cuadrícula).
aerodromeOfDeparture	*string*	Código OACI [84] del aeropuerto de origen
aerodromeOfDestination	*string*	Código OACI del aerpuerto de destino
operator	*string*	Aerolínea que opera el vuelo
aircraftType	*string*	Tipo de modelo de avión
flightDate	*float*	Fecha del vuelo en YYYY-MM-HH
timestamp	*int*	Marca temporal que indica la fecha con precisión
touchdown	*int*	Marca temporal en el momento del aterrizaje de la aeronave
RTA	*int*	Tiempo restante en segundos hasta que la aeronave aterriza en el momento de la obtención del dato
vectorId	*string*	-
reportId	*string*	-

Tabla 4.1: Atributos que conforman el conjunto de datos

- REQ-3: tener en el entorno todas las librerías que se van a utilizar en versiones que sean compatibles entre sí, para que no haya ningún impedimento a la hora de implementar la solución.
- REQ-5: establecer un perímetro cuadrado con centro en el aeropuerto.
- REQ-6: tratar los datos tal cual están en el modelo de datos obtenido, pasándolos a matrices que contengan las trayectorias.
- REQ-7: teniendo un conjunto de matrices de los diferentes rumbos de las aeronaves, entrenar un modelo que posteriormente prediga mediante una imagen, la trayectoria siguiente de un vuelo dado, es decir, que prediga cuáles son los siguientes píxeles que en la imagen

estarán marcados (píxeles por donde el avión pasará).

- REQ-8: evaluar la predicción realizada mediante la utilización de dos métricas.
- REQ-9: guardar el modelo creado.
- REQ-10: cargar el modelo guardado previamente.

4.2. Diseño

En la Figura 4.1 muestra el proceso para la obtención de una matriz a partir de los datos del *dataset*. Primero de todo se seleccionan los atributos más importantes a partir de los cuales se crearán las matrices. Estos son latitud, longitud y *timestamp*. Después con esos campos se crearán las matrices, y a partir de las matrices se crearán las ventanas, que aunarán varias imágenes.

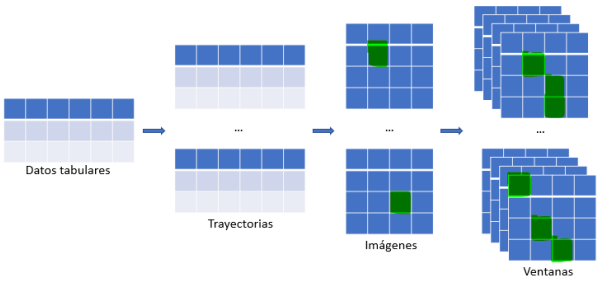


Figura 4.1: Diseño de las matrices a partir de los datos tabulares

En la Figura 4.2 se plantea el proceso para la predicción de trayectorias, desde que se entrena el modelo con los datos de entrenamiento, y luego usando los de validación, hasta que con dicho modelo entrenado, se utilizan los datos de test para ver cómo predice el modelo.

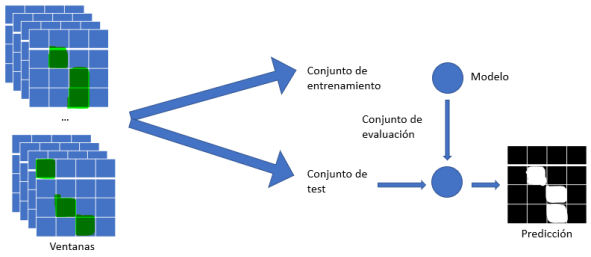


Figura 4.2: Diseño del modelo y posterior predicción

En la Figura 4.3 se puede ver la primera aproximación que se hizo para mostrar gráficamente las trayectorias de las aeronaves.

4.3. Implementación

El proceso que se ha seguido para realizar la predicción de trayectorias es el que sigue, pudiéndose ver resumido en la Figura 4.4:

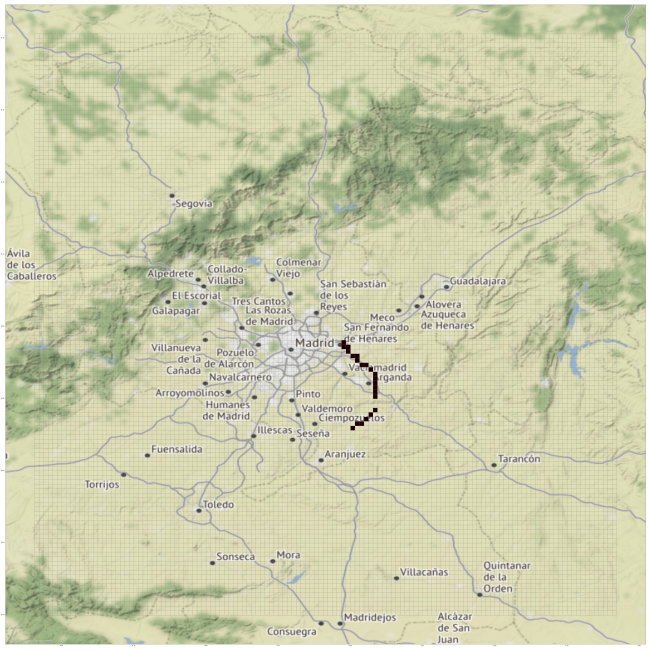


Figura 4.3: Representación de uno de los fotogramas resultantes sobre un mapa

1. Se leen los datos desde el fichero almacenado en el ordenador, el cual está en formato Parquet.

2. Se crea el *dataframe* que contendrá todos los datos extraídos, que se explorarán, de los cuales los más importantes serán *latitude*, *longitude* y *timestamp*.

3. Se crean las matrices que contienen las trayectorias. Esto se realiza a partir de la información de posición de la aeronave en cada tiempo dado (usando la latitud, la longitud y un tiempo determinado). El proceso detallado para crear dichas matrices es el siguiente:

   - Se crea un subconjunto de datos que contenga sólo las columnas de *latitude*, *longitude* y *timestamp*
   - Se definen una serie de variables que serán clave en la creación de las matrices. Básicamente, se quiere establecer un área cuadrada con centro en el aeropuerto, y ver las trayectorias de todos los aviones en un cierto tiempo, que pasan por dicho recuadro.
   - Se van a almacenar las latitudes, longitudes y tiempos tanto mínimos como máximos. Todo esto servirá para establecer las posiciones en la matriz.
   - Se creará un perímetro con centro en el aeropuerto, filtrando todos los vectores, y de todos ellos, las latitudes y longitudes que estén fuera del rango del cuadrado, serán eliminadas, para ahorrar espacio en memoria y tiempo de ejecución.

