UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Mantenimiento predictivo de motores Turbofán mediante Deep Learning

Trabajo Fin de Máster

Máster Universitario en Ingeniería Aeronáutica

AUTOR/A: Álvarez González, Javier

Tutor/a: Rodas Jordá, Ángel

CURSO ACADÉMICO: 2021/2022

Resumen

Los métodos de *Machine Learning* se expanden de manera inexorable por todas las capas de la sociedad. Entre otros, el campo de la aeronáutica ha implementado estas herramientas en diversas áreas como el ATM (*Air Traffic Management*), marketing, etc. Siendo de especial interés el campo del mantenimiento predictivo.

El presente Trabajo Final de Máster aborda este último campo, el mantenimiento predictivo enfocado a motores Turbofán, conformando estos componentes una de las partes más críticas y complejas del mantenimiento de aeronaves.

Para ello, mediante la herramienta MATLAB y un extensivo análisis bibliográfico, se ha buscado un profundo entendimiento del problema propuesto de manera tanto analítica como experimental. Se ha realizado un análisis exhaustivo de las características de la base de datos, así como alcanzar una arquitectura de red neuronal que nos permita realizar predicciones de la vida del motor que rivalicen en precisión con las publicaciones actuales del problema. Utilizando lo anterior como base, también se han estudiado frentes abiertos del problema, proponiendo y analizando estructuras alternativas.

Para llevar a cabo este estudio se ha hecho uso de la conocida base de datos "*Turbofan engine degradation simulation data set*" creada por el *prognostics CoE de Nasa Ames*, la cual ha servido de base para explorar las posibilidades de las redes neuronales aplicadas a este tipo de problemas.

Palabras clave: mantenimiento predictivo, MATLAB, *Machine Learning*, redes neuronales, Turbofán, vida del motor.

Abstract

Machine Learning methods are inexorably spreading to all layers of society. Among others, the field of aeronautics has implemented these tools in different areas such as ATM (Air Traffic Management), marketing, etc. Of particular interest is the field of predictive maintenance.

This Master's thesis deals with the latter field, predictive maintenance focused on turbofan engines, these components being one of the most critical and complex parts of aircraft maintenance.

For this purpose, using the MATLAB tool and an extensive bibliographic analysis, a deep understanding of the proposed problem has been sought both analytically and experimentally. An exhaustive analysis of the characteristics of the database has been carried out, as well as reaching a neural network architecture that allows us to make predictions of engine life that rival in accuracy with current publications on the problem. Using the above as a basis, open fronts of the problem have also been studied, proposing and analysing alternative structures.

To carry out this study, use has been made of the well-known "Turbofan engine degradation simulation data set" database created by the prognostics CoE at NASA Ames, which has served as the basis for exploring the possibilities of neural networks applied to this type of problem.

Keywords: predictive maintenance, MATLAB, Machine Learning, neural networks, Turbofan, engine life.

Resum

Els mètodes de *Machine Learning* s'expandeixen de manera inexorable per totes les capes de la societat. Entre d'altres, el camp de l'aeronàutica ha implementat aquestes eines en diverses àrees com l'ATM (*Air Traffic Management*), el màrqueting, etc. Sent especial interès el camp del manteniment predictiu.

Aquest Treball Final de Màster aborda aquest darrer camp, el manteniment predictiu enfocat a motors Turbofán, conformant aquests components una de les parts més crítiques i complexes del manteniment d'aeronaus.

Per això, mitjançant l'eina MATLAB i una extensiva anàlisi bibliogràfica, s'ha buscat una profunda entesa del problema proposat de manera tant analítica com experimental. S'ha realitzat una anàlisi exhaustiva de les característiques de la base de dades, així com assolir una arquitectura de xarxa neuronal que ens permeti fer prediccions de la vida del motor que rivalitzin en precisió amb les publicacions actuals del problema. Utilitzant això com a base, també s'han estudiat fronts oberts del problema, proposant i analitzant estructures alternatives.

Per dur a terme aquest estudi s'ha fet ús de la coneguda base de dades "*Turbofan engine degradation simulation data set*" creada pel *prognostics CoE de Nasa Ames*, que ha servit de base per explorar les possibilitats de les xarxes neuronals aplicades a aquest tipus de problemes.

Paraules clau: manteniment predictiu, MATLAB, Machine Learning, xarxes neuronals, Turbofán, vida del motor.

Índice de contenido

Índice de Figuras

Índice de tablas

Glosario

ANN ..... *Artificial Neural Network*
CNN ..... *Convolutional Neural Network*
DBSCAN..... *Density-Based Spatial Clustering of Applications with Noise*
LSTM ..... *Long Short Term Memory*
MLP..... *Multi Layer Perceptron*
OEM..... *Original Equipment Manufacturers*
RNN ..... *Recurrent Neural Network*
RUL..... *Remaining Useful Life*

1. Introducción

El *Machine Learning* (en adelante *ML*) se está estableciendo como una disciplina clave para que todo tipo organizaciones puedan competir en el mercado contemporáneo. En el contexto industrial actual, la empresa ya no solo busca saber que está pasando en el momento, sino que también busca conocer como se desarrollarán las operaciones en el futuro. Esto es posible gracias a los algoritmos de *ML* que, mediante el análisis de los datos disponibles, son capaces de crear modelos analíticos que nos permiten extraer nuevas conclusiones de las fuentes de información. En definitiva, estos métodos nos aportan sobre todo una mayor rapidez y eficiencia a la hora de tomar decisiones, lo que puede resultar muy ventajoso desde un punto de vista competitivo.

Actualmente la aviación genera un volumen de datos mayor que en ningún otro momento de la historia, estimándose en una cantidad en torno a los 2 millones de terabytes al año a través de programas de seguimiento de vuelo o de monitoreo del estado de las aeronaves. Es por esto, y considerando además que esta cantidad de datos se incrementará en gran medida en el futuro, que las aerolíneas buscan poder sacarle partido a esta mina de información.

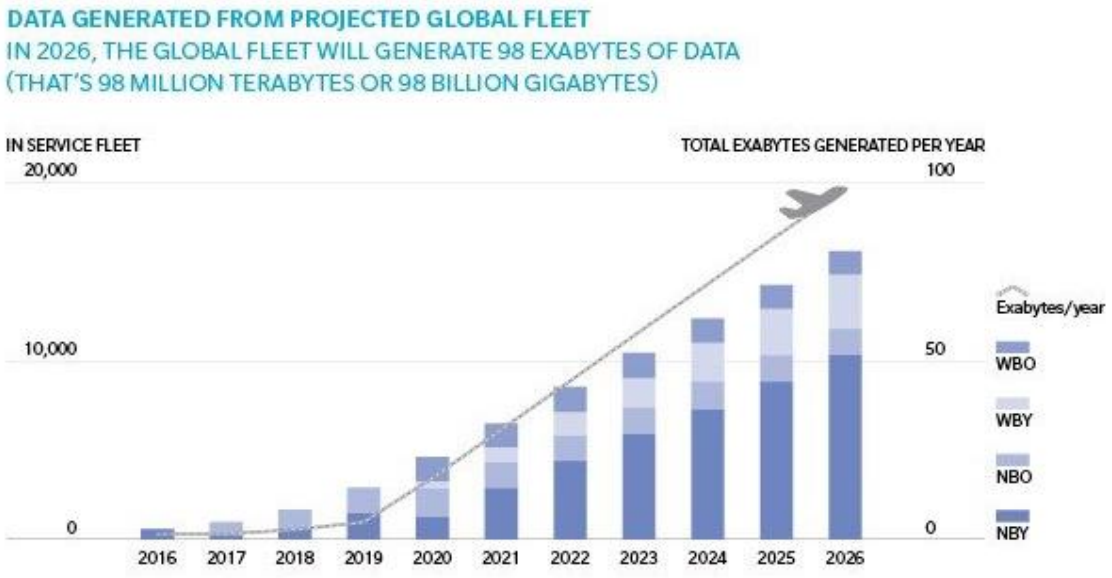


Figura 1-1. Volumen de datos generados por la flota global de aviones cada año [1]

Gracias a esto, las herramientas de *Big Data* se han presentado ante la industria aeronáutica como una herramienta capaz de revolucionar la forma en la que las aerolíneas llevaban a cabo muchas de sus actividades. El concepto de *Big Data* hace referencia justamente a este enorme volumen de datos que inunda las empresas actualmente, cuyo tamaño, complejidad y velocidad de crecimiento hacen difícil su análisis mediante herramientas convencionales [2].

Generalmente esta tarea se encuentra englobada por lo que es conocido como *Flight Operational Quality Assurance* (FOQA), aunque también conocido como *Flight Data Monitoring* (FDM) o *Flight Data Analysis*, que consiste en métodos para obtener, analizar y visualizar los datos generados por una aeronave que se encuentra en operación. Muchas aerolíneas y fuerzas aéreas han implementado programas en su flota dentro de este ámbito con objetivo de mejorar la seguridad general, incrementar la eficiencia del mantenimiento y reducir los costes operacionales [3].

Es de hecho el mantenimiento una de las áreas donde se han advertido mayores posibilidades, conociéndose esta disciplina como mantenimiento predictivo. El mantenimiento predictivo podría definirse como "la manera inteligente de maximizar la vida útil de la máquina", ya que con la información y los métodos adecuados es posible conocer las condiciones en las que se encuentra el dispositivo y así predecir cuándo las labores de mantenimiento han de ser realizadas. Esto por tanto nos permite realizar un uso más óptimo de los recursos de la aerolínea a la vez que se previene la aparición de fallos en los dispositivos [4].

En este trabajo nos centraremos en el mantenimiento predictivo orientado a los motores aeronáuticos. Las razones para centrarnos en esta parte del avión han sido principalmente por lo crítico que resulta mantener los motores en un estado óptimo para asegurar toda la operación del avión y por la existencia de una base de datos abierta, contrastada y de calidad proporcionada por el *Prognostics CoE at Nasa Ames* [5].

1.1. Motivación

La importante ventaja operativa que supone tener predicciones realistas acerca del funcionamiento de máquinas complejas ha impulsado el desarrollo de los métodos orientados al análisis de datos en la última década. Los avances en los métodos de *ML*, y más específicamente de *Deep Learning* (en adelante *DL*), unidos al aumento imparable del volumen de datos disponibles convierte a este campo del análisis de datos en uno de los más prometedores para la mayor parte de las aerolíneas [1], a la vez que su uso ya se encuentra implementado en muchas de ellas y principalmente por los OEM (*Original Equipment Manufacturer*).

Dentro de una aeronave, el motor es uno de los componentes más críticos y complejos, conformando una de las partes más determinantes a la hora de llevar a cabo las labores de mantenimiento. Es por estas características que la mayor parte de las iniciativas relativas al mantenimiento predictivo en aeronáutica se centran en esta área, existiendo sobre el motor la mayor cantidad de las bases de datos disponibles.

Es por las innumerables posibilidades que ofrece el apasionante campo de los algoritmos de *DL*, unido al excelente desarrollo profesional que presenta esta área del mantenimiento, que se ha elegido realizar este estudio sobre las aplicaciones del *DL* en el mantenimiento predictivo de motores de aviación.

1.2. Objetivos

En el presente Trabajo de Fin de Máster se busca afrontar la tarea del mantenimiento predictivo en motores Turbofán mediante la utilización de redes neuronales profundas, también conocidas como *DL*. De esta máxima principal se extraen diversos objetivos secundarios:

- Estudiar las posibilidades y metodología del campo del *DL* en profundidad.
- Revisar el estado del arte de los métodos utilizados en el mantenimiento predictivo enfocados al *DL*.
- Analizar las características de las bases de datos utilizadas.
- Realizar una comparación entre las diferentes propuestas implementadas.
- Proponer una estructura que rivalice con las publicaciones actuales del problema.

1.3. Estructura del documento

La estructura del presente Trabajo de Fin de Máster se organiza de la siguiente manera.

- **Capítulo 1:** presenta la introducción, motivación y objetivos del trabajo.
- **Capítulo 2:** estado del arte del área del mantenimiento predictivo.
- **Capítulo 3:** explicación de las estructuras y conceptos básicos de las redes neuronales.
- **Capítulo 4:** descripción de las bases de datos y herramientas utilizadas.
- **Capítulo 5:** descripción de la metodología llevada acabo en la solución propuesta.
- **Capítulo 6:** resultados obtenidos con las diferentes arquitecturas implementadas.
- **Capítulo 7:** conclusiones extraídas y posibles líneas de investigación futuras.

2. Introducción al mantenimiento predictivo

2.1. Trabajos previos realizados en la UPV

Se ha indagado dentro de la base de datos RiuNet de la UPV en busca de diferentes trabajos previos relativos a la temática estudiada. Si bien la cantidad de publicaciones relativas al campo de la inteligencia artificial es variada, estas escasean cuando ponemos el foco en áreas más acotadas como pueden ser sus aplicaciones en el mantenimiento o la aeronáutica como tal.

Dentro del mantenimiento predictivo, las aplicaciones de redes neuronales más cercanas a nuestro estudio son, por ejemplo, trabajos relativos al mantenimiento de palas eólicas. Algunos de estos son el [6] en el que se utilizan distintos métodos de *ML* a la hora de encontrar fallos en los aerogeneradores mediante datos obtenidos por análisis de vibraciones, o el [7] en el que se busca identificar fallos en las palas eólicas gracias a el análisis de imágenes mediante redes convolucionales. Existen otros trabajos relacionados con el mantenimiento predictivo, aunque estos se centran principalmente en el análisis de ciertas técnicas de obtención de datos, como pueden ser técnicas de termografía, o en la implantación de este tipo de tecnología en ciertas empresas.

En cuanto a las aplicaciones del *ML* en el campo de la aeronáutica en general, estas tampoco son abundantes. Se pueden destacar algunas relativas al establecimiento de rutas como [8], en el que se busca encontrar la ruta más óptima basándose en datos meteorológicos.

En resumen, podemos concluir que el mantenimiento predictivo en motores aeronáuticos no es un tema para nada común en los artículos de la UPV, siendo necesario recurrir a información proporcionada por otras fuentes.

2.2. Mantenimiento predictivo

Desde que la humanidad empezó a valerse del uso de máquinas para la realización de diferentes tareas, el mantenimiento ha sido una parte inherente en el proceso de utilización de estas [9]. Por otro lado, las estrategias seguidas para llevar a cabo esta tarea han sido enfocadas desde distintas perspectivas, las cuales pueden diferenciarse en las siguientes:

- **Mantenimiento correctivo:** este sería el más evidente ya que consiste en reparar la máquina cuando se produce el fallo de esta. Como estrategia empresarial no es lo más indicado ya que el tiempo durante el que se encuentra la máquina inactiva supone un coste económico para la empresa.
- **Mantenimiento preventivo:** consiste en realizar tareas de mantenimiento, aunque la máquina no haya fallado con el objetivo de evitar ese riesgo en el

futuro. Para ello se establece un plan de mantenimiento propio de la máquina, el cual previene que se estropee en condiciones normales.