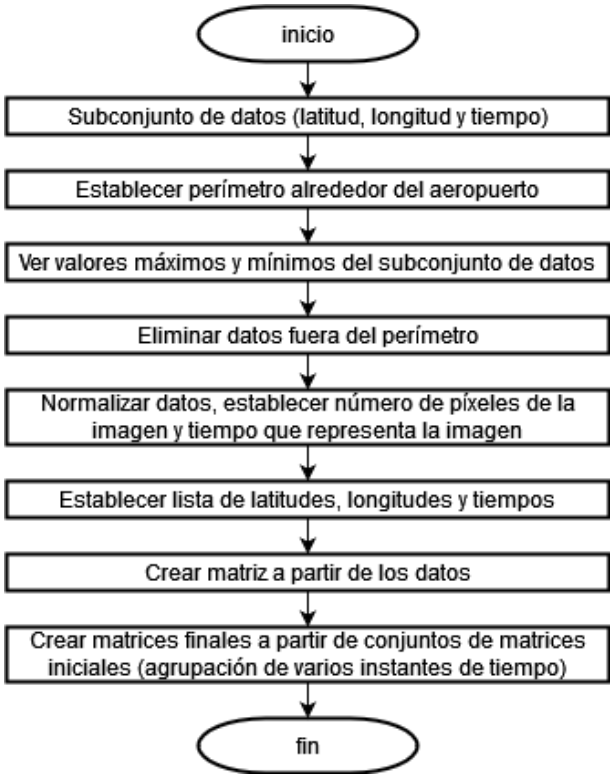


Figura 4.4: Diagrama de flujo de la obtención de la imagen

- Se va a establecer un número de píxeles que tendrán las imágenes posteriormente. Este tamaño por defecto será de 84. Por otro lado se establecerá un *tamanoBucket* de 60. Esta variable representa el tiempo en segundos que se coge para cada imagen, es decir, todo lo que ocurra en 60 segundos, se plasmará en una sola matriz. En este paso, también se hará una normalización de los datos, para agruparlos.
- Se agruparán por latitud, longitud y *timestamp* los datos ya normalizados. De esta manera se crea otro subconjunto de datos ya agrupados.
- De dicho subconjunto, se extraen las latitudes, longitudes y tiempos máximos y mínimos, y se almacenan en variables.
- Se almacenan en variables las latitudes, longitudes y tiempos únicos.
- Se tendrán las latitudes, las longitudes y los tiempos. Las latitudes y longitudes irán desde 0 hasta el número de píxeles que se definió anteriormente, 84. Los tiempos van del tiempo mínimo al máximo.
- Se almacena el tamaño de la lista total de longitudes, de latitudes y de tiempos. En este caso, tanto la longitud como la latitud es de 84, y el número de tiempos es de 13583.
- Se recorren todos los tiempos. Cada tiempo se almacena en una variable. Si esta variable está vacía, quiere decir que la matriz está llena de 0, por lo que esa matriz no se almacena. En caso contrario, se crea una matriz auxiliar. Ahora bien, en esta matriz, se debe marcar a 1 (trayectoria del avión) todas aquellas posiciones para las cuales

se tenga un registro de latitud y longitud. En las posiciones que no haya registros, el valor se dejará en 0 (no pasa el avión).

- Por último,se almacenan las matrices finales con las que se trabajará en el modelo. Se crearán varias variables:
  - *ventana*: será el número de imágenes creadas en el paso anterior que se cogerán. En este caso se cogen 4.
  - *movimientoVentana*: será el salto que se da de una ventana a otra. Se puede imaginar como un carrete con fotogramas de trayectorias. En un primer momento se utilizan los 4 primeros fotogramas, y después se cogen otros 4 fotogramas, pero empezando en el cuarto. De esta manera se consigue que los fotogramas sean dependientes unos de otros.
  - *varAux*: contador que se utiliza para que pare cuando llegue al final de la lista de matrices creadas anteriormente.

  Por último se irán creando las matrices agrupadas. Estas matrices, se rellenarán con la suma de las matrices creadas en el paso anterior. De esta manera, al final se tendrá una serie de matrices agrupadas, que ya están listas para utilizarlas en la predicción de trayectorias.

4. Ahora se crean unas ventanas fijas deslizantes. Se utilizará otra vez una ventana de 4 imágenes (o fotogramas) cada vez. En este caso, las imágenes no son sumadas, puesto que lo que se quiere conseguir es tener una lista con imágenes que estén relacionadas entre ellas, no 4 imágenes fusionadas en 1.

5. A continuación se procede a la carga de los datos y a su división en entrenamiento, validación y prueba. Se cargará cada *bucket*. Se separa en entrenamiento, validación y test. A su vez, se tiene que definir una función para hacer las ventanas fijas deslizantes. Se crearán 2 variables, una que contenga las matrices de 0 a n - 1, y otra que contenga las matrices de 1 a n. De esta manera, se crea una temporalidad.

6. Se construye el modelo.

7. Se entrena el modelo, utilizando los datos de entrenamiento junto con los de validación. Se entrenará al modelo por defecto durante un total de 10 épocas.

8. Se realizan las predicciones con el conjunto de test.

9. Se visualizan los fotogramas tanto de las predicciones como de los originales.

10. Se calculan las las métricas definidas para evaluar la eficacia del modelo.

4.4. Pruebas

En este apartado se visualizan las imágenes para verificar los resultados y comprobar que las trayectorias han sido creadas correctamente.

Parte III

Resultados

Capítulo 5

Experimentación y evaluación

En este capítulo se explica la experimentación con los diferentes modelos y la posterior evaluación para establecer cuál produce mejores resultados en cuanto a la predicción de trayectorias de aeronaves.

5.1. Diseño experimental

5.1.1. Métricas

Se han utilizado 2 métricas para evaluar el modelo:

- *Accuracy* del modelo: representa el porcentaje total de valores correctamente clasificados. La fórmula es la que sigue a continuación:

$$Acierto = \frac{Clasificados\ Correctamente}{Todos\ Los\ Ejemplos}$$

- Métricas *ad hoc*: consiste en calcular *accuracy*, *precision*, *recall* y F1 de de cada predicción en comparación con la imagen real. A continuación se detallan las fórmulas y la definición de cada una de las métricas [38]:

  - Acierto (*Accuracy*): porcentaje total de valores correctamente clasificados, tanto positivos como negativos.

$$Acierto = \frac{Clasificados\ Correctamente}{Todos\ Los\ Ejemplos}$$

  - Precisión (*Precision*): qué porcentaje de valores que se han clasificado como positivos lo son realmente.

$$Precisión = \frac{Clasificados\ Correctamente\ Como\ Positivos}{Todos\ Los\ Ejemplos\ Clasificados\ Como\ Positivos}$$

  - *Recall*: cuántos valores positivos son correctamente clasificados.

$$Recall = \frac{Clasificados\ Correctamente\ Como\ Positivos}{Todos\ Los\ Verdaderamente\ Positivos}$$

- F1: combinación de precisión y *recall*. Se usa cuando el conjunto de datos a analizar está desbalanceado, obteniendo un valor más objetivo.

$$F1 = 2 \times \frac{Precisión \times Recall}{Precisión + Recall}$$

Todas estas métricas requieren para su utilización la declaración de un parámetro llamado *umbral*, debido a que se tiene que binarizar la imagen para poder ejecutar la métrica, puesto que la salida que se obtiene del modelo no son imágenes estrictamente binarias, sino que se tienen valores en punto flotante. En todas las métricas se ha escogido un umbral de 0.45, después de experimentar con varios umbrales para ver cuál es la mejor opción.

5.1.2. Datos de prueba

Los datos de prueba que se han utilizado son un conjunto de matrices de trayectorias de vuelos, es decir, un subconjunto de los ejemplos (ventanas construidas), que equivale a un 20 % del total de ejemplos, no vistos durante el entrenamiento.