- **Mantenimiento predictivo:** este es el método más novedoso y con mayor potencial actualmente, siendo el tema del presente trabajo. Mediante la recopilación y análisis de datos extraídos del funcionamiento de la máquina estudiada, se pueden identificar anomalías en su operación y establecer un plan de mantenimiento más flexible y adaptado a las necesidades reales de la máquina para evitar su futura avería.

A primera vista ya podemos ver que el mantenimiento predictivo presenta ciertas ventajas con respecto a los otros, ya que cuando simplemente se programa un plan de mantenimiento cada cierto tiempo, se cae en el problema de gastar recursos cuando no se necesitan o de no actuar a tiempo ya que no se han identificado las señales. Es por esto que el mantenimiento predictivo se considera una herramienta que sobre todo optimiza los procesos en las empresas.

De todas maneras, la implementación de esta técnica conlleva una importante inversión de recursos para que pueda ser desarrollada con todo su potencial. Se necesita de toda una estructura capaz de monitorear las máquinas estudiadas de manera continua para poder conocer en todo momento el estado del dispositivo, este sistema por su lado podemos dividirlo en tres componentes principales:

- **Sensores:** dispositivos instalados en las máquinas que proveen de todo tipo de datos en función de sus características. Estos datos serán la fuente de información de donde se obtendrán las conclusiones pertinentes para llevar a cabo el mantenimiento.

- **Las soluciones de software y almacenamiento en la nube:** son métodos que nos permiten servirnos de la minería de datos para recopilar y analizar el enorme volumen de información que se genera mediante el uso de aplicaciones *Big Data*.

- **Los modelos predictivos:** están formados por los métodos que analizan la información generada y recopilada. Cuando estos se encuentran adecuadamente calibrados son capaces de simular la operación del sistema y predecir su comportamiento, lo que permite programar el mantenimiento de manera mucho más eficiente. Entre estos métodos destaca actualmente la utilización de algoritmos de *ML*.

Para obtener estos codiciados datos que nutren a nuestros modelos hace falta aplicar diferentes ensayos y monitoreo de la máquina. Dado que lo que se busca es maximizar la vida de la máquina, es obvio que los ensayos escogidos han de ser no destructivos y que no afecten en absolutos a la salud de esta. Dentro de este campo existen multitud de técnicas que deberán elegirse de manera individual según su idoneidad a cada problema

propuesto. Algunas de ellas pueden ser el análisis de vibraciones, termografía, mediciones de temperatura y presión, monitoreo acústico, etc.

En la Figura 2-1 se puede apreciar una comparativa en la que se resalta como el mantenimiento predictivo actúa con el mayor margen de tiempo, con el sistema estudiado en mejor estado y sirviéndose de métodos para la obtención de datos cuando el riesgo de fallo todavía es bajo.

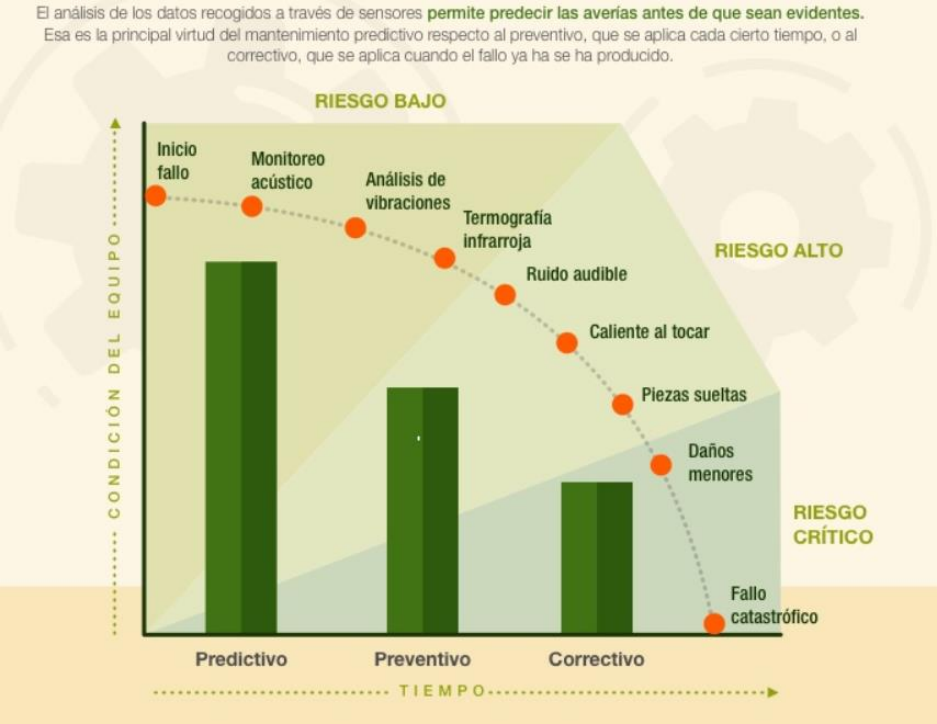


Figura 2-1. Comparativa de los distintos tipos de mantenimiento [9]

En definitiva, el mantenimiento predictivo se trata de una disciplina con tremendas posibilidades y que mejora el rendimiento general de la maquinaria, aunque su implementación presenta un reto técnico a la vez que económico ya que supone una infraestructura compleja y cara. Es por esto que su uso dependerá mucho de la actividad realizada por cada empresa.

2.3. Modelos de predicción de vida útil

La RUL (*Remaining Useful Life*) se define como la longitud (medida en las unidades pertinentes en cada caso) que existe entre el momento en el que se hace la predicción y el momento de fallo del elemento estudiado. Para obtener este tipo de predicciones existen diferentes técnicas caracterizadas por el método que utilizan para operar. En la Figura 2-2 se recoge una clasificación de los tipos de técnicas con diferentes ejemplos de cada una.

A continuación, se realizará una explicación más detallada de cada uno de estos grupos en los apartados posteriores.

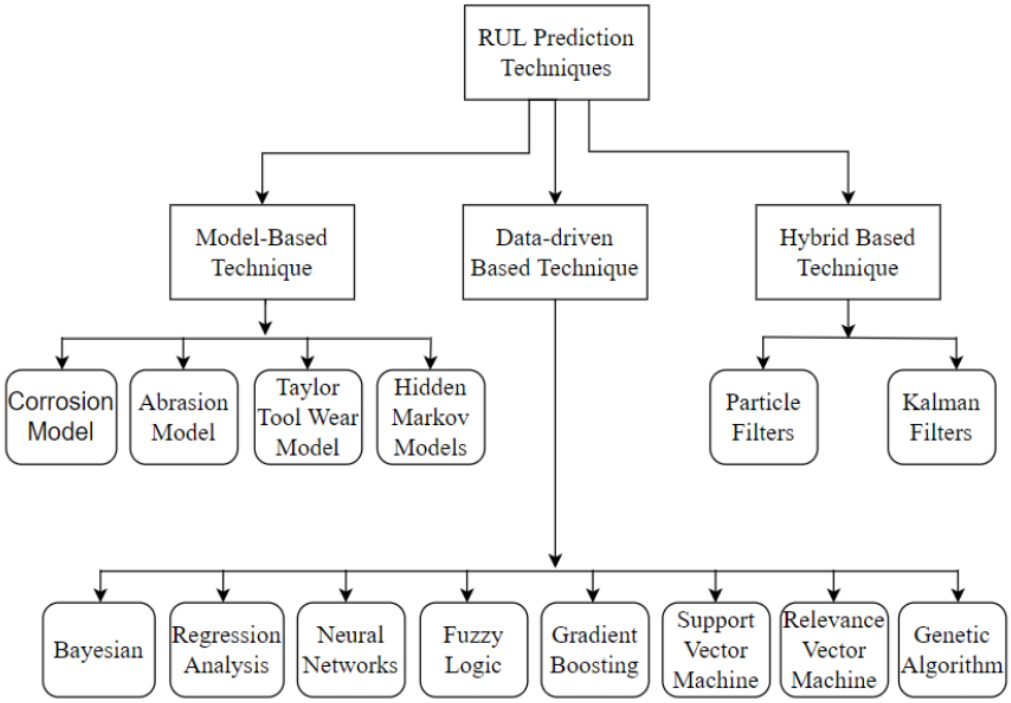


Figura 2-2. Diferentes técnicas para la estimación de la RUL

2.3.1. Técnica basada en modelos físicos

Este tipo de métodos se basan en modelos físicos que representan fielmente la dinámica de funcionamiento del sistema en cuestión y que, introduciendo los datos que obtenemos de la máquina, son capaces de simular y predecir su comportamiento.

Este tipo de modelos son fáciles y rápidos de implementar, pudiendo ser reutilizados constantemente. A pesar de esto, el desarrollo de este tipo de modelos es largo y costoso, necesitándose poseer un profundo entendimiento de todos los procesos involucrados en el sistema. Además, modelos con un alto de grado de exactitud pueden resultar también muy exigentes computacionalmente.

2.3.2. Métodos basados en el análisis de datos

Este tipo de métodos hacen uso de los datos almacenados sobre el funcionamiento del sistema para analizar sus características y como se producen y evolucionan los fallos para luego, a través de esta información aprendida, realizar predicciones futuras. Es en esta área donde sobresale el campo del *ML*, ofreciendo muy diversas posibilidades como *Random Forest*, *Support Vector Machine* o las conocidas redes neuronales o *DL* [10].

Este tipo de métodos son fáciles y rápidos de implementar, no siendo necesario poseer conocimientos exhaustivos sobre el funcionamiento del sistema estudiado, siendo posible además que encuentren patrones o relaciones que antes no se habían

considerado. De todas formas, el desarrollo de los modelos dependerá de la complejidad del problema tratado.

Por otro lado, la base de datos disponible toma aquí una dimensión primordial ya que generalmente se necesitan elevados volúmenes de estos. Además, las bases de datos han de poder proveer al algoritmo de la información más completa posible, permitiendo que este pueda "entender" el comportamiento del sistema en su totalidad y no se centre en un caso concreto.

2.3.3. Métodos híbridos

Este tipo de métodos son una combinación de los dos anteriores. Tratan de utilizar las fortalezas de cada uno y utilizan los datos donde el entendimiento del sistema no es completo, y modelos cuando faltan datos. Por contraparte, en estos métodos tanto un elevado número de datos como conocimiento del funcionamiento del sistema van a ser necesarios.

2.4. Mantenimiento predictivo de motores Turbofán mediante redes neuronales

En este apartado se recoge la bibliografía relativa al problema que se estudia en el presente trabajo. La base de datos que se ha decidido utilizar para el análisis es bastante conocida en el campo de estudio del mantenimiento predictivo mediante algoritmos de *ML*, quizá debido a la escasez de bases de datos de calidad y de carácter abierto, por lo que existe un número considerable de publicaciones que hacen uso de ella. En la bibliografía encontrada algunos trabajos se centran totalmente en el estudio de esta base de datos y otros la desplazan a un uso más accesorio, pero en esta sección se van a resumir los proyectos que más interesantes nos han parecido o de donde se han extraído ideas o información.

Las redes neuronales son con diferencia el método más utilizado para la estimación de la vida restante de motores turbofán. Esto es debido a la complejidad de la base de datos y la necesidad de un método capaz de encontrar patrones en los datos de manera independiente, muchas veces difíciles de identificar con precisión desde un punto de vista humano. En [11] podemos apreciar una comparativa entre métodos tradicionales de *ML* y redes neuronales, viendo como estás últimas destacan claramente en la tarea.

En un inicio, redes neuronales del tipo MLP (*Multi Layer Perceptron*) eran utilizadas. Las redes del tipo perceptrón multicapa agrupan capas de neuronas ocultas que procesan las entradas en una única dirección pudiendo representar funciones no lineales. De todas formas, la salida de estas redes aplicada al análisis de motores turbofán contiene bastante ruido como podemos ver en [12].

Es por lo anterior que entran en juego las redes neuronales recurrentes o RNN (*Recurrent Neural Network*). Este tipo de redes son capaces de identificar relaciones temporales en los datos ya que, en líneas generales, para cada paso temporal utiliza

tanto la entrada recibida de la capa que la precede como su propia salida, pero en el paso temporal anterior. Esta característica las hace especialmente idóneas para tareas como la que nos ocupa, en la cual los datos son series temporales del funcionamiento de un motor. Dentro de esta gama de redes destacan las conocidas como LSTM (*Long Short Term Memory*), las cuales pueden realizar un uso más "inteligente" de la información que van acumulando de pasos temporales previos, pudiendo retener o eliminar información según lo relevante que se considere. En [13] se propone una arquitectura estándar para tratar datos secuenciales, formada por varias capas LSTM seguidas de una MLP que produce la predicción final.

En [14] se explora la idea de utilizar redes LSTM junto MLP para ayudar a la parte de la red recurrente a obtener un mejor entendimiento de las secuencias. En particular, la idea base de esta publicación es que una capa MLP situada antes que la capa LSTM procesará las entradas obteniendo una buena representación de estas, para que luego la LSTM pueda captar mejor las dependencias temporales presentes en los datos y finalmente otra capa MLP haga las predicciones. Esta estructura destaca por su sencillez si la comparamos con otras publicaciones contemporáneas, pero también por su rendimiento. Su buen funcionamiento se ve reflejado principalmente en los buenos resultados obtenidos sobre los 4 subconjuntos de la base de datos C-MAPSS [15], cuya división se explica en la Sección 4.1.2. Dentro de estos subconjuntos, unos presentan más dificultades para ser analizados y esto se refleja claramente en la pobreza de las predicciones de la mayoría de las publicaciones en ellos y como muchas de ellas limitan su análisis solo a los subgrupos que dan menos problemas.

Aunque menos común, ciertas publicaciones como [16] hacen uso de una variante de las redes LSTM conocidas como Bi-LSTM. Este tipo redes presentan una estructura similar a las LSTM normales, salvo por que buscan obtener información recorriendo ambas direcciones temporales. Más información sobre su funcionamiento puede ser encontrada en [17].

Otro tipo de estructura que también ha gozado de cierto protagonismo es la utilización de redes convolucionales o CNN (*Convolutional Neural Network*). Estas redes utilizan una ventana temporal para estudiar la base de datos y extraer la información pertinente. Los resultados obtenidos con este tipo de redes son satisfactorios en general, aunque no suelen ser constantes en todas las bases de datos disponibles. Una comparación del desempeño de redes convolucionales en el mantenimiento predictivo puede encontrarse en [11].

De hecho también existen casos en el que ambas estructuras (LSTM y CNN) se han combinado en una misma red tratando de aprovechar las fortalezas de cada una de ellas como en [18], donde se consiguen resultados muy competitivos, aunque no se ofrecen datos del comportamiento de la red en todos los subconjuntos de la base de datos disponibles.

Si bien la mayoría de las publicaciones modelan el deterioro de los motores como una degradación lineal para entrenar los distintos algoritmos estudiados, existen trabajos que proponen cambios en este ámbito. Por un lado tenemos [19] donde el estado del motor se modeliza a través de una función exponencial que va desde un estado inicial de no degradación y evoluciona hasta 0. Por otro lado tenemos [20], donde se realiza un proceso de extracción de características de la base de datos y fusión de ellas para construir un índice de salud del motor que modeliza su degradación, teniéndose de nuevo una progresión no lineal que se sale de las técnicas habituales. En ambas publicaciones los resultados obtenidos son ciertamente competitivos por lo que abren la puerta a cambiar ciertas convenciones existentes a la hora de entrenar los algoritmos, al menos en este caso de estudio.

A parte de las estructuras anteriores, se han utilizado otros enfoques como en [21] en el que se hace una clasificación de los diferentes estados por los que pasa el motor antes de fallar mediante un proceso de agrupación (*Clustering*), para luego identificar en que estado se encuentra el motor y con ello se estima el número de ciclos que le quedan para dejar de estar operativo. Esta manera de obtener las predicciones se sale un poco de la línea habitual observada en la bibliografía, ofreciendo un punto de vista interesante sobre el problema. No obstante, los resultados obtenidos son menos satisfactorios que en las publicaciones previas.

3. Redes neuronales

3.1. Introducción y principales estructuras de las redes neuronales

En la actualidad los métodos de *ML* se encuentran muy extendidos en la sociedad, abarcando diferentes aplicaciones como pueden ser el reconocimiento de voz en dispositivos móviles o dar recomendaciones de productos a los usuarios a partir de sus búsquedas previas.

De todos modos, los métodos tradicionales de *ML* están limitados a la hora de procesar datos en bruto, necesitando de la intervención continua de la ayuda humana. Esto es debido a que estos algoritmos requieren que los datos sean procesados previamente extrayendo las características subyacentes que esconden para que el modelo pueda aprender de ellos. Esto evidentemente hace que estos métodos sean menos independientes y que requieran de más tiempo y esfuerzo por parte de mano de obra experta, siendo a veces incapaces de encontrar estos patrones en los datos.

Es aquí donde entran los algoritmos de *DL* en juego. El campo del *DL* se define como una disciplina dentro del *ML* que crea redes de módulos no-lineales que transforman los datos de entrada en una representación más abstracta de estos. Este conocimiento aprendido en cada capa se transfiere a la siguiente de un modo jerarquizado. Estos algoritmos son mucho más autónomos, ya que extraen las características relevantes de los datos de manera independiente y a menudo llegan a predicciones más precisas y en ámbitos de mayor complejidad.

Los algoritmos de *DL* llevan tiempo existiendo, pero ha sido en los últimos tiempos cuando su popularidad se ha disparado gracias al incremento incesante de la capacidad computacional o la existencia de bases de datos de gran tamaño necesarias para nutrir a este tipo de algoritmos.

El *DL* se trata de un campo de gran densidad y continuo desarrollo, por lo que las posibilidades son muy variadas. No obstante, podemos diferenciar entre tres tipos de redes neuronales principales:

- *Artificial Neural Network* (ANN)
- *Recurrent Neural Network* (RNN)
- *Convolutional Neural Network* (CNN)

3.1.1. Perceptrón

El perceptrón es la unidad básica de una red neuronal y se puede asemejar a una neurona artificial [22]. Desde un punto de vista matemático responde a una suma ponderada de unas entradas seguida de una función de activación.

En la Figura 3-1 podemos un esquema de la estructura del perceptrón.

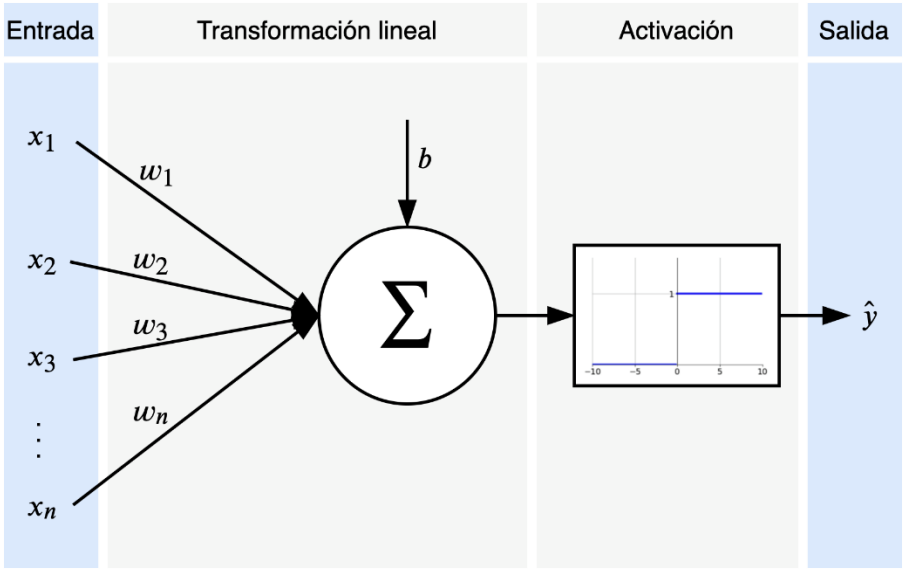


Figura 3-1. Estructura del Perceptrón [23]

Por su lado en la Ecuación (3-1) podemos ver la suma ponderada que se realiza dentro de la neurona, donde las entradas son multiplicadas por los pesos atribuidos a ésta y a los que se le añade un término de sesgo o *bias*. Por otro lado, la Ecuación (3-2) presenta una ecuación de activación a modo de ejemplo, utilizada para obtener la salida de la neurona. Estas funciones de activación pueden ser no-lineales y de varios tipos como se recoge en la Sección 3.3.1.

$$z(x) = \sum_{i=1}^{D} w_i x_i + w_0 = w^T x + w_0 \tag{3-1}$$

$$s(x) = f(z(x))$$
$$f(z) = \begin{cases} 1 & if\ z > 0 \\ -1 & if\ z < 0 \end{cases} \tag{3-2}$$

Estos pesos y el sesgo son los parámetros que deben ser determinados o aprendidos.

La no-linealidad de la función de activación es un concepto primordial en la estructura del perceptrón ya que es lo que permite que pueda utilizarse satisfactoriamente en problemas que no sean estrictamente lineales, que son la gran mayoría de los problemas complejos en los que se va a utilizar.

3.1.2. *Artificial Neural Networks* (ANN)

Las redes ANN se definen como un grupo de múltiples neuronas o perceptrones en cada capa de la red [24]. También se las conoce como redes neuronales *feed-forward*, dado que la información solo se procesa en una dirección. En la Figura 3-2 podemos observar una imagen de la estructura de esta red:

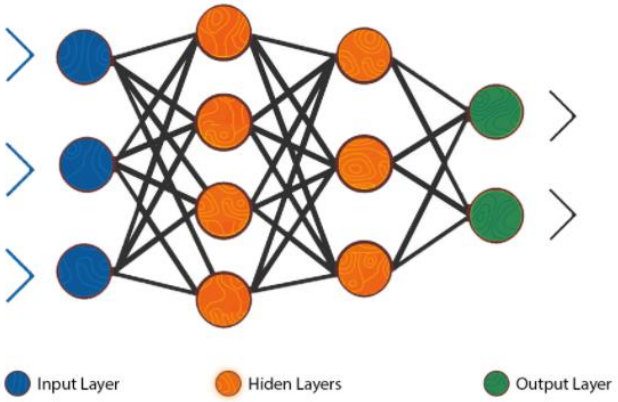


Figura 3-2. Estructura de las redes ANN [24]

Podemos apreciar 3 capas diferenciadas. Por un lado, la capa inicial que acepta las entradas, las capas ocultas que procesan esta información y las capas finales que producen el resultado final.

Las redes ANN son capaces de aprender cualquier función no lineal. Tienen la capacidad de optimizar sus parámetros para aprender cualquier relación entre entradas y salidas. Esto es debido en gran medida a las funciones de activación comentadas en el perceptrón que añaden las propiedades no lineales a la red.

De todas maneras, este tipo de arquitecturas puede presentar problemas en campos como el análisis de imágenes, debido al elevado volumen de datos que pueden llegar a presentar este tipo de problemas. Por otra parte, también son incapaces de detectar información secuencial en las bases de datos que estudian.

3.1.3. *Recurrent Neural Network* (RNN)

Las redes neuronales recurrentes son una herramienta utilizada principalmente con datos secuenciales o series temporales [25]. La principal diferencia que caracteriza a estas redes neuronales es su “memoria”, dado que tiene en cuenta información de entradas anteriores para condicionar las entradas y salidas actuales. En la Figura 3-3 podemos ver una comparación entre RNN y ANN.

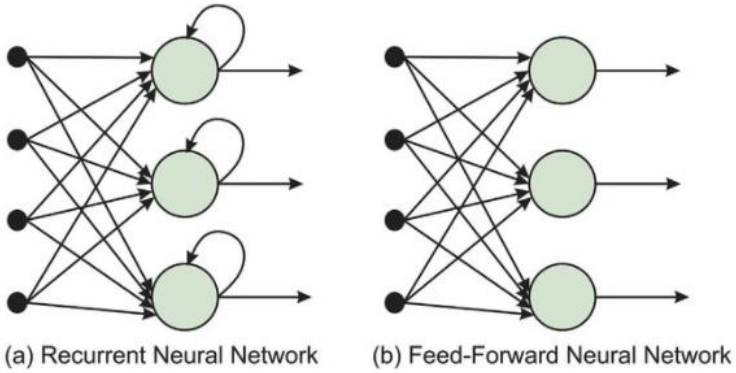


Figura 3-3. Comparación entre RNN y ANN [26]

Esta capacidad de retener información es especialmente útil cuando tenemos bases de datos donde la información está relacionada entre sí. De todas formas, un importante problema existente en este tipo de redes es lo que se conoce como "memoria de pez". Las RNN convencionales no son capaces de retener demasiada información por lo que su principal ventaja no se ve explotada tanto como podría esperarse. Es por esto que, dentro de este tipo de redes existen arquitecturas como pueden ser las LSTM, que presenta neuronas más complejas pero que nos permiten una utilización más efectiva de esta memoria. Otras arquitecturas derivadas de las RNN también son posibles para esta tarea como puede ser las GRU (*Gated Recurrent Unit).*

- ***Long Short Term Memory* (LSTM)** [27]

Las redes LSTM son una versión mejorada de las redes RNN estándar. Las RNN, debido a la idea que subyace dentro de ellas, no son capaces de aprender de dependencias a largo plazo, problema que se soluciona con las LSTM.

Podemos apreciar un primer resumen de las diferencias entre las RNN convencionales y las LSTM comparando la Figura 3-4 y Figura 3-5. En estas imágenes se aprecia que ambos tipos de neuronas presentan una estructura que produce una salida utilizando la entrada del paso temporal actual y la salida del paso temporal anterior (simbolizado por los subíndices t y t-1), pero el proceso para llegar a ellas contiene ostensibles diferencias.

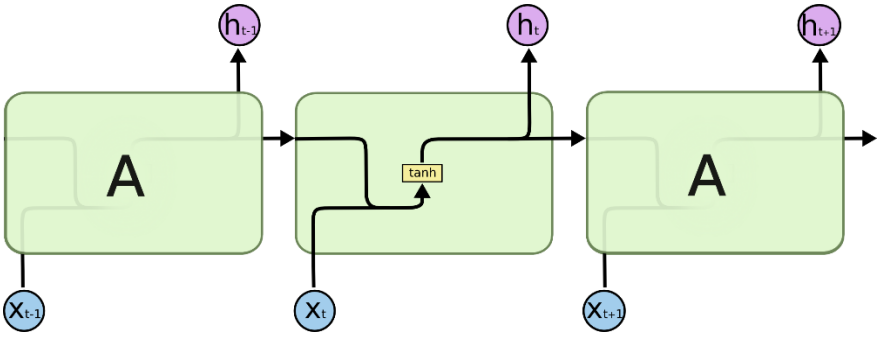


Figura 3-4. Módulo de repetición de una red RNN estándar [27]

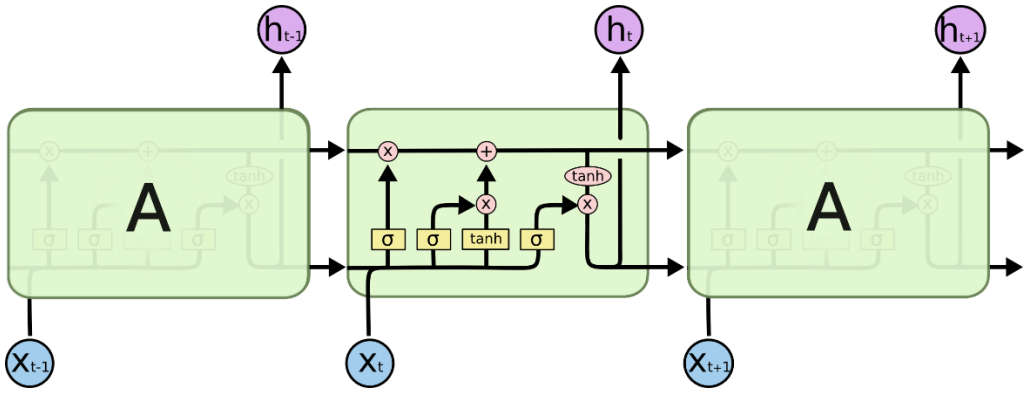


Figura 3-5. Módulo de repetición de una red LSTM [27]

La razón principal por la que este tipo de redes son capaces de aprender relaciones en los datos en largos períodos, se debe a la línea horizontal superior de la neurona. Esta línea resaltada en la Figura 3-6 se denomina "línea de estado".