5.1.3. Arquitecturas

Con respecto a la arquitectura del modelo, se han probado diferentes configuraciones, explicadas en la Tabla 5.1, para, de esta manera, hallar la configuración que arroje el mejor resultado.

Capa	Parámetro	Modelo 1	Modelo 2	Modelo 3
ConvLSTM2D	Nº filtros	4	4	4
	Tamaño *kernel*	(5,5)	(5,5)	(5,5)
	Función activación	sigmoide	sigmoide	sigmoide
Batch Normalization	-	Sí	Sí	Sí
ConvLSTM2D	Nº filtros	-	4	4
	Tamaño *kernel*	-	(5,5)	(5,5)
	Función activación	-	sigmoide	sigmoide
Batch Normalization	-	No	Sí	Sí
ConvLSTM2D	Nº filtros	-	-	4
	Tamaño *kernel*	-	-	(5,5)
	Función activación	-	-	sigmoide
Conv3D	Nº filtros	1	1	1
	Tamaño *kernel*	(3,3,1)	(3,3,1)	(3,3,1)
	Función activación	ReLU	ReLU	ReLU

Tabla 5.1: Arquitectura de los modelos considerados

5.1.4. Proceso de evaluación

A la hora de entrenar el modelo, se ha evaluado mediante 2 métricas: la primera métrica es*accuracy* del modelo. A su vez, se tienen 3 conjuntos, uno de entrenamiento, otro de validación

	Modelo 1	Modelo 2	Modelo 3
Accuracy (10 épocas)	0.9996	0.9994	0.9984

Tabla 5.2: *Accuracy* de los 3 modelos con un entrenamiento de 10 épocas

y un último conjunto de prueba. Para entrenar se han utilizado el conjunto de entrenamiento y el de validación. Después de obtener un modelo ya entrenado, se procede a realizar una predicción que tiene como entrada el conjunto de prueba. Una vez realizada la predicción, se procede a visualizar la misma, para ver cómo de bien el modelo es capaz de predecir las trayectorias de las aeronaves. La segunda métrica es la métrica *ad hoc*, explicada en la Sección 5.1.1 Las características más importantes de los modelos, los cuales se pueden observar en arquitecturas la Tabla 5.1 son la función de activación, siendo sigmoide en la capa ConvLSTM2D y ReLU en la Conv3D. Las capas van variando según aumente la complejidad del modelo, es decir, cuantas más capas tenga el modelo, más complejo será. El modelo más sencillo consta de una capa ConvLSTM2D y otra Conv3D; el de complejidad intermedia añade una capa ConvLSTM2D, y el último y más complejo añade una capa ConvLSTM2D más.

5.2. Experimentación y resultados

En un principio se hicieron las pruebas con un modelo simple, que se entrenó durante 10 épocas. Más tarde se probó con modelos más complejos a los cuales se les fueron añadiendo más capas. En la Tabla 5.2 se pueden ver los resultados que arrojaron los diferentes modelos con 10 épocas:

Algunas de las predicciones de los tres modelos se pueden ver en las Figuras 5.1, 5.3 y 5.5, así como las trayectorias reales en las Figuras 5.2, 5.4 y 5.6.

Modelo 1 (Simple):

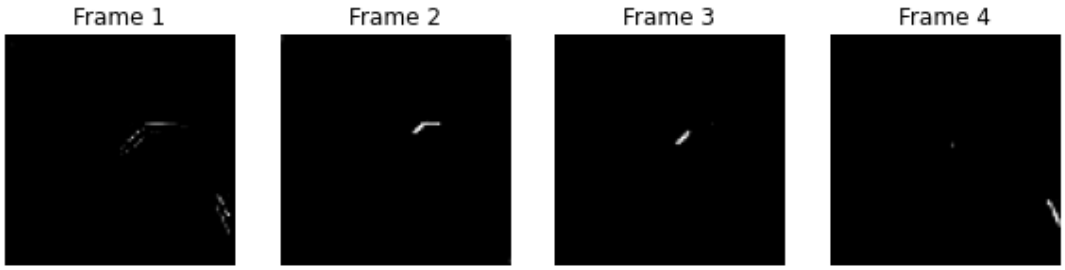


Figura 5.1: Predicciones modelo simple

Modelo 2 (Intermedio):

Modelo 3 (Complejo):

Como se puede observar, entre el modelo 1 y el modelo 2 no existen muchas diferencias, ambos predicen muy bien. Sin embargo, el modelo 3, predice claramente peor que los otros dos. Puede que trate de sobreajustar y de esta manera hace que las predicciones no sean tan buenas como se debiera.