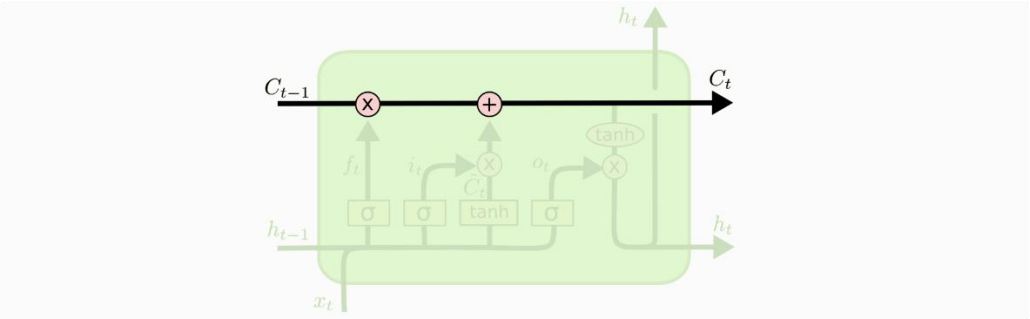
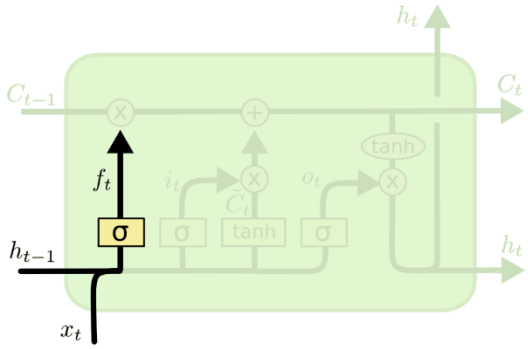


Figura 3-6. Línea de estado en una neurona LSTM [27]

La información que circula a través de este canal pasa por la cadena de neuronas sufriendo solo algunas interacciones lineares menores, siendo fácil que la información se mantenga intacta. Son de hecho estas interacciones a través de las cuales la neurona es capaz de quitar o añadir información al estado de la neurona según sea requerido.

El proceso de funcionamiento de esta neurona sigue el esquema siguiente:

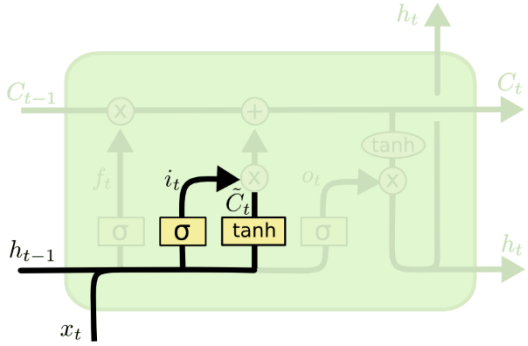
1. **Eliminar información, *forget gate layer*:** el primer paso consiste en decidir que información eliminaremos del estado. Esta decisión se lleva a cabo a través de una capa sigmoide. Mediante las entradas, se crea una salida entre 0 y 1 para cada número del estado C_{t-1} donde "0" significa eliminar esa información y "1" mantenerla.



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

Figura 3-7. *Forget gate layer* [27]

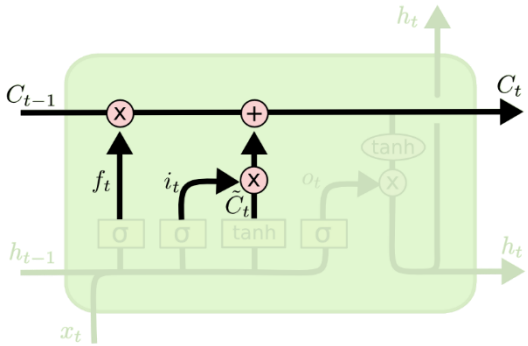
2. **Actualizar la información:** en este apartado decidimos que nueva información vamos a incluir en el estado, esta operación presenta dos partes. Primero, tenemos la capa de entrada, una capa sigmoide decide que valores se van a actualizar, y luego una capa tanh crea un vector de nuevos valores candidatos C_t que pueden ser añadidos al estado. La combinación de ambos será con lo que actualizaremos el estado.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 3-8. Creación de la actualización para el estado [27]

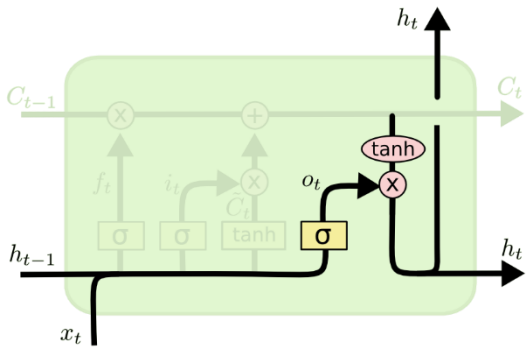
Una vez tenemos calculado tanto lo que debemos eliminar como lo que debemos añadir solo queda realizar unas pequeñas operaciones para implementarlo.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 3-9. Actualización del estado [27]

3. **Resultado:** finalmente lo que nos queda es emitir la predicción como resultado. Esta salida estará basada en el estado actualizado, aunque será una versión filtrada.



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

Figura 3-10. Emisión del resultado [27]

3.1.4. *Convolutional Neural Network* (CNN)

Este tipo de redes son principalmente utilizadas por sus ventajas en labores de clasificación de imágenes o *computer vision* entre otras.

Están formadas por diferentes capas, aunque la principal y la que las hace más especiales, es la inicial conocida como capa convolucional [28]. Básicamente, la operación que ocurre en esta capa se lleva a cabo a través de un "filtro" que revisa la imagen en busca de ciertas características (estas dependerán del filtro elegido). El filtro va examinando cada área del mapa dando lugar a un valor para cada una de estas y creando un mapa de características de la imagen inicial. En la Figura 3-11 se resume esta operación.

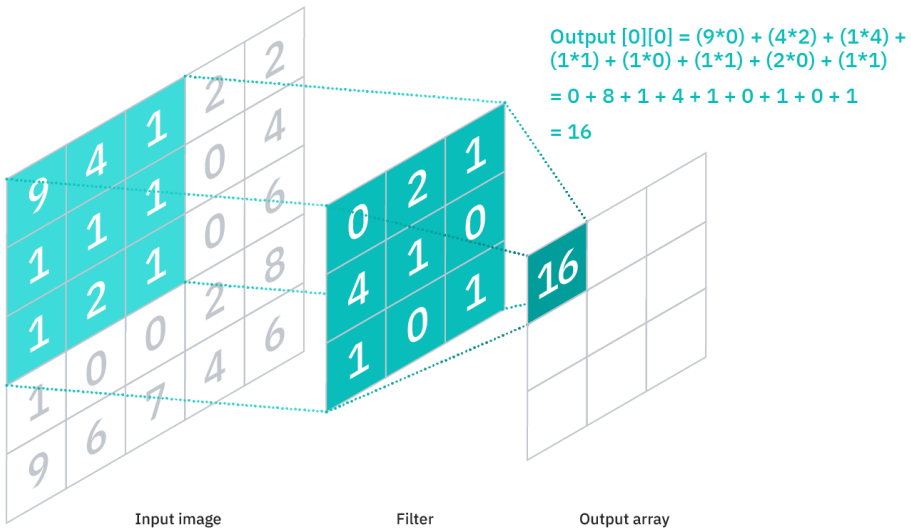


Figura 3-11. Capa convolucional [28]

Una vez se han obtenido estos mapas de características estos se analizan en capas posteriores del tipo de las que hemos visto anteriormente, pero utilizando ya imágenes

más fáciles de procesar que las entradas iniciales o que resaltan cualidades buscadas en las imágenes, que dependerán de los filtros elegidos. Hay que destacar que, en este tipo de redes, además de los pesos y sesgo de cada neurona, se deberán entrenar los coeficientes de cada uno de los filtros empleados.

3.2. Fundamentos del entrenamiento

El proceso de entrenamiento de una red neuronal lo que busca es optimizar parámetros internos de la red (como los pesos o el sesgo) adaptándose a la información presente en la base de datos de entrenamiento. A continuación, se presenta una explicación general de la metodología empleada para llevar a cabo esta tarea.

3.2.1. Descenso del gradiente

El descenso del gradiente es el algoritmo más utilizado para entrenar redes neuronales. Básicamente es un algoritmo que busca los coeficientes adecuados de una función para minimizar una función de coste.

En el caso de una red neuronal, los coeficientes son los parámetros internos de la red y la función de coste una función, elegida en base al tipo de problema concreto, que evalúa la diferencia entre las predicciones de nuestra red y las salidas reales. Es el valor que produce la función de coste lo que se trata de minimizar en todo momento en el entrenamiento.

El algoritmo calcula el gradiente como la derivada multivariable de la función de coste con respecto a los parámetros de la red. Este gradiente puede verse como la pendiente de la función de coste en el punto en el que nos encontramos actualmente. Desde un punto de vista matemático nos ofrece un vector que indica la dirección y el sentido en el que la función de coste aumenta con mayor rapidez, por tanto, se busca avanzar en sentido opuesto para tratar de minimizarla.

De todas maneras, el cálculo de este gradiente no es algo para nada trivial ya que tendremos muchos parámetros o dimensiones y además la función de coste presentará mínimos locales que pueden equivocarse con el mínimo global. Para esto se hace uso del algoritmo de *back-propragation*, explicado en la Sección 3.2.2.

Una vez tenemos el vector del gradiente, se actualizan los distintos parámetros de la red restando su valor actual con el valor del gradiente correspondiente, multiplicado este por una tasa de aprendizaje o *learning rate*. Esta tasa de aprendizaje se explica en la Sección 3.3.2.

Más información sobre el descenso del gradiente puede encontrarse en [10].

3.2.2. *Back-propagation*

A la de hora de optimizar la red neuronal mediante el método del descenso del gradiente, debemos solucionar el problema de como determinar la influencia de la variación de un parámetro, por ejemplo, de la primera capa de la red, en el coste final.

Este problema es relevante dado que ese cambio influye a su vez en todas las demás neuronas de las capas sucesivas. Esto se puede afrontar actualmente gracias al algoritmo de *back-propagation* (propagación hacia atrás).

Este método se basa en calcular las derivadas parciales de la última función de coste a la salida de la red en función de los parámetros de la capa final únicamente. Estos cálculos resultan relativamente sencillos debido a la regla de la cadena.

Calculadas las derivadas, podemos avanzar a la capa anterior obteniendo nuevamente las derivadas parciales de la función de coste, aunque esta vez con respecto a los parámetros de la nueva capa. Estos cálculos en parte ya estarán resueltos gracias a la regla de la cadena nuevamente. Esto proceso se seguirá hasta llegar al inicio de la red.

Más información sobre este algoritmo puede encontrarse en [10].

3.2.3. *Overfitting y Underfitting*

Un concepto que se repite bastante a la hora de entrenar una red neuronal es buscar que la red sea capaz de generalizar [29]. La generalización hace referencia a como de bien aplica la red neuronal los conceptos aprendidos en la fase de entrenamiento sobre nuevos datos no vistos anteriormente. Obviamente que la red sea capaz de realizar esta generalización es uno de los principales objetivos de todo proceso de *ML*. Esto muchas veces es complicado de conseguir, pudiéndose apreciar principalmente dos tendencias a la hora de obtener una generalización pobre:

- ***Underfitting:*** El *underfitting* hace referencia a un modelo que no es capaz de aprender correctamente la base de datos de entrenamiento por lo que su rendimiento en nuevos datos también será bastante deficiente.
- ***Overfitting:*** El *overfitting* ocurre cuando la red aprende la base de datos de entrenamiento con un nivel de detalle tan elevado que afecta negativamente al funcionamiento del algoritmo. Esto es debido a que el modelo coge el ruido o fluctuaciones aleatorias de la base de datos como conceptos, siendo esto no aplicable a nuevos datos por lo que limita su capacidad de generalizar.

3.3. Hiperparámetros

Los hiperparámetros son variables que determinan la estructura de nuestra red y la forma en la que esta es entrenada [30]. Este tipo de parámetros son fijados antes del entrenamiento y condicionan el funcionamiento de toda la red. Es por esto que su elección se trata de un apartado de gran importancia.

3.3.1. Hiperparámetros relativos a la estructura de la red

Dentro de estos podemos diferenciar entre las siguientes variables:

- **Número de capas ocultas y unidades que las forman**

Se trata de las capas que se encuentran entre las capas de entradas y salidas.

El aumento de unidades en estas capas puede aumentar la precisión. Por el contrario, un número demasiado reducido de estas puede producir *underfitting*.

- ***Dropout*** [31]

Se trata de una técnica de regularización que evita el *overfitting* aumentando así la generalización de la red. Una forma de evitar este *overfitting* de la red sería utilizar todas las posibles estructuras de una red en la misma base datos y luego utilizar la media de todas las predicciones.

Por su parte el *dropout* es una técnica que trata de aproximar el entrenamiento de un considerable número de redes neuronales con diferentes estructuras en paralelo. Esto se obtiene ignorando de manera aleatoria algunas salidas de cada capa. Esto consigue hacer ver a la capa como un número de nodos y conectividad diferente con respecto a la capa anterior, teniéndose una "visión" diferente de la capa en cada actualización del proceso de entrenamiento.

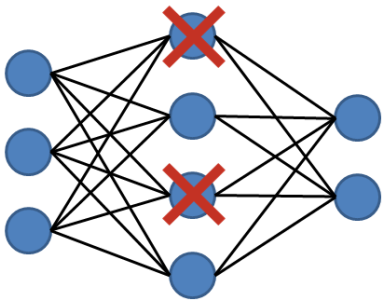


Figura 3-12. Esquema del proceso del *dropout* [30]

- **Inicialización de pesos de la red**

Es aconsejable utilizar diferentes esquemas de inicialización de pesos en función de la función de activación utilizada en cada capa. Por lo general, la distribución uniforme es la más utilizada.

- **Función de activación** [32]

La función de activación es la que define de que manera va a ser transformada la suma ponderada de las entradas en una salida en el nodo de una capa. Muchas de estas funciones son no-lineales para permitir a la red aprender funciones más complejas según sea el problema.

Las funciones de activación más comunes y que van a ser utilizadas en el presente trabajo se recogen a continuación.

- **Función de activación Lineal**

También conocida como la "no activación" ya que la salida de esta función es proporcional a la entrada, no cambia nada simplemente devuelve el valor que se le ha dado. Esta caracterizada por la Ecuación (3-3) y viene representada en la Figura 3-13.

$$f(x) = x \tag{3-3}$$

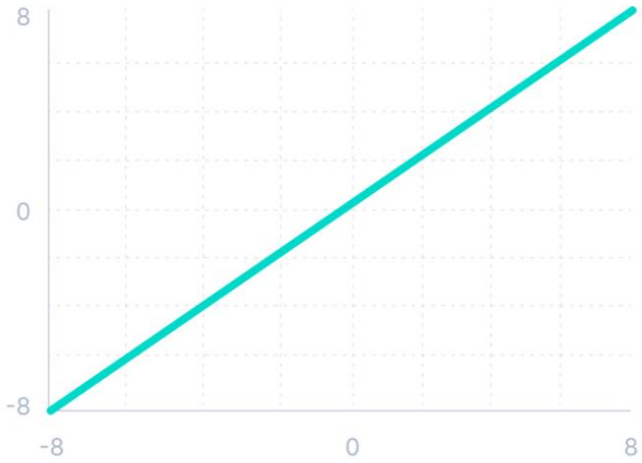


Figura 3-13. Función de activación lineal [33]

- **Función de activación Sigmoide**

Esta función transforma cualquier valor real que reciba como entrada en un valor que se encuentre entre 0 y 1. Cuanto mayor sea la entrada más cercana será la salida a 1 y viceversa. Está caracterizada por la Ecuación (3-4) y viene representada en la Figura 3-14.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3-4}$$

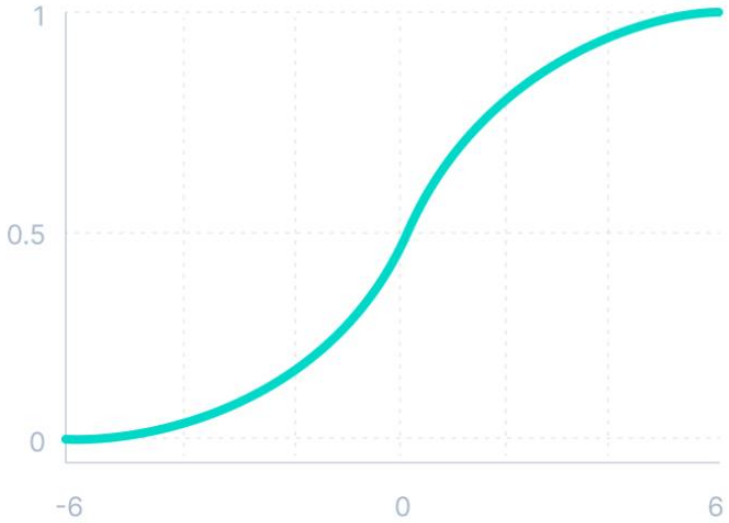


Figura 3-14. Función de activación Sigmoide [33]

- **Función Tangente Hiperbólica**

Se trata de una función similar a la anterior con la diferencia de que produce una salida en un rango entre -1 y 1. Cuanto más grande sea el valor de entrada más cercana será la salida correspondiente a 1 y viceversa. Esta caracterizada por la Ecuación (3-5) y viene representada en la Figura 3-15.

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{3-5}$$

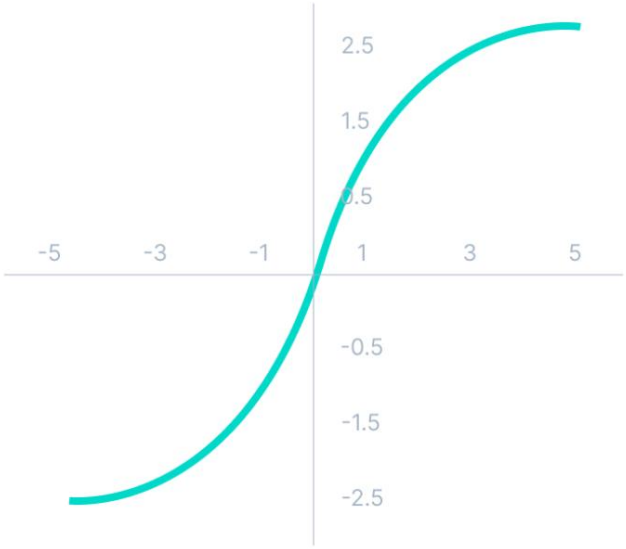


Figura 3-15. Función de activación Tanh [33]

- **Función de activación ReLu**

Su nombre completo es *Rectified Linear Unit*. Es un tipo de función de activación bastante utilizada actualmente que se basa en una función lineal para entradas mayores que 0 y un valor de 0 (lo que se puede considerar como que la neurona es desactivada) para entradas inferiores. Esta caracterizada por la Ecuación (3-6) y viene representada en la Figura 3-16.

$$f(x) = \max(0, x) \tag{3-6}$$

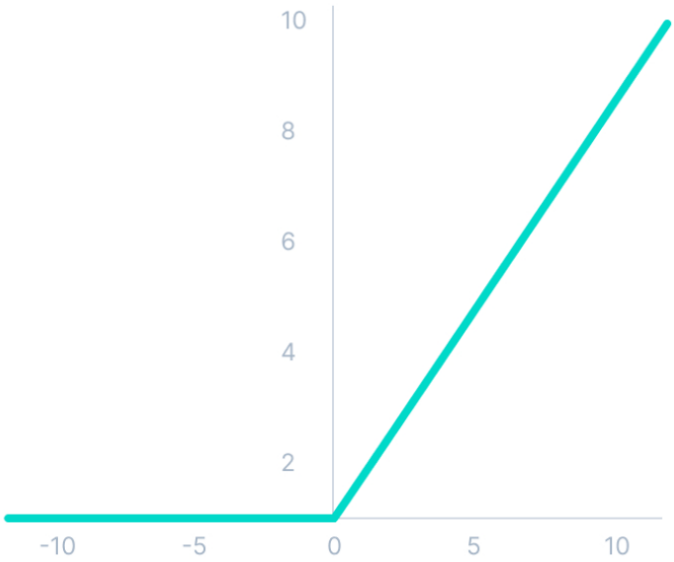


Figura 3-16. Función de activación ReLu [33]

- **Función de activación Softmax**

La función Softmax funciona como una combinación de múltiples funciones Sigmoides. Calcula la probabilidad relativa de cada una de las clases que se estén estudiando, siendo la suma de todas ellas igual a 1. Es por esta característica que suele ser la función de activación utilizada al final de una red diseñada para un problema de clasificación multi clase. Esta función está definida matemáticamente por la Ecuación (3-7).

$$f(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3-7}$$

A la hora de elegir una función de activación tenemos que diferenciar entre capas ocultas y capas de salida que realizan la predicción. Por lo general todas las capas ocultas presentarán la misma función de activación y esta dependerá del tipo de red que

estemos diseñando, por otro lado, la función de activación de las capas de salida dependerá del tipo de problema que estemos resolviendo (clasificación, regresión, etc).

A pesar de que existen multitud de tipos de funciones de activación, existen ciertas de ellas que son con diferencia las más utilizados. Para las capas ocultas la clasificación se recoge en la Figura 3-17.

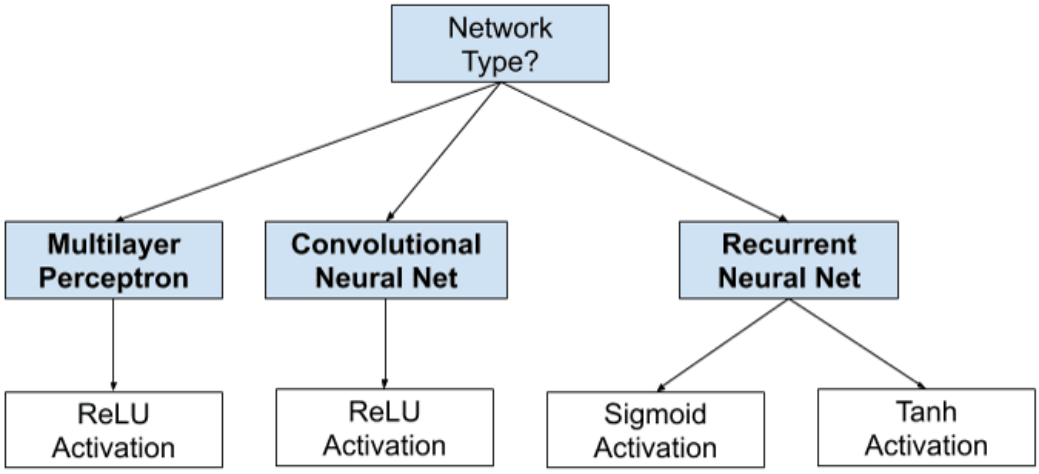


Figura 3-17. Esquema de elección para capas ocultas [32]

Por otro lado, para las capas de salida generalmente se utilizan según la Figura 3-18.

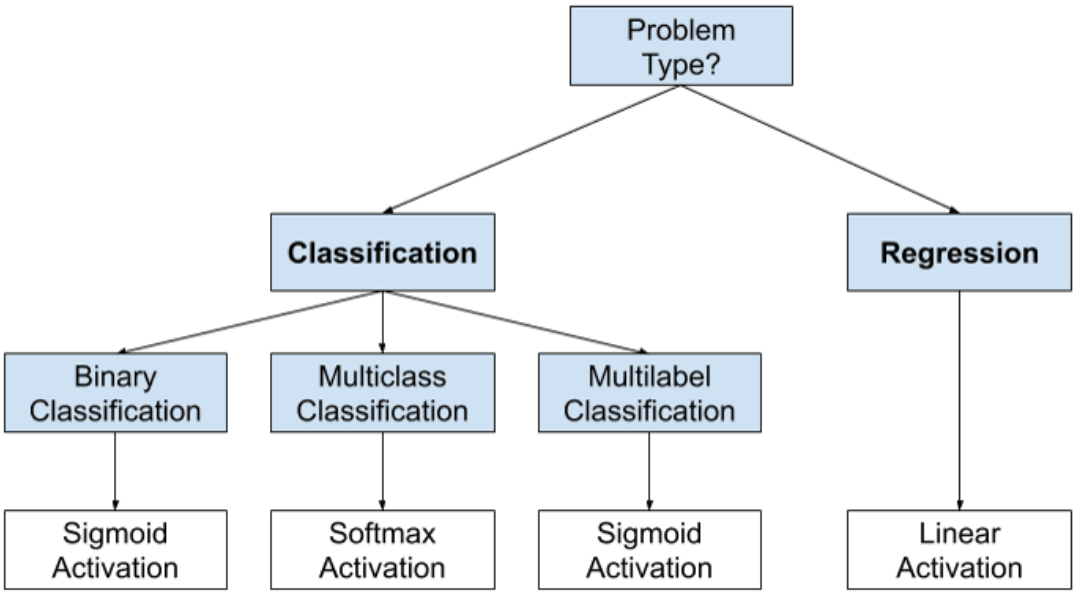


Figura 3-18. Esquema de elección para capas de salida [32]

3.3.2. Hiperparámetros relativos al entrenamiento de la red

Destacan entre estos los siguientes:

- ***Learning rate***

El ratio de aprendizaje o *learning rate*, es el margen que tienen los pesos de la red durante el entrenamiento cada vez que estos son ajustados. Este parámetro suele ser un valor pequeño que está entre 0 y 1.

Generalmente un valor grande de *learning rate* supone que la red aprende a gran velocidad, pero probablemente no alcanzando una solución demasiado óptima. Por otro lado, un *learning rate* más bajo probablemente llegará a una distribución de pesos óptima, aunque el tiempo computacional es más elevado.

De todas maneras, en los extremos, los valores grandes desembocarán en valores muy oscilantes y un comportamiento muy pobre después en el testeo y un valor demasiado pequeño puede llegar a no converger a una solución. En la Figura 3-19 se ejemplifica el efecto de valores extremos de *learning rate* a la hora de realizar el descenso del gradiente sobre la función de coste comentado en la Sección 3.2.1.

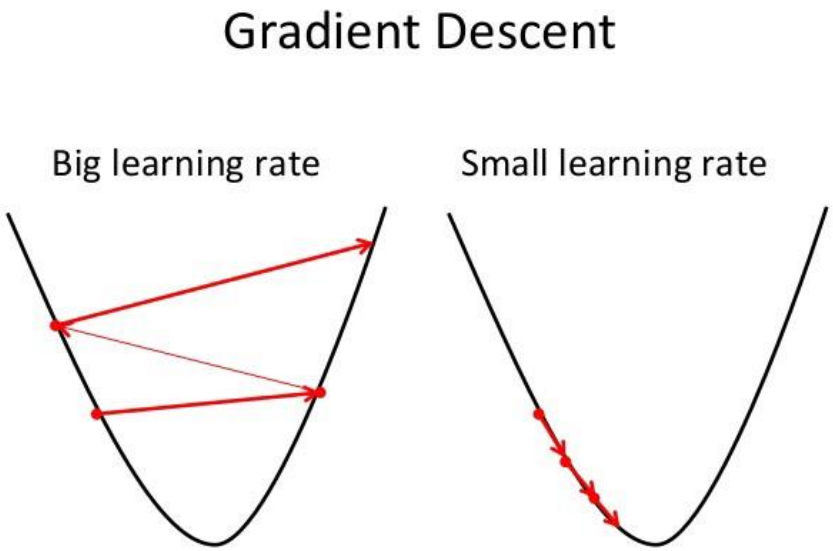


Figura 3-19. Diferencias producidas en el descenso del gradiente por el *learning rate* [30]

- ***Batch size***

Hace referencia al número de muestras de entrenamiento que se utilizan en la estimación del error para reajustar los parámetros de la red. Cuanto mayor sea este número de muestras de entrenamiento mejor será la estimación del error, pero más lento

será el método y viceversa. Podemos diferenciar entre 3 tipos de métodos en función del *batch size*:

- ***Batch Gradient Descent*:** el valor del *batch* se fija al número total de muestras de entrenamiento de la base de datos.
- ***Stochastic Gradient Descent*:** el valor del *batch* se fija en 1.
- ***Minibatch Gradient Descent*:** el valor del *batch* se fija entre 1 y el total de muestra de la base de datos de entrenamiento.

- ***Momentum***

Es un parámetro que ayuda al algoritmo a identificar la dirección en la función del problema hacia la que se debe avanzar en la optimización basándose en las actualizaciones anteriores. Los valores de *momentum* suelen estar entre 0.5 y 0.9, estando la nueva actualización de pesos más influenciada por las anteriores a mayor sea este valor.

- **Número de *epochs***

Hace referencia a el número de veces que la base de datos de entrenamiento al completo es suministrada a la red durante el entrenamiento. Debe ser un valor suficiente para que la red pueda aprender adecuadamente los datos que recibe. Por lo general el número de *epochs* debe ser incrementado hasta que la precisión de la validación empiece a empeorar (*overfitting*).

3.4. Optimización de hiperparámetros

Como hemos podido ver la elección de los hiperparámetros es un proceso que condiciona enormemente el funcionamiento de nuestra red por lo que su elección deberá ser una parte importante de nuestro estudio.