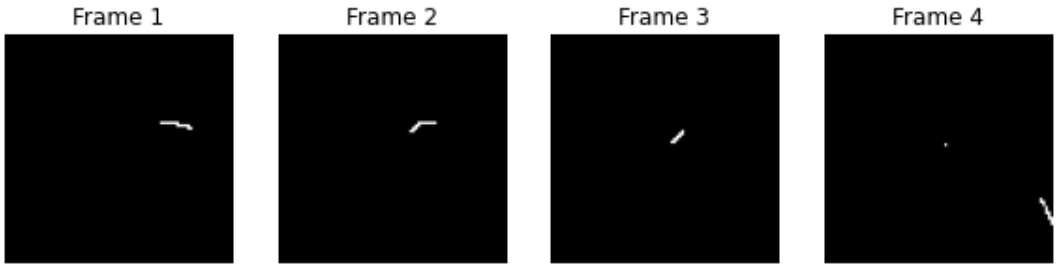


Figura 5.2: Trayectorias reales modelo simple

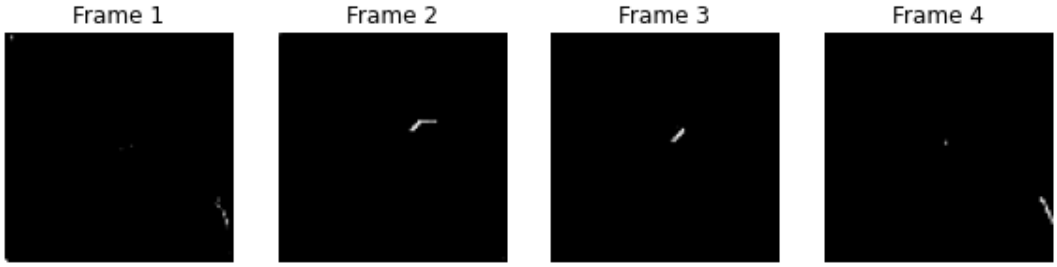


Figura 5.3: Predicciones modelo intermedio

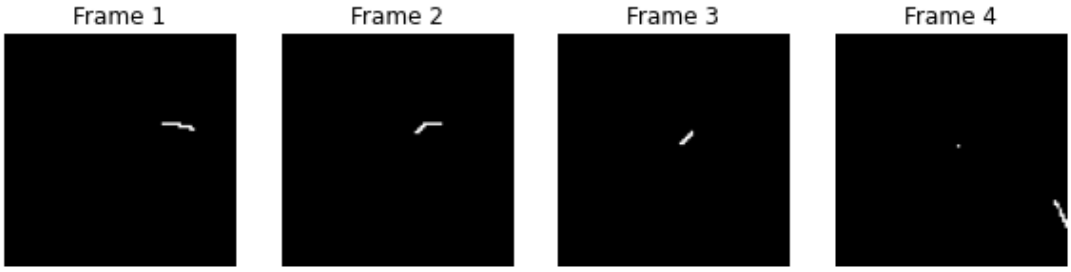


Figura 5.4: Trayectorias reales modelo intermedio

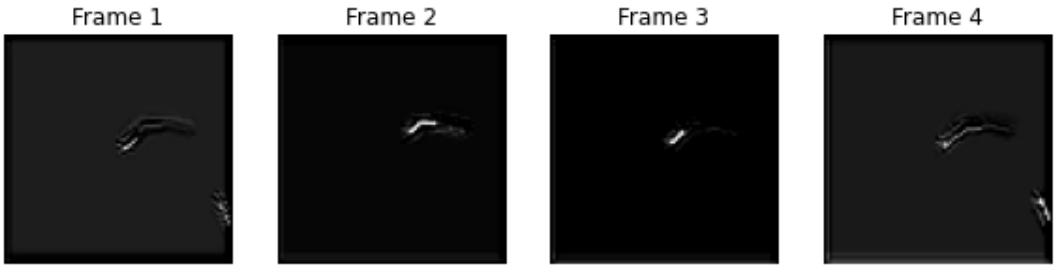


Figura 5.5: Predicciones modelo complejo

Después, mediante la métrica creada *ad hoc*, citada en 5.1.1, se vieron *accuracy*, *precision*, *recall* y F1 de las predicciones una a una. Los resultados son los expuestos en la Figura 5.7, con una media de acierto en todas las métricas de aproximadamente 0.98. Estas predicciones

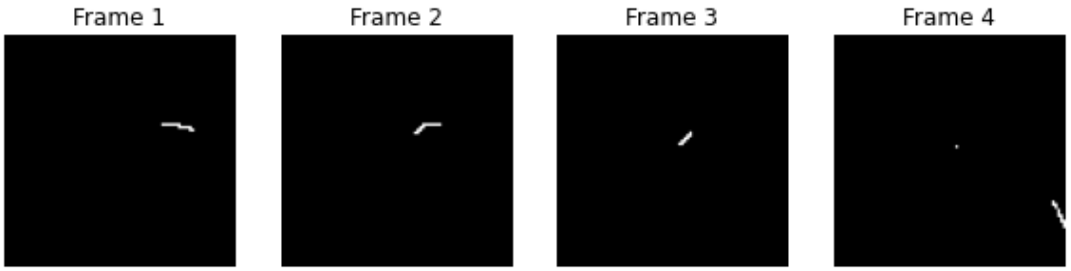


Figura 5.6: Trayectorias reales modelo complejo

pertenecen al modelo más sencillo de todos. En la Figura 5.2 se pueden ver las trayectorias reales del avión, y en la Figura 5.1, se pueden observar las predicciones sobre el conjunto de prueba, con el modelo entrenado previamente con los conjuntos de entrenamiento y de validación.

	Accuracy	Precision	Recall	F1
count	111.000000	111.000000	111.000000	111.000000
mean	0.999971	0.997667	0.982874	0.989523
std	0.000079	0.018987	0.046708	0.027635
min	0.999575	0.812500	0.750000	0.857143
25%	1.000000	1.000000	1.000000	1.000000
50%	1.000000	1.000000	1.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000

Figura 5.7: Resumen métrica

5.3. Discusión de resultados

Al aumentar la complejidad de los modelos, el *accuracy* del modelo era muy similar, por lo que el mejor modelo de los 3 es el primero, que a su vez es el más sencillo de todos. Esto se debe a que el modelo más simple requiere aproximadamente el mismo tiempo que los demás y menor coste de computación. El modelo más sencillo tardaba en entrenar 16.50 minutos, el intermedio 16.56 minutos, y el más complejo de todos tardaba 15.81 minutos. Todos ellos fueron probados durante diferentes épocas. Estos tiempos pertenecen al entrenamiento durante 10 épocas. Se hicieron pruebas con 5 épocas, con 10, con 15 y con 20 épocas. Sin embargo, los resultados eran muy similares, y aumentaba de manera importante el tiempo de computación, por lo que se decidió optar por el modelo más simple de todos durante 10 épocas.

El optimizador utilizado para el entrenamiento del modelo fue Adam (*Adaptive Moment Estimation*).

Las funciones de activación usadas fueron ReLU y sigmoide. Al principio se usó sólo la función de activación sigmoide, pero las imágenes no daban resultados en blanco y negro, por lo que después se probó con ReLU, y finalmente se utilizó una combinación de ambas, así obteniendo como salida una predicción en blanco y negro.

Estos resultados son muy precisos para el *dataset* del que se dispone, con datos del 1 de enero de 2020 al 10 de enero del mismo año. Sin embargo, sería muy interesante introducir más datos y ver el comportamiento del modelo para una cantidad de información mayor. De esta manera, es probable que la precisión del modelo fuera diferente. En cuanto al tamaño de la ventana elegido para el modelo, finalmente de 4, a la resolución de las imágenes, de 84x84 píxeles y al tamaño del *bucket* (tiempo plasmado en cada imagen, de 60 segundos), son valores que se consideran adecuados finalmente para realizar el posterior entrenamiento del modelo.

Sin embargo, en un principio fueron considerados otros valores para dichas variables, los cuales se relatan a continuación:

En un principio, se escogió un tamaño de imagen de 1024x1024, pero resultó imposible utilizarlo puesto que las imágenes ocupaban demasiado y los cálculos con las matrices eran mucho más arduos. Después se fue reduciendo el tamaño de las imágenes hasta llegar al tamaño actual, de 84x84, reduciendo considerablemente los tiempos de computación y los costes.

Con respecto al tamaño de ventana elegido para el modelo, se fueron probando varios, en un principio más pequeños y luego aumentando el tamaño. Finalmente se dejó en el tamaño actual de 4, puesto que recogía la información necesaria.

En relación al tamaño del *bucket*, se escogieron 5 minutos, pero luego se vio que era demasiado tiempo y no arrojaba unos resultados buenos. Más tarde se fue probando con otros tiempos, hasta llegar al actual, de 60 segundos, que recoge los suficientes sucesos para que se pueda entrenar al modelo con unos datos de calidad.