Aunque podemos saber que efecto puede tener la variación de cada hiperparámetro individualmente, elegir la combinación de estos que mejor funcionarán para nuestro problema es una tarea complicada. Por lo general, la mejor aproximación suele ser evaluar diferentes subconjuntos de hiperparámetros y ver cual es el que mejor funciona, esto es conocido como optimización de hiperparámetros.

- ***Grid Search*** [34]

Este método consiste en evaluar el funcionamiento de distintas combinaciones de hiperparámetros que especifica el usuario para ver cual es la que más nos conviene. Se trata de un modelo sencillo y fácil de implementar, aunque con un coste computacional muy alto ya que hay que evaluar la red repetidas veces.

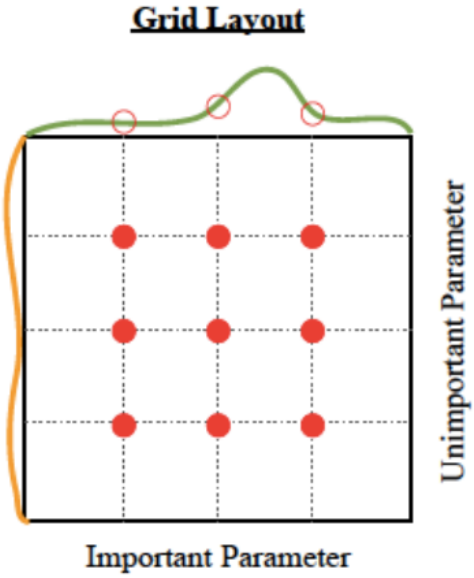


Figura 3-20. *Grid search* [34]

- ***Random search*** [34]

El método de *Random search* es muy similar al *Grid search*, salvo por el hecho de que las combinaciones de hiperparámetros son aleatorias entre los límites especificados por el usuario. Esto permite caracterizar mejor el comportamiento de la variación de cada parámetro. Se trata de un modelo más eficiente, aunque aun así presenta un alto coste computacional para problemas exigentes.

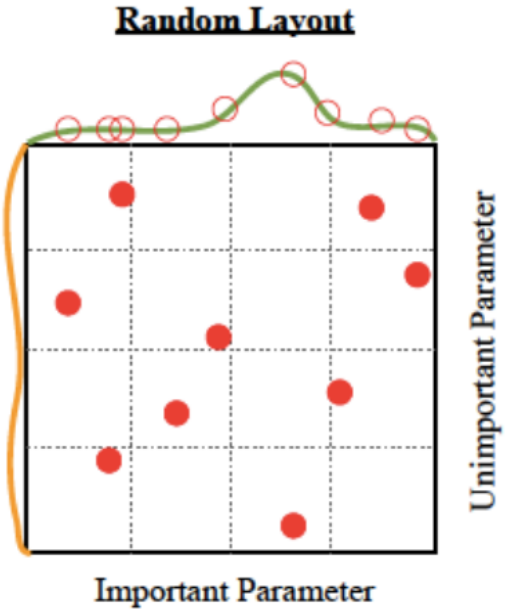


Figura 3-21. *Random search* [34]

- **Bayesian Hyperparameter optimization** [35]

Este método se puede resumir en construir un modelo de probabilidad de la función de coste y utilizarlo para seleccionar los hiperparámetros mas prometedores para evaluarlos en la función de coste verdadera.

La principal ventaja con respecto a los anteriores consiste en que con este método se tiene en cuenta los resultados de evaluación anteriores, lo que nos permite concentrarnos en las zonas donde se estén encontrando los mejores resultados.

El modelo de probabilidad se conoce como *surrogate* o sustituto de la función de coste. Esta función es más sencilla de optimizar que la función de coste original. El ciclo de trabajo podría resumirse en los siguiente:

1. Crear un modelo de probabilidad *surrogate* de la función de coste.
2. Buscar los hiperparámetros que mejor funcionan en la función *surrogate*.
3. Utilizar estos hiperparámetros en la verdadera función de coste.
4. Se actualiza el modelo *surrogate* incorporando los nuevos resultados.
5. Se repiten los pasos 2-4 hasta llegar a las máximas iteraciones o el límite de tiempo.

Por decirlo de alguna manera, este método lo que prioriza es destinar algo más de tiempo en seleccionar los nuevos hiperparámetros para acabar realizando menos llamadas a la función de coste. De todas formas, este tiempo de más seleccionando los hiperparámetros es mucho menor que el que se destina a evaluar la función de coste.

En la Figura 3-22 podemos apreciar la estimación del modelo *surrogate* con solo dos evaluaciones.

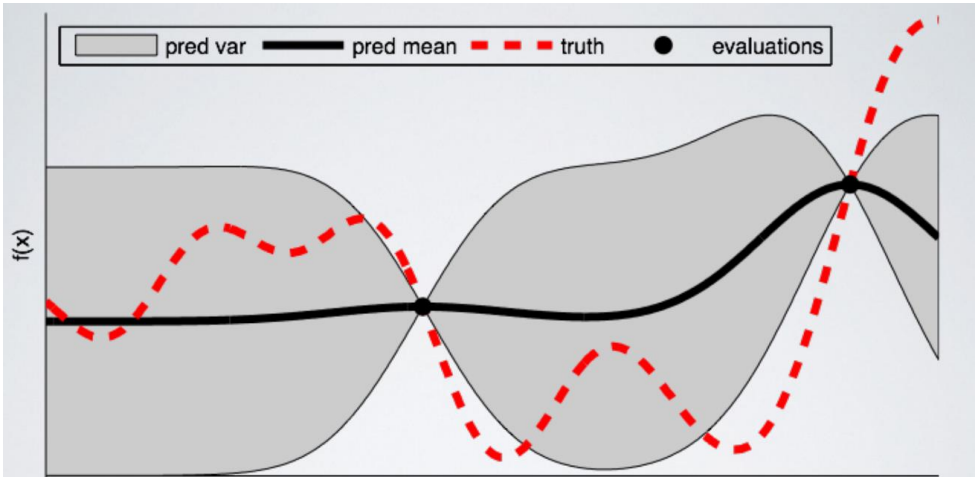


Figura 3-22. Estimación del modelo *surrogate* con 2 evaluaciones [35]

Se puede apreciar que la aproximación de la función de coste no es demasiado fiable. Pero comparándola con la Figura 3-23 tras 8 evaluaciones se pueden apreciar grandes diferencias:

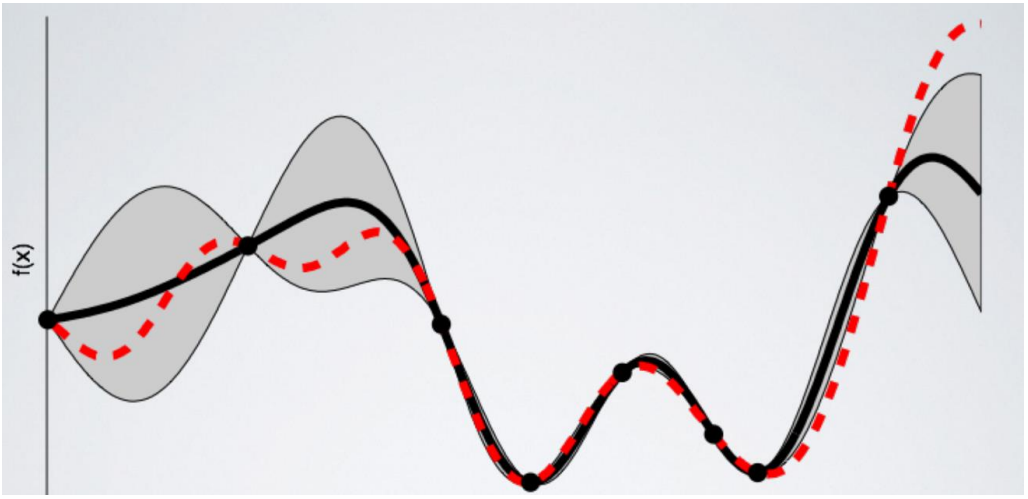


Figura 3-23. Estimación del modelo *surrogate* con 8 evaluaciones [35]

3.5. *Clustering*

La agrupación o *clustering* es un tipo de tarea cuya finalidad principal consiste en agrupar conjuntos de objetos sin etiquetar, buscando crear diferentes subconjuntos de datos conocidos como *clusters* [36]. Estos *clusters* se caracterizan por estar formados por elementos que, en el foco del análisis, resultan similares entre sí y que poseen características distintivas a otros elementos del conjunto de datos que por su parte pueden formar otro *cluster* independiente. En la Figura 3-24 vemos un ejemplo de *clustering*.

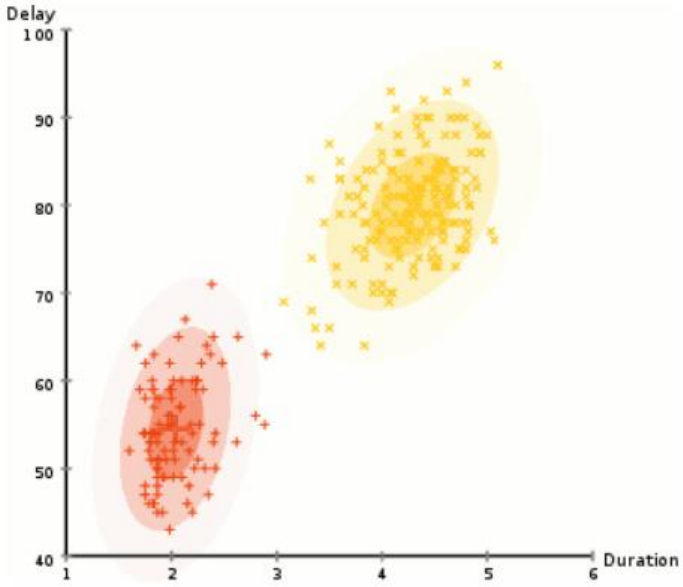


Figura 3-24. Problema típico de *clustering* [37]

Este método se engloba dentro del grupo del *ML* conocido como aprendizaje no supervisado. Esto es así debido a que no comprueba sus soluciones con un valor que nosotros proveemos al algoritmo, sino que toda la optimización la realiza el propio programa. Aunque existen diversos métodos, generalmente esto se consigue estableciendo una métrica que evalúa los *clusters* propuestos por el algoritmo y en sucesivas iteraciones se trata de minimizar el error.

Algunas técnicas de estas técnicas son el *hierarchical clustering*, *k-means*, métodos de densidad, *spectral clustering* o *k-nearest*.

En este trabajo se ha decidido utilizar el método *k-means*. Se trata de un método que divide los datos en "k" *clusters* mutuamente exclusivos. Este método requiere que el usuario establezca el número de *clusters* deseados como entrada del método. Su funcionamiento se basa en medir la suma de las distancias entre los datos y los centroides y, en un proceso iterativo, se elige la distribución que ofrezca la suma más pequeña.

4. Materiales y métodos

En este capítulo se recogen los diferentes materiales y métodos que han sido utilizados para la realización de este trabajo.

Por un lado, tenemos la base de datos sobre la que se elabora todo el análisis, la cual es una conocida base de datos en los estudios de mantenimiento predictivo, contrastada y que recoge información variada relativa a la operación de motores Turbofán.

Por otro lado, tenemos el software MATLAB a partir del cual se ha realizado el análisis de la base de datos anterior. Este software ha sido elegido por ser un lenguaje con el que ya estábamos familiarizados, aparte de por la gran cantidad de opciones y herramientas de *DL* que ofrece MATLAB dentro de su variada oferta.

4.1. Base de datos utilizada

Para estudiar la viabilidad de los métodos de *ML* en el mantenimiento predictivo de motores aeronáuticos, nos hemos servido de una base de datos libre proporcionada por *Prognostics CoE at Nasa Ames* y conocida como "*Turbofan engine degradation simulation data set*" [5]. Se trata de una base de datos en la que se monitoriza el funcionamiento de una serie de motores hasta el fin de su vida útil, donde se busca caracterizar a través de las señales de diferentes sensores del motor, la evolución de los fallos que se producen en estos. Existen cuatro subconjuntos diferenciados, no presentando las mismas condiciones de operación en cada uno de ellos para poder tener una información más variada y completa.

La simulación de la degradación de los motores en esta base de datos ha sido realizada mediante el software C-MAPSS (*Comercial Modular Aero-Propulsion System Simulation*) [38].

La creación de esta base de datos se debió, entre otras motivaciones, a proporcionar una información libre con la que los investigadores pudiesen comparar sus propuestas entre ellos, dado que la mayoría de las bases de datos de este tipo no son de libre acceso debido a que las compañías no las comparten por razones de competitividad. Es por esto por lo que esta base de datos fue proporcionada para la competición de PHM (*Prognostics and Health Management*), donde se buscaba obtener la mejor estimación de la vida útil restante de un motor utilizando la información histórica proporcionada únicamente, con independencia de los procesos físicos subyacentes.

4.1.1. Herramienta C-MAPSS

Se trata de una herramienta que sirve para simular motores Turbofán comerciales [39]. El código sobre el que se estructura es una combinación de MATLAB y Simulink, que permite, desde una serie de interfaces gráficas, customizar la simulación a través de un gran número de posibilidades como pueden ser condiciones de operación, factores ambientales, etc.

C-MAPSS presenta un modelo de motor de la clase de 90000 lb de empuje, incluyendo en el paquete un modelo atmosférico que funciona en altitudes desde el nivel del mar a 40000 ft, números de Mach de 0 a 0.9 y temperaturas a nivel del mar desde -60 a 103 ºF. También existe la posibilidad de hacer uso de un sistema de control de potencia que permite simular el motor con una amplia variedad de niveles de potencia a lo largo de todo el rango de condiciones de operación.

En el esquema de la Figura 4-1 vienen representados los principales elementos del modelo del motor y el diagrama de flujo de la Figura 4-2 presenta como las diferentes subrutinas están implementadas en la simulación.