Por último, como se puede ver en las Figuras 5.1 y 5.2, se están prediciendo los fotogramas inmediatamente futuros, con un salto en el tiempo de 1, pero se podrían predecir de igual manera fotogramas a n tiempo futuro.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Después de haber finalizado el proyecto, se puede afirmar que los objetivos planteados en el inicio del proyecto se han cumplido satisfactoriamente. A continuación se va a realizar un breve resumen de cómo estos objetivos han sido alcanzados:

- El primer objetivo era estudiar el entorno de negocio relacionado con la gestión del tráfico aéreo analizando el aeropuerto Adolfo Suárez Madrid-Barajas. Además, se estudiaron las tecnologías existentes a día de hoy en relación con ATM. Este objetivo se puede ver completado en el Capítulo 1 y Capítulo 3
- El segundo objetivo era realizar un análisis de las características y el funcionamiento del Aprendizaje Automático y del Aprendizaje Profundo, además de ver sus algoritmos principales . Este objetivo se puede ver completado en el Capítulo 3
- En el tercer objetivo, mediante una serie de datos, utilizando algoritmos de Aprendizaje profundo, se creó un modelo para predecir las trayectorias de los vuelos. A partir de la obtención de un primer modelo sencillo, se fueron probando varios modelos más, para estudiar cuál de las configuraciones del modelo era la más adecuada a la hora de predecir. Posteriormente se hizo una comparación entre los modelos y se determinó que el modelo más sencillo era el que mejor predecía. Este objetivo queda realizado en el Capítulo 4

6.1.1. Perspectiva del proyecto

A continuación se va a valorar la perspectiva del proyecto.

- Objetivos específicos: todos los objetivos han sido completados satisfactoriamente. Se ha investigado sobre ATM, ML, DL, se han visto los algoritmos principales. Por último, se ha realizado un modelo que permite crear predicciones sobre las próximas trayectorias de las aeronaves.
- Utilidad para el futuro: este proyecto puede resultar de gran utilidad puesto que establece una manera de predecir las trayectorias que van a tomar las aeronaves, por lo que se podrán evitar en gran medida los conflictos que pudieran surgir entre 2 aeronaves. De esta manera, en un futuro, aplicando estas técnicas, los vuelos serán más seguros, se ahorrará combustible y los tiempos de llegada serán mejores.

- Desarrollo del proyecto y metodología de trabajo: El proyecto se ha desarrollado siguiendo la metodología de trabajo UVagile, descrita en el Capítulo 2. Gracias al empleo de dicha metodología de trabajo, se ha conseguido llegar al resultado final, de manera satisfactoria. A su vez, se ha conseguido realizar un uso efectivo del tiempo, que, aun habiendo surgido problemas y cuellos de botella, gracias a UVagile se han conseguido subsanar mediante las reuniones semanales, los canales destinados a la comunicación, y varias tutorías.

6.1.2. Perspectiva personal

La realización del presente TFG ha supuesto un aprendizaje muy grande, por varias razones:

- Gracias a la investigación, se ha conseguido llegar a un entendimiento muy grande de ATM y del aeropuerto de Barajas. A su vez, se han adquirido muchos conocimientos de Aprendizaje Automático y Aprendizaje Profundo.
- Gracias a UVagile, se ha gestionado el tiempo de una manera más eficaz.
- Mediante Teams, se ha conseguido lograr una comunicación muy efectiva.
- Ha habido un grado de aprendizaje muy alto del lenguaje Python.
- Se han mejorado mucho las habilidades con Python y con LaTeX
- He crecido como estudiante y persona.
- Ser resiliente y, ante un problema, buscar la solución hasta encontrarla, no importando cuánto tiempo o esfuerzo cueste, porque al final siempre acaba saliendo el trabajo con dedicación y esfuerzo.

6.2. Trabajo futuro

Debido a las restricciones de tiempo, no se han podido realizar otras mejoras del proyecto, pero podrían ser realizadas en un futuro, y aportarían más información a la hora de realizar las predicciones de las trayectorias de las aeronaves:

- Ahora mismo se tienen imágenes binarias de las trayectorias, que indican por dónde irán las aeronaves. Ahora bien, se podría establecer un código de colores, es decir, una escala de grises, en la que se mostrara la altitud de la aeronave, cuanto más tenue el gris, más altitud y viceversa. Sería de gran utilidad para saber además de la trayectoria, saber a qué altura pasa la aeronave y así poder evitar conflictos.
- Otra idea a realizar en un futuro sería añadir un fondo en el cual se viera el mapa de la zona por donde la aeronave se supone que va a pasar.
- Probar más configuraciones en el modelo para buscar soluciones más efectivas.
- Incorporar datos de más vuelos y más días para que así la red entrene con más ejemplos y sea todavía más efectiva.

Parte IV

Apéndices

Apéndice A

Contenido adjunto

En esta sección se va a describir el contenido adjunto al presente TFG. Uno de los archivos adjunto contiene el programa en formato .ipynb, es decir, cuaderno de *Jupyter*, necesario para realizar todas las operaciones, desde la transformación de los datos a matrices, hasta la predicción de las trayectorias de aeronaves. A su vez, está contenido el fichero en formato Parquet con el *dataset* completo para que el cuaderno pueda ser ejecutado correctamente. Se adjuntan también los diferentes modelos con los que se ha experimentado, en concreto tres, llamados *ModeloConvLSTM.h5*, (el más sencillo), *ModeloConvLSTMComplejidadIntermedia.h5* (el de complejidad media) y *ModeloConvLSTMComplejidadAlta.h5* (el de complejidad más alta). Por último, se incluye la memoria del TFG. A continuación se detalla el árbol de directorios y de programas adjuntos:

```
+-- TFGIrenePenasPerez
     |
     |
     +----Código
     |      CodigoTFGIrenePenasPerez.ipynb
     |
     +----Datos
     |      sample_data.parquet
     |
     +----Memoria
     |      TFGIrenePenasPerez.pdf
     |
     +----Modelos
            ModeloConvLSTM.h5
            ModeloConvLSTMComplejidadIntermedia.h5
            ModeloConvLSTMComplejidadAlta.h5
```

Apéndice B

Siglas y acrónimos

A continuación se van a exponer todas las Siglas y acrónimos que han sido utilizadas durante la redacción de la memoria del TFG.

- Adam: Adaptative Moment Estimation (Estimación del Momento Adaptativo).
- ADS-B: Automatic Dependent Surveillance - Broadcast (Transmisión de Vigilancia Dependiente Automática).
- AI: Artificial Intelligence (Inteñligencia artificial).
- ASM: Airspace Management (Gestión del Espacio Aéreo).
- ATC: Air Traffic Control (Control del Tráfico Aéreo).
- ATFM: Air Traffic Flow Management (Gestión del flujo del Tráfico Aéreo).
- ATM: Air Traffic Management (Gestión del Tráfico Aéreo).
- ATS: Air Traffic Services (Servicio de Tráfico Aéreo).
- BPTT: Backpropagation Through Time (Retropropagación a través del tiempo)
- CNN: Convolutional Neural Networks (Redes neuronales convolucionales).
- ConvLSTM2D (Redes Neuronales Convolucionales combinadas con LSTM 2D).
- Conv3D (Red neuronal convolucional 3D)
- DL: Deep Learning (Aprendizaje profundo).
- LSTM: Long Short-Term Memory Networks (Redes de memoria a corto y largo plazo).
- MAE: Mean Absolute Error (Error Absoluto Medio).
- MAPE: Mean Absolute Percentage Error (Error Porcentual Absoluto Medio).
- ML: Machine Learning (Aprendizaje Automático).
- ReLU: Rectified Linear Unit (Unidad lineal rectificada).

- RMSE: Root Mean Squared Error (Error Cuadrático Medio).
- RNN: Recurrent Neural Networks (Redes neuronales recurrentes).
- RRNN (Redes neuronales)
- TMA: Terminal Control Area (Área de control terminal)

Bibliografía

[1] FEDERAL AVIATION ADMINISTRATION. «ADS-B In Trail Procedures (ITP)». En: (2021 Consultado el 20 de marzo, 2022). URL: https://www.faa.gov/nextgen/programs/adsb/pilot/itp.

[2] AEGIS. «How to Learn Machine Learning in Three months and advance your IT Career?» En: (2022. Consultado el 12 de septiembre, 2022). URL: https://www.aegissofttech.com/articles/learn-machine-learning.html.

[3] IVAO AERO. «Tipos de separaciones». En: (2022. Consultado el 11 de abril, 2022). URL: https://es.ivao.aero/?module=assets&page=infoATC&id=59.

[4] SKYBRARY AERO. «Air Traffic Management (ATM)». En: (2022 Consultado el 20 de marzo, 2022). URL: https://skybrary.aero/articles/air-traffic-management-atm.

[5] MARTIN STROHMEIER ET AL. «Crowdsourced air traffic data from the OpenSky Network 2019–2020». En: (2022. Consultado el 1 de mayo, 2022). URL: https://essd.copernicus.org/articles/13/357/2021/.

[6] MAURICIO ANDERSON. «Curso de Latex Referencias y Bibliografía». En: (2022. Consultado el 10 de abril, 2022). URL: https://mauricioanderson.com/curso-latex-referencias-bibliografia-bibtex.

[7] DANI ARRIBAS-BEL. «Introduction guide to contextily». En: (2022. Consultado el 15 de mayo, 2022). URL: https://contextily.readthedocs.io/en/latest/intro_guide.html.

[8] AVIACIONDIGIT@Ll. «La Comisión Europea amplia el plazo para el cumplimiento del ADS-B». En: (2020. Consultado el 3 de abril, 2022). URL: https://aviaciondigital.com/la-comision-europea-amplia-el-plazo-para-el-cumplimiento-del-ads-b.

[9] EL VUELO DE LA GRAN AVUTARDA. «Gestión del tráfico aéreo (ATM) de forma muy simplificada». En: (2018. Consultado el 1 de junio, 2022). URL: https://greatbustardsflight.blogspot.com/2018/06/gestion-del-trafico-aereo-atm-de-forma.html.

[10] SCRUM MANAGER BOOK. «Tarea». En: (2021. Consultado el 11 de abril, 2022). URL: https://www.scrummanager.net/bok/index.php?title=Tarea.

[11] ANÍBAL BREGÓN BREGÓN. «Apuntes Sistemas Inteligentes». En: (2022. Consultado el 6 de junio, 2022). URL: https://campusvirtual.uva.es/.

[12] JAMES BRENNAN. «Fast and easy gridding of point data with geopandas». En: (2022. Consultado el 15 de mayo, 2022). URL: https://james-brennan.github.io/posts/fast_gridding_geopandas/.

[13] JASON BROWNLEE. «A Gentle Introduction to RNN Unrolling». En: (2017. Consultado el 26 de marzo, 2022). URL: https://machinelearningmastery.com/rnn-unrolling.

[14] DIEGO CALVO. «Red Neuronal Recurrente – RNN». En: (2018. Consultado el 26 de marzo, 2022). URL: https://www.diegocalvo.es/red-neuronal-recurrente.

[15] AKSHAY L CHANDRA. «McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron». En: (2018. Consultado el 26 de marzo, 2022). URL: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1.

[16] CODEZUP. «What is the fastai library and What we can do with it?» En: (2021. Consultado el 16 de abril, 2022). URL: https://codezup.com/what-is-fastai-library-and-what-i-can-do-with-it/.

[17] COLAH. «Understanding LSTM Networks». En: (2015. Consultado el 26 de marzo, 2022). URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs.

[18] ComoFuncionaADSB. «Cómo funciona el ADS – B. La tecnología que viene ya está aquí.» En: (2018. Consultado el 3 de abril, 2022). URL: http://www.aviacionglobal.com/articulos-tecnicos-de-aviacion/como-funciona-el-ads-b-la-tecnologia-que-viene-ya-esta-aqui.

[19] DANIEL. «Cómo funciona el ADS – B. La tecnología que viene ya está aquí.» En: (2018 Consultado el 20 de marzo, 2022). URL: http://www.aviacionglobal.com/articulos-tecnicos-de-aviacion/como-funciona-el-ads-b-la-tecnologia-que-viene-ya-esta-aqui.

[20] DELOITTE. «Artefactos Scrum: las 3 herramientas clave de gestión». En: (2022. Consultado el 11 de abril, 2022). URL: https://www2.deloitte.com/es/es/pages/technology/articles/artefactos-scrum.html.

[21] REAL ACADEMIA ESPAÑOLA. «predecir». En: (2022. Consultado el 10 de abril, 2022). URL: https://dle.rae.es/predecir.

[22] SARAHÍ SILVA ESTEFANÍA FREIRE. «Redes Neuronales». En: (2019. Consultado el 26 de marzo, 2022). URL: https://bootcampai.medium.com/redes-neuronales-13349dd1a5bb.

[23] EUROCONTROL. «Trajectory Prediction». En: (2020. Consultado el 26 de marzo, 2022). URL: https://www.eurocontrol.int/phare/public/standard_page/TrajPred.html.

[24] CRAIG FREUDENRICH. «How Air Traffic Control Works». En: (2021 Consultado el 20 de marzo, 2022). URL: https://science.howstuffworks.com/transport/flight/modern/air-traffic-control.htm.

[25] FIDEL GALLA. «Konsep Convolution Neural Network (CNN)». En: (2020. Consultado el 26 de marzo, 2022). URL: https://iglab.tech/konsep-convolutional-neural-network-cnn.

[26] FIDEL GALLA. «Konsep Convolution Neural Network (CNN)». En: (2020. Consultado el 26 de marzo, 2022). URL: https://iglab.tech/konsep-convolutional-neural-network-cnn.

[27] GEOPANDAS. «Adding a background map to plots». En: (2022. Consultado el 15 de mayo, 2022). URL: https://geopandas.org/en/stable/gallery/plotting_basemap_background.html.

[28] GEOPANDAS. «Creating a legend». En: (2022. Consultado el 15 de mayo, 2022). URL: https://geopandas.org/en/stable/docs/user_guide/mapping.html#creating-a-legend.

[29] LAUREN WASHINGTON GEORFE MCINTIRE BRENDAN MARTIN. «Python Pandas Tutorial: A Complete Introduction for Beginners». En: (2022. Consultado el 8 de mayo, 2022). URL: https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/.

[30] JUAN CARLOS GONZÁLEZ. «Siguiendo la pista a un avión: qué tecnologías se usan y qué problemas hay». En: (2015 Consultado el 19 de marzo, 2022). URL: https://www.xataka.com/otros/siguiendo-la-pista-a-un-avion-que-tecnologias-se-usan-y-que-problemas-hay.

[31] TANISH GUPTA. «Fastest way to install Geopandas in jupyter notebook on Windows». En: (2022. Consultado el 13 de mayo, 2022). URL: https://medium.com/analytics-vidhya/fastest-way-to-install-geopandas-in-jupyter-notebook-on-windows-8f734e11fa2b.