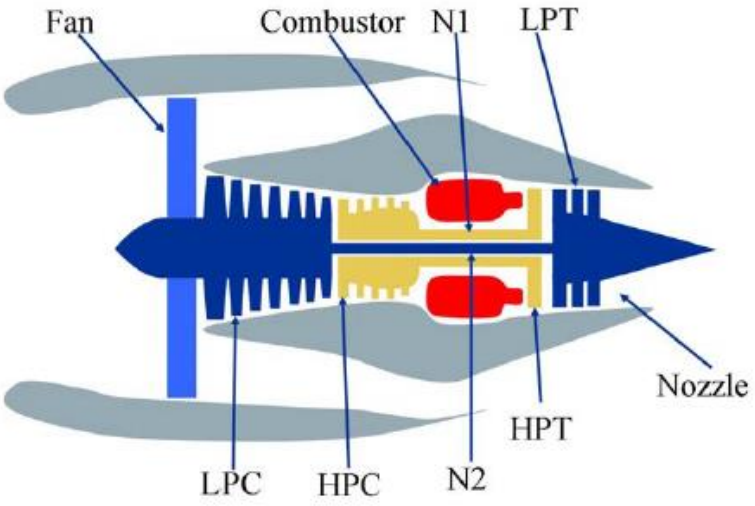


Figura 4-1. Modelo del motor simulado [15]

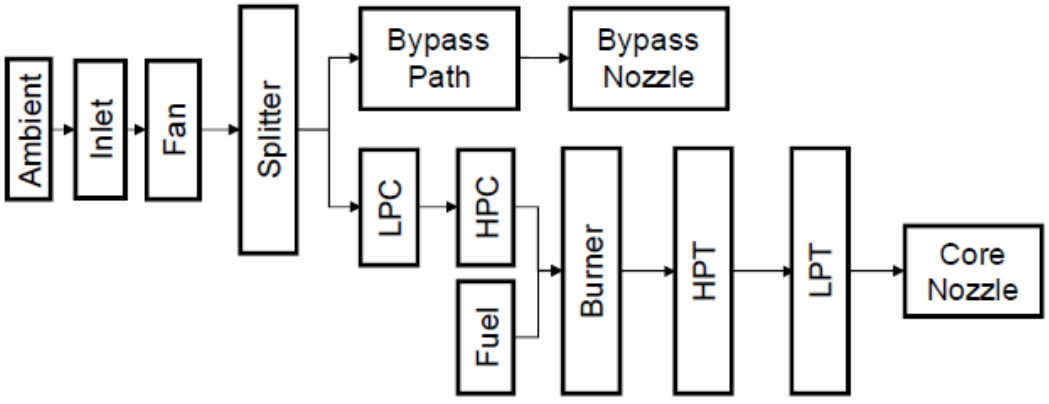


Figura 4-2. Diagrama de flujo de la simulación [15]

El programa C-MAPSS puede ser operado tanto en bucle abierto (sin ningún tipo de controlador) o en bucle cerrado (con el sistema de control en operación). Para la

creación de la base de datos utilizada en el presente trabajo, se operó exclusivamente en bucle cerrado.

El programa recibió 14 entradas para producir diversas salidas. Estos valores de entrada vienen recogidos en la Figura 4-3.

Name	*Symbol*
Fuel flow	W_f
Fan efficiency modifier	fan_eff_mod
Fan flow modifier	fan_flow_mod
Fan pressure-ratio modifier	fan_PR_mod
LPC efficiency modifier	LPC_eff_mod
LPC flow modifier	LPC_flow_mod
LPC pressure-ratio modifier	LPC_PR_mod
HPC efficiency modifier	HPC_eff_mod
HPC flow modifier	HPC_flow_mod
HPC pressure-ratio modifier	HPC_PR_mod
HPT efficiency modifier	HPT_eff_mod
HPT flow modifier	HPT_flow_mod
LPT efficiency modifier	LPT_eff_mod
HPT flow modifier	LPT_flow_mod

Figura 4-3. Entradas del programa C-MAPSS [15]

Estas entradas incluyen consumo de combustible y un conjunto de 13 parámetros referentes al correcto funcionamiento del motor que permiten simular el deterioro y fallo en cualquiera de los componentes rotativos del motor (Fan, LPC, HPC, HPT y LPT).

Por su lado, las salidas del simulador se componen de un total de 21 variables que miden la respuesta del sistema y de una serie de márgenes de operación del motor. Estos márgenes son utilizados para calcular el índice de salud del motor y no son accesibles para los usuarios de la base de datos de manera explícita. En la Figura 4-4 se recogen las salidas comentadas.

Symbol	*Description*	*Units*
Parameters available to participants as sensor data		
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	--
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	--
farB	Burner fuel-air ratio	--
htBleed	Bleed Enthalpy	--
Nf_dmd	Demanded fan speed	rpm
PCNfR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s
Parameters for calculating the Health Index		
T48 (EGT)	Total temperature at HPT outlet	°R
SmFan	Fan stall margin	--
SmLPC	LPC stall margin	--
SmHPC	HPC stall margin	--

Figura 4-4. Salidas del programa C-MAPSS [15]

4.1.2. Estructura de los datos

La base de datos está dividida en 4 subgrupos en formato ".txt", presentando cada uno unas particularidades diferentes. Estos subconjuntos están formados por una cierta cantidad de series temporales multivariable, donde cada serie temporal se corresponde con un motor diferente. Por otro lado, cada uno de estos subgrupos está dividido en entrenamiento y testeo, así como incluyendo cada uno un archivo que contiene la RUL correspondiente a la parte de testeo.

Por otra parte, cada uno de los motores presenta un cierto grado de desgaste inicial y variaciones de fabricación encontrándose todas ellas dentro de los límites de la normalidad y siendo desconocidas por el usuario.

En los datos se refleja que el motor se encuentra operando con normalidad al inicio de cada serie temporal, pero que presenta un fallo en un cierto momento. En el caso del entrenamiento, este fallo crece hasta provocar el fallo del sistema. Por otro lado, en la

parte de testeo lo que buscamos predecir es el número de ciclos restantes del motor, los cuáles vienen recogidos en los archivos RUL de cada subconjunto.

Cada fila de los archivos presenta una captura del funcionamiento del motor en la serie temporal. En el caso de las columnas, estas están divididas en 26 variables diferentes que se resumen en la clasificación siguiente:

1) **Número de unidad:** cada número hace referencia a un motor distinto.
2) **Tiempo (en ciclos):** hace referencia al ciclo en el que se encuentra el motor a lo largo de su operación.
3) **4) y 5):** hacen referencia a la configuración de la operación del motor.
6) **Sensor 1**

…

26) Sensor 21

En la Figura 4-5 se resumen las características de cada una de las bases de datos. Se puede apreciar claramente como las bases de datos presentan características ciertamente dispares tanto en tamaño como modos de fallo y operación.

	C-MAPSS			
Dataset	FD001	FD002	FD003	FD004
Engine units for training	100	260	100	249
Engine units for testing	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2
Training samples (default)	17,731	48,819	21,820	57522
Testing samples	100	259	100	248

Figura 4-5. Características de las bases de datos estudiadas [11]

4.2. MATLAB

4.2.1. Introducción y primeras aproximaciones

El software de MATLAB es una plataforma que abarca una enorme cantidad de ámbitos, no escapándosele en absoluto el del *ML*. Su oferta es variada y es posible encontrar desde aplicaciones ciertamente automatizadas que nos permiten utilizar modelos de aprendizaje automático sin que haga falta un conocimiento profundo de la materia, a toda una gama de funciones y herramientas que nos permiten desarrollar nuestro modelo de *ML* al milímetro y adaptarlo totalmente a nuestras necesidades. Además, estas capacidades de MATLAB son altamente compatibles con Simulink, herramienta con la cual se pueden generar datos para luego probar en los algoritmos de

inteligencia artificial desarrollados o para testear la implantación de estos en algún simulador que se encuentre en desarrollo.

Como una primera aproximación, MATLAB nos ofrece un variado conjunto de modelos de *ML*. Además, se ofrece acceso a una serie de aplicaciones para la utilización de estos modelos de manera rápida y que facilita en gran medida la realización de comparaciones de rendimiento entre ellos. En estas aplicaciones generalmente solo hace falta poseer la base de datos a analizar y elegir los modelos que queremos entrenar. Dentro de la propia interfaz de la aplicación podemos analizar y optimizar los modelos estudiados y, una vez estemos satisfechos, existe la posibilidad de exportarlos a código de MATLAB para ser más tarde utilizados en nuevos datos, *scripts*, etc.

Por un lado, existen aplicaciones que cuentan con diversos algoritmos de *ML*, como pueden ser la aplicación *Regression Learner* o *Classification Learner*, donde se pueden utilizar métodos como modelos de regresión lineal, SVM (*Support Vector Machine*), *Tree*, etc. En la Figura 4-6 se muestra un ejemplo de la interfaz que presenta este tipo de aplicaciones.

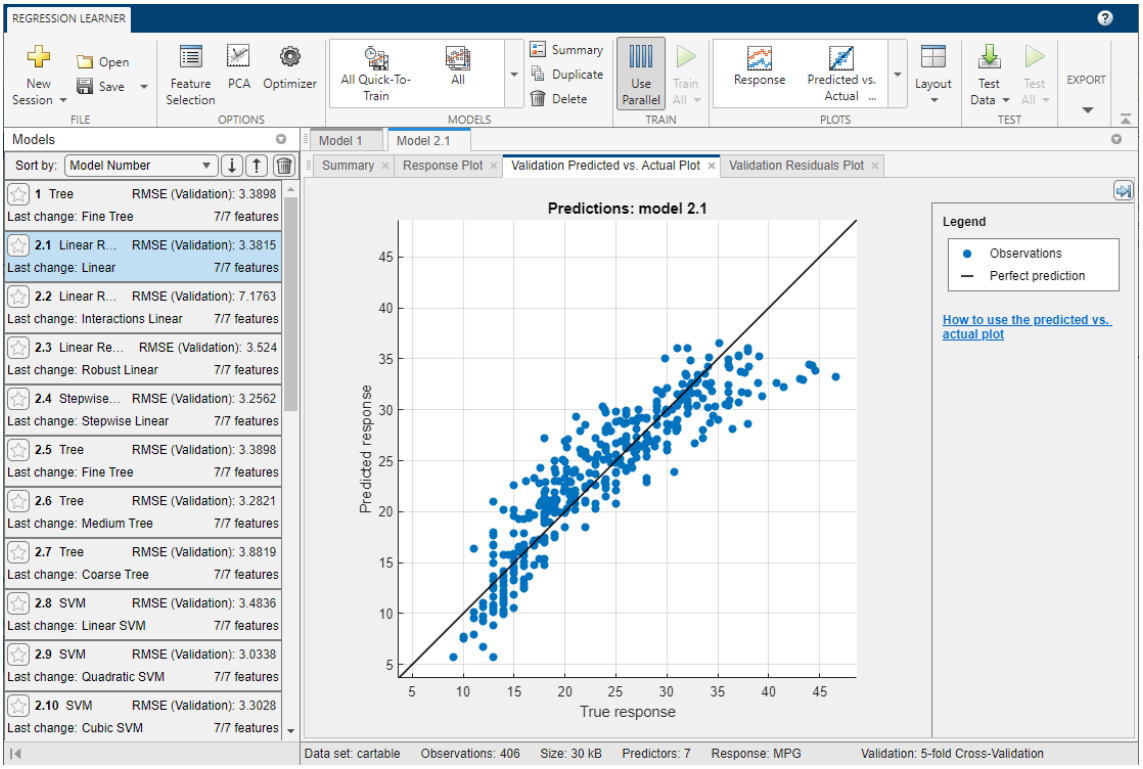


Figura 4-6. Interfaz gráfica aplicaciones de *ML* [40]

Por otro lado, MATLAB también presenta aplicaciones que nos permiten utilizar redes neuronales no profundas. Las redes neuronales no profundas presentan un bajo número de capas ocultas, teniendo solo 1 en estas aplicaciones comentadas, lo que limita la capacidad de la red de aprender conceptos complejos. Estas aplicaciones ofrecen modelos poco flexibles, en los que solo se nos permite cambiar ciertos hiperparámetros y opciones de entrenamiento de la red. Algunas de estas aplicaciones son *Neural Net*

Fitting o *Neural Net Time Series*. En la Figura 4-7 se recoge la interfaz gráfica típica de este tipo de aplicaciones.

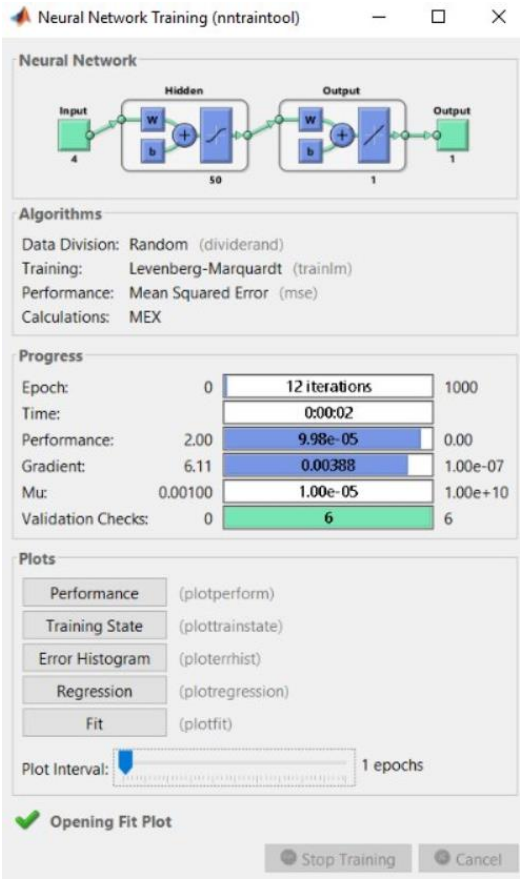


Figura 4-7. Ejemplo de interfaz gráfica para aplicaciones *DL* [40]

Diversos métodos de *clustering* también se encuentran en la carta de posibilidades. Existe la posibilidad de utilizar funciones con los diferentes métodos ya implementados. Entre sus posibilidades se encuentran:

- *Hierarchical clustering*.
- *K-means*.
- DBSCAN (*Density Based Spatial Clustering of Applications with Noise*).
- Modelo Gaussiano.
- *Spectral clustering*.

Debido a la complejidad de la tarea que nos ocupa en el presente proyecto, cobra especial interés la utilización de redes neuronales profundas o *DL*. Este tipo de algoritmos son capaces de modelizar sistemas muy complejos como es el caso que nos ocupa, lo que los hace un método más idóneo tal y como se ha visto en la Sección 2.4. La implementación de este tipo de redes se ha conseguido gracias a la extensión conocida como **"*Deep Learning Toolbox*"** con la que se ha desarrollado la mayor parte

de este trabajo. Esta extensión establece un marco que da acceso a una enorme cantidad de posibilidades para el desarrollo de redes neuronales de todo tipo.

4.2.2. *Deep Learning Toolbox*

Mediante esta *toolbox* podremos desarrollar nuestro código de manera totalmente flexible y libre. A las posibilidades habituales de programación de MATLAB (*scripts*, funciones, etc.), se unen las herramientas necesarias para diseñar, entrenar y evaluar de manera completa una red neuronal. En base a la programación utilizada en este trabajo podemos diferenciar diferentes partes o estructuras.