[32] ROBERTO GÓMEZ. «Navegación 4D – a lo que apunta el mundo civilizado.» En: (2012 Consultado el 19 de marzo, 2022). URL: https://flap152.com/2014/02/13/navegacion-4d-lo-que-apunta-el-mundo.

[33] JOSÉ MARTÍNEZ HERAS. «15 Librerías de Python para Machine Learning». En: (2020. Consultado el 26 de marzo, 2022). URL: https://www.iartificial.net/librerias-de-python-para-machine-learning/#Librerias_de_Python_para_Deep_Learning.

[34] JOSÉ MARTÍNEZ HERAS. «¿Clasificación o Regresión?» En: (2020. Consultado el 26 de marzo, 2022). URL: https://www.iartificial.net/clasificacion-o-regresion.

[35] IBIBLIO. «2.1. La neurona artificial». En: (2022. Consultado el 15 de abril, 2022). URL: https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x38.html.

[36] IBM. «Conceptos básicos (redes neuronales)». En: (2021. Consultado el 26 de marzo, 2022). URL: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-basics-neural.

[37] NICOLÁS LARENAS. «El sistema radar de vigilancia de los Servicios de Tránsito Aéreo». En: (2019 Consultado el 20 de marzo, 2022). URL: https://www.nlarenas.com/2019/02/el-sistema-radar-de-vigilancia-de-los-servicios-de-transito-aereo.

[38] THE MACHINE LEARNERS. «Métricas de Clasificación». En: (2022. Consultado el 15 de septiembre, 2022). URL: https://www.themachinelearners.com/metricas-de-clasificacion/#Principales_Meacutetricas_de_clasificacioacuten.

[39] OCTAVIO LEVY. «El Product Owner». En: (2020 Consultado el 21 de marzo, 2022). URL: https://ittude.com.ar/b/scrum/product-owner.

[40] LUIS. «Bibliografía en LaTeX». En: (2011. Consultado el 13 de abril, 2022). URL: http://minisconlatex.blogspot.com/2011/03/bibliografia.html.

[41] LAN MA y SHAN TIAN. «A Hybrid CNN-LSTM Model for Aircraft 4D Trajectory Prediction». En: (2020. Consultado el 2 de abril, 2022).

[42] SCRUM MANAGER. «Estimación de póquer». En: (2021. Consultado el 7 de abril, 2022). URL: https://www.scrummanager.net/bok/index.php?title=EstimaciC3%B3n_de_p%C3%B3quer.

[43] MIGUEL A. MARTÍNEZ-PRIETO, JORGE SILVESTRE y ANIBAL BREGON. «Metodología UVAgile». En: (2019. Consultado el 26 de marzo, 2022). URL: https://uvadoc.uva.es/bitstream/handle/10324/42479/UVagile-Alumnos.pdf?sequence=7&isAllowed=y.

[44] MIGUEL A. MARTÍNEZ-PRIETO y col. «Agilizando el aprendizaje de bases de datos». En: *Actas de las JENUI* 6 (2021), págs. 83-90.

[45] MATPLOTLIB. «Choosing Colormaps in Matplotlib». En: (2022. Consultado el 15 de mayo, 2022). URL: https://matplotlib.org/3.5.0/tutorials/colors/colormaps.html.

[46] MATTHEW MAYO. «Gestión de flujos de trabajo de machine learning con pipelines de Scikit-Learn Parte 1: Una introducción amable». En: (2022. Consultado el 2 de mayo, 2022). URL: https://medium.com/datos-y-ciencia/gesti%C3%B3n-de-flujos-de-trabajo-de-machine-learning-con-pipelines-de-scikit-learn-parte-1-una-8a37fcc7c1d0.

[47] SERVICIOS A LA NAVEGACIÓN EN EL ESPACIO AÉREO MEXICANO. «Sistema de Radar». En: (2022 Consultado el 19 de marzo, 2022). URL: https://www.gob.mx/seneam/acciones-y-programas/sistema-de-radar.

[48] JAVIER MONTERO. «LaTeX – Capítulo 24: Subíndices y superíndices». En: (2012. Consultado el 11 de abril, 2022). URL: http://elclubdelautodidacta.es/wp/2012/04/latex-capitulo-24-subindices-y-superindices/.

[49] BBC MUNDO. «Por qué los pilotos escogerán su propia ruta aérea». En: (2014. Consultado el 11 de abril, 2022). URL: https://www.bbc.com/mundo/noticias/2014/12/141218_tecnologia_aviacion_free_routing_futuro_elegir_rutas_ig.

[50] NA8. «¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador». En: (2018. Consultado el 26 de marzo, 2022). URL: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador.

[51] THE OPEN SKY NETWORK. «The Open Sky Network». En: (2011. Consultado el 20 de marzo, 2022). URL: https://opensky-network.org/.

[52] JORGE ONTIVEROS. «Free flight, ¿la navegación aérea que viene?» En: (2013. Consultado el 13 de abril, 2022). URL: https://www.hispaviacion.es/free-flight-la-navegacion-aerea-que-viene-2/.

[53] ABIGAIL ORUS. «Tráfico total de pasajeros de avión en España 2012-2021». En: (2022. Consultado el 2 de abril, 2022). URL: https://es.statista.com/estadisticas/540927/trafico-total-de-pasajeros-de-avion-en-espana/.

[54] PANDAS. «pandas.read_parquet». En: (2022. Consultado el 4 de mayo, 2022). URL: https://pandas.pydata.org/docs/reference/api/pandas.read_parquet.html.

[55] YUTIAN PANG, NAN XU y YONGMING LIU. «Aircraft Trajectory Prediction using LSTM Neural Network with Embedded Convolutional Layer». En: (2019. Consultado el 2 de abril, 2022).

[56] JUAN DIEGO POLO. «Iridium vs Inmarsat ¿Qué servicio de telefonía satelital es mejor?» En: (2022 Consultado el 20 de marzo, 2022). URL: https://wwwhatsnew.com/2022/03/07/iridium-vs-inmarsat-que-servicio-de-telefonia-satelital-es-mejor.

[57] JONATHAN QUIZA. «Pipeline Python». En: (2022. Consultado el 2 de mayo, 2022). URL: https://medium.com/datos-y-ciencia/pipeline-python-20c84e255444.

[58] MAX REHKOPF. «Epics, historias, temas e iniciativas». En: (2022. Consultado el 11 de abril, 2022). URL: https://www.atlassian.com/es/agile/project-management/epics-stories-themes.

[59] JOAQUÍN AMAT RODRIGO. «Machine learning con Python y Scikit-learn». En: (2022. Consultado el 7 de mayo, 2022). URL: https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn.html.

[60] MEHREEN SAEED. «Modeling Pipeline Optimization With scikit-learn». En: (2022. Consultado el 8 de mayo, 2022). URL: https://machinelearningmastery.com/modeling-pipeline-optimization-with-scikit-learn/.

[61] SCRUM. «The Scrum Guide». En: (2022. Consultado el 11 de abril, 2022). URL: https://www.scrum.org/resources/scrum-guide.

[62] MATLAB y SIMULINK. «Redes neuronales convolucionales». En: (2022. Consultado el 10 de abril, 2022). URL: https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[63] SKYBRARY. «Aircraft Call-sign». En: (2022. Consultado el 7 de mayo, 2022). URL: https://skybrary.aero/articles/aircraft-call-signl.

[64] SKYBRARY. «Heading, Track and Radial». En: (2022. Consultado el 8 de mayo, 2022). URL: https://skybrary.aero/articles/heading-track-and-radial.

[65] STACKOVERFLOW. «Error installing geopandas:.ª GDAL API version must be specified ïn Anaconda». En: (2022. Consultado el 13 de mayo, 2022). URL: https://stackoverflow.com/questions/54734667/error-installing-geopandas-a-gdal-api-version-must-be-specified-in-anaconda.

[66] STACKOVERFLOW. «How to filter some data by read_parquet() in pandas?» En: (2022. Consultado el 4 de mayo, 2022). URL: https://stackoverflow.com/questions/51233436/how-to-filter-some-data-by-read-parquet-in-pandas.

[67] STACKOVERFLOW. «Pipeline Python». En: (2022. Consultado el 4 de mayo, 2022). URL: https://stackoverflow.com/questions/50760351/how-to-identify-pandas-backend-for-parquet.

[68] STACKOVERFLOW. «Proper reuse of Axes in GeoDataFrame.plot()». En: (2022. Consultado el 15 de mayo, 2022). URL: https://stackoverflow.com/questions/58945250/proper-reuse-of-axes-in-geodataframe-plot.

[69] STACKOVERFLOW. «Use pykalman to predict further steps on dynamic objects». En: (2022. Consultado el 13 de mayo, 2022). URL: https://stackoverflow.com/questions/65432475/use-pykalman-to-predict-further-steps-on-dynamic-objects.

[70] MARTIN STROHMEIER. «The basic structure of Mode S downlink replies». En: (2022. Consultado el 6 de junio, 2022). URL: https://www.researchgate.net/figure/The-basic-structure-of-Mode-S-downlink-replies-The-downlink-format-DF-determines-the_fig1_311610416.

[71] JUNZI SUN. «A Guide to Decoding Mode S and ADS-B Signals». En: (2022. Consultado el 6 de junio, 2022). URL: https://mode-s.org/decode/content/ads-b/1-basics.html.

[72] JORDI TORRES. «Redes neuronales recurrentes». En: (2019. Consultado el 10 de abril, 2022). URL: https://torres.ai/redes-neuronales-recurrentes.

[73] TRABBER. «¿Cómo ir de la T4 a la T4S en Barajas paso a paso?» En: (2016. Consultado el 11 de abril, 2022). URL: https://respuestas.trabber.com/preguntas/2569/como-ir-de-la-t4-a-la-t4s-en-barajas-paso-a-paso.

[74] DIANA RAMÓN VILARASAU. «La congestión del espacio aéreo europeo aumenta un 30 % el tiempo de vuelo». En: (2019 Consultado el 19 de marzo, 2022). URL: https://www.hosteltur.com/129931_la-congestion-del-espacio-aereo-europeo-aumenta-un-30-la-duracion-de-vuelo.html.

[75] VEDRAN VUKOTI y col. «One-Step Time-Dependent Future Video Frame Prediction with a Convolutional Encoder-Decoder Neural Network». En: (2017. Consultado el 2 de abril, 2022).

[76] WICHO. «Así se pone nombre a las pistas de un aeropuerto». En: (2021. Consultado el 11 de abril, 2022). URL: https://www.microsiervos.com/archivo/aerotrastorno/sabias-como-se-nombran-las-pistas-de-vuelo.html.

[77] WIKIPEDIA. «Aeronáutica». En: (2022. Consultado el 10 de abril, 2022). URL: https://es.wikipedia.org/wiki/Aeron%C3%A1utica.

[78] WIKIPEDIA. «Aeropuerto Adolfo Suárez Madrid-Barajas». En: (2022. Consultado el 20 de marzo, 2022). URL: https://es.wikipedia.org/wiki/Aeropuerto_Adolfo_Su%C3%A1rez_Madrid-Barajas.

[79] WIKIPEDIA. «Aerovía». En: (2021 Consultado el 20 de marzo, 2022). URL: https://es.wikipedia.org/wiki/Aerov%C3%ADa.

[80] WIKIPEDIA. «Air traffic management». En: (2022. Consultado el 20 de marzo, 2022). URL: https://en.wikipedia.org/wiki/Air_traffic_management.

[81] WIKIPEDIA. «Algoritmo Hill Climbing». En: (2019. Consultado el 11 de abril, 2022). URL: https://es.wikipedia.org/wiki/Algoritmo_hill_climbing#Descripci%C3%B3n_matem%C3%A1tica.

[82] WIKIPEDIA. «Backpropagation through time». En: (2022. Consultado el 26 de marzo, 2022). URL: https://en.wikipedia.org/wiki/Backpropagation_through_time#Advantages.

[83] WIKIPEDIA. «Control del tráfico aéreo». En: (2022 Consultado el 20 de marzo, 2022). URL: https://es.wikipedia.org/wiki/Control_del_tr%C3%A1fico_a%C3%A9reo.

[84] WIKIPEDIA. «Código de aeropuertos de OACI». En: (2022. Consultado el 7 de mayo, 2022). URL: https://es.wikipedia.org/wiki/C%C3%B3digo_de_aeropuertos_de_OACI.

[85] WIKIPEDIA. «Free flight (air traffic control)». En: (2021. Consultado el 14 de abril, 2022). URL: https://en.wikipedia.org/wiki/Free_flight_(air_traffic_control).

[86] WIKIPEDIA. «Ground Speed». En: (2021. Consultado el 11 de abril, 2022). URL: https://en.wikipedia.org/wiki/Ground_speed.

[87] WIKIPEDIA. «Iridium». En: (2021 Consultado el 20 de marzo, 2022). URL: https://es.wikipedia.org/wiki/Iridium.

[88] WIKIPEDIA. «Perceptrón Multicapa». En: (2022. Consultado el 11 de abril, 2022). URL: https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa.

[89] WIKIPEDIA. «Radar secundario». En: (2020 Consultado el 20 de marzo, 2022). URL: https://es.wikipedia.org/wiki/Radar_secundario.

[90] WIKIPEDIA. «Terminal control area». En: (2022. Consultado el 2 de junio, 2022). URL: https://en.wikipedia.org/wiki/Terminal_control_area.

[91] WIKIPEDIA. «Using Pandas and Python to Explore Your Dataset». En: (2022. Consultado el 7 de mayo, 2022). URL: https://realpython.com/pandas-python-explore-dataset/.

[92] HMONG WIKIPEDIA. «Separación (aeronáutica)». En: (2022. Consultado el 11 de abril, 2022). URL: https://hmong.es/wiki/Separation_(air_traffic_control).

[93] MICHAL WOJCZULIS. «Introduction To Deep Learning With Fastai: This Is Why Deep Learning Can Work For Everyone». En: (2020. Consultado el 16 de abril, 2022). URL: https://dlabs.ai/blog/introduction-to-deep-learning/.

[94] JUN WU. «AI, Machine Learning, Deep Learning Explained Simply». En: (2019. Consultado el 26 de marzo, 2022). URL: https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960.

[95] ZHUANG DAI XIAOLEI MA y col. «Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction». En: (2017. Consultado el 2 de abril, 2022).