- **Creación de la arquitectura de la red neuronal**

MATLAB incluye en su oferta una gran variedad de funciones que nos permiten crear nuestras redes neuronales con los diferentes tipos de capas que busquemos implementar. Si bien es cierto que estas funciones poseen características muy variadas, se pueden agrupar por similitud en la Tabla 4-1. De todas formas, por las posibilidades de su aplicación, habría algún tipo de capas que podría encajar en más de una categoría. Más información en [41].

Tipo de capa	Función
Input Layers	Serían capas a situar en la posición inicial. Se encargan de recibir los datos que van a ser analizados por la red. Dependiendo del formato de estos datos, tenemos capas orientadas a datos secuenciales, imágenes 2D y 3D, etc.
Convolution and Fully Connected Layers	Entrarían en esta categoría las capas convolucionales y la capa MLP básica. Dentro de las convolucionales aparecen diferentes tipos distinguiendo entre 1D, 2D y 3D.
Sequence Layers	Dentro de este grupo se encontrarían las capas que propiamente tratan con datos secuenciales. Es en esta clasificación donde tenemos las conocidas redes recurrentes como LSTM, Bi-LSTM or GRU. Por otro lado, tenemos capas menos comunes y de carácter más específico realizan transformaciones sobre secuencias de imágenes o palabras, por ejemplo.
Activation Layers	Serían las funciones de activación que podemos añadir a nuestra red. Desde las más comunes como ReLu o Tanh, a otras más específicas como swish o leakyReLu.
Normalization, Dropout and Cropping Layers	Capas que nos permiten transformar, los datos a lo largo de la red para optimizar el entrenamiento, como puede ser el conocido *Dropout*.
Combination Layers	Capas utilizadas para recibir inputs de varias redes neuronales, realizando algún tipo de transformación sobre ellos.

Object detection Layers	Capas utilizadas para la detección de objetos con múltiples variantes. Están enfocadas al uso en problemas de *Computer Vision*, que se escapa del foco de este trabajo.
Output Layers	son las capas finales en la arquitectura de nuestra red que utilizaremos en función del tipo de problema que afrontemos, ya sea regresión, clasificación. También aparecen un cierto número de posibilidades para problemas de *Computer Vision*.

Tabla 4-1. Tipos de capas disponibles en MATLAB

Todas estas capas admiten la customización de sus parámetros, como la adaptación de los filtros en redes convolucionales o ajustar el número de neuronas de las capas. Si aún no encontramos exactamente una opción que satisfaga nuestras necesidades, MATLAB ofrece también la posibilidad de crear nuestras propias capas. Más información sobre esto en [42].

- **Entrenamiento y predicción con redes neuronales**

Las funciones de entrenamiento de MATLAB permiten entrenar las redes de manera sencilla y ofreciendo al usuario un gran control sobre el proceso. La estructura general consiste en proveer a la función de entrenamiento la base de datos, la arquitectura de nuestra red y una serie de opciones de entrenamiento que podemos customizar a nuestro gusto.

Estas opciones de entrenamiento son muy variadas y su uso dependerá bastante del tipo de problema y red neuronal que estemos estudiando. En el presente trabajo se han especificado apartados del entrenamiento como el tipo de *solver* empleado, número de *epochs*, tamaño del *batch*, *learning rate*, frecuencia de validación, etc. De todas formas, para más información sobre la totalidad de la oferta de MATLAB, puede consultarse [43].

El proceso de entrenamiento abre una interfaz gráfica donde es posible monitorear como este se está desarrollando, ofreciendo una visión general de los principales parámetros y pudiendo ser interrumpido en cualquier momento. En la Figura 4-8 se recoge la interfaz gráfica comentada.

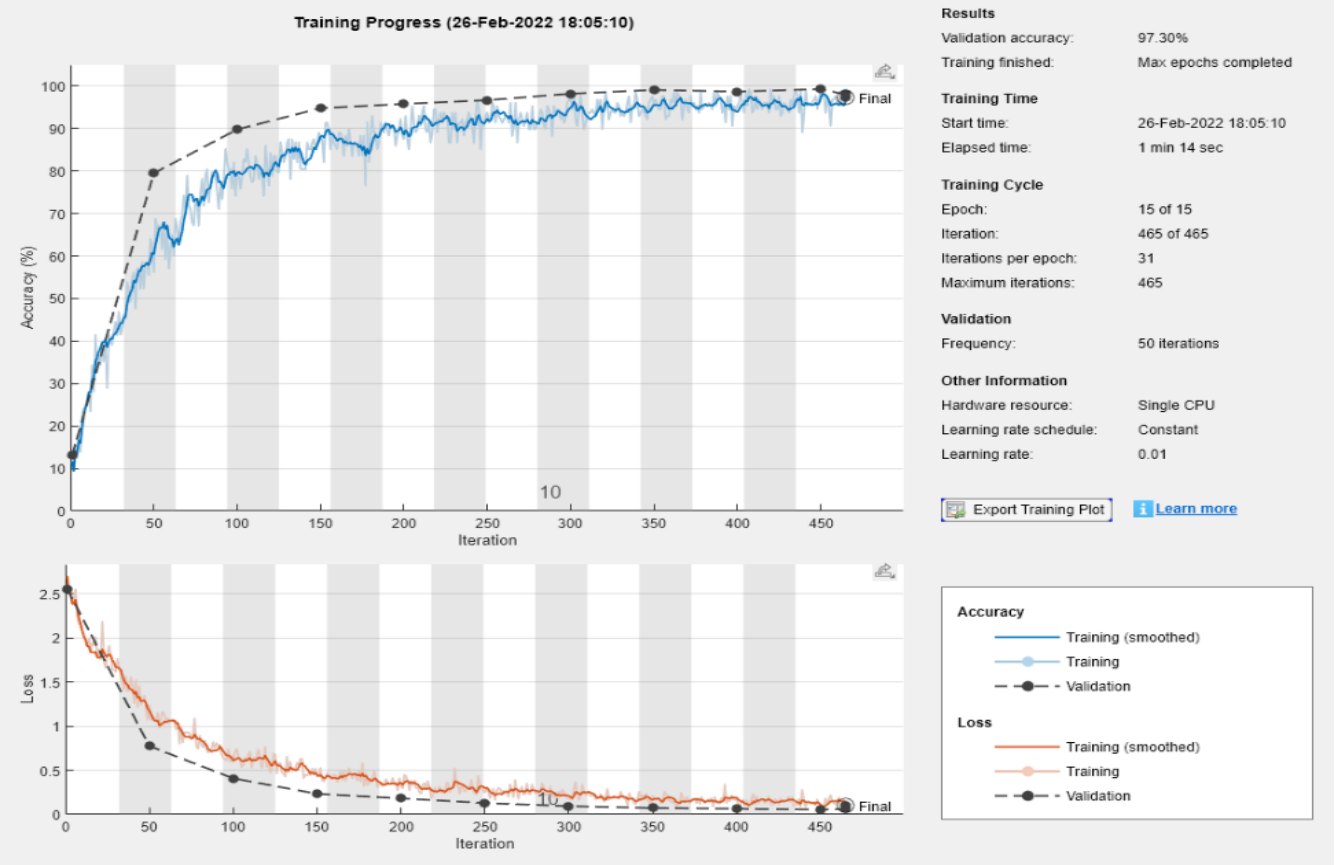


Figura 4-8. Interfaz gráfica de entrenamiento *Deep Learning Toolbox* [43]

Una vez entrenada la red, podemos realizar predicciones con ella utilizando las funciones de predicción. Estas funciones solo requerirán como inputs la red entrenada y la base de datos sobre la que realizar las predicciones.

- **<u>Supervisión y visualización de la activación de las capas</u>**

Una vez entrenada la red neuronal, MATLAB nos brinda funciones que nos permiten conocer como la red se comporta internamente para llevar a cabo las predicciones. Gracias a esto podemos ver como se procesan los datos capa por capa, lo cual puede brindarnos una información muy valiosa para entender que está pasando en el interior de nuestro algoritmo y así extraer nuevas ideas para optimizarlo.

Estas posibilidades van desde ver las imágenes que se procesan a lo largo de una red convolucional, o más aplicado a nuestro problema, ver como se interpretan las secuencias temporales del funcionamiento del motor. En la Figura 4-9 podemos ver una imagen extraída de una capa de una red convolucional.

Figura 4-9. Comparación entre un input y una imagen de una capa convolucional [40]

Además, la *toolbox* permite intercambiar modelos con TensorFlow [44] y PyTorch [45], las cuales son dos bibliotecas de código abierto cuya utilidad se basa en la creación de grafos computacionales y siendo extensamente utilizadas para la creación de redes neuronales, modelos de *ML*, etc. La utilización de modelos previamente entrenados también es posible, con arquitecturas bien conocidas como SqueezeNet [46] o GoogleNet [47], los cuales tenemos la posibilidad de modificar para adaptarlos mejor a nuestras necesidades.

Unido a lo anterior, la "*Deep Learning Toolbox*" cuenta con una cierta variedad de aplicaciones asociadas que nos ayudan a automatizar y simplificar el proceso de diseño y optimización de nuestras redes neuronales. Entre estas aplicaciones destacan las siguientes:

- ***Experiment Manager***

Se trata de una aplicación que permite crear experimentos de *DL*, entrenando redes neuronales bajo diferentes condiciones iniciales y así poder comparar el rendimiento entre unas y otras de manera fácil y rápida.

Alguna de las formas comunes de utilizar esta herramienta son las siguientes:

  - Optimización de los hiperparámetros de la red mediante barrido o optimización bayesiana, lo cual nos permite automatizar el proceso introduciendo simplemente los límites de la optimización.
  - Comparar resultados utilizando diferentes bases de datos, redes neuronales, etc.

La forma de utilizar esta herramienta presenta diferentes fases:

1. Se crea una función que recoja la base de datos de la red, su arquitectura y sus opciones de entrenamiento.

2. Dentro de esta función, los parámetros de esta red que quieran ser modificados a lo largo de los experimentos deben ser recogidos en una estructura de datos. Esto puede ir desde cambiar un parámetro como el *learning rate*, a arquitecturas enteras de red o bases de datos.

3. Se carga esta función en el *Experiment Manager*, establecemos el tipo de análisis que queremos realizar (barrido u optimización bayesiana), imponemos los límites entre los que variarán los parámetros a modificar y finalmente establecer métricas que nos interesen que evalúen los resultados o, en el caso de la optimización bayesiana, que métrica ha de tratar de optimizar el algoritmo.

4. Una vez ponemos a funcionar el programa, las diferentes iteraciones con unas combinaciones específicas de los parámetros seleccionados se irán ejecutando y podremos ir viendo sus resultados. También existe la posibilidad de ver el gráfico de entrenamiento de cada iteración entre otras.

El *Experiment Manager* cuenta con diversos modelos predefinidos, aunque también permite la customización de tus experimentos ofreciendo una gran flexibilidad a las necesidades de cada proyecto. Un ejemplo de la interfaz puede verse en la Figura 4-10. Más información sobre esta herramienta puede encontrarse en [48].

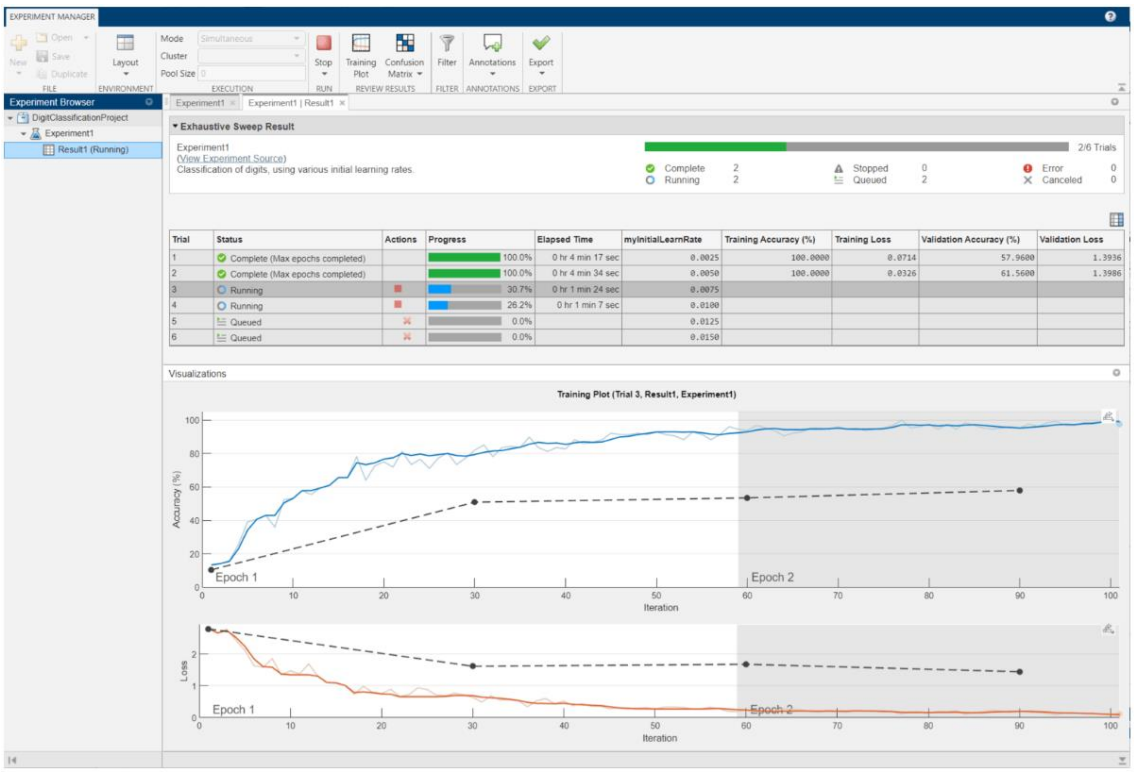


Figura 4-10. Interfaz gráfica *Experiment Manager* [48]

- ***Deep Network Designer***

Esta aplicación ofrece una interfaz interactiva donde diseñar nuestras redes neuronales capa por capa, adaptando los parámetros según mejor interese.

Es una buena opción si preferimos diseñar en un inicio la arquitectura de nuestra red de una manera más esquemática y visual, pudiendo luego más tarde pasarla a formato de código. También es una buena opción de cara a utilizar *Transfer Learning*, donde importamos el esquema de una red ya existente para luego modificarlo a nuestro gusto.

Un ejemplo de la interfaz de esta aplicación se presenta en la Figura 4-11. Más información sobre las posibilidades de esta herramienta puede encontrarse en [49].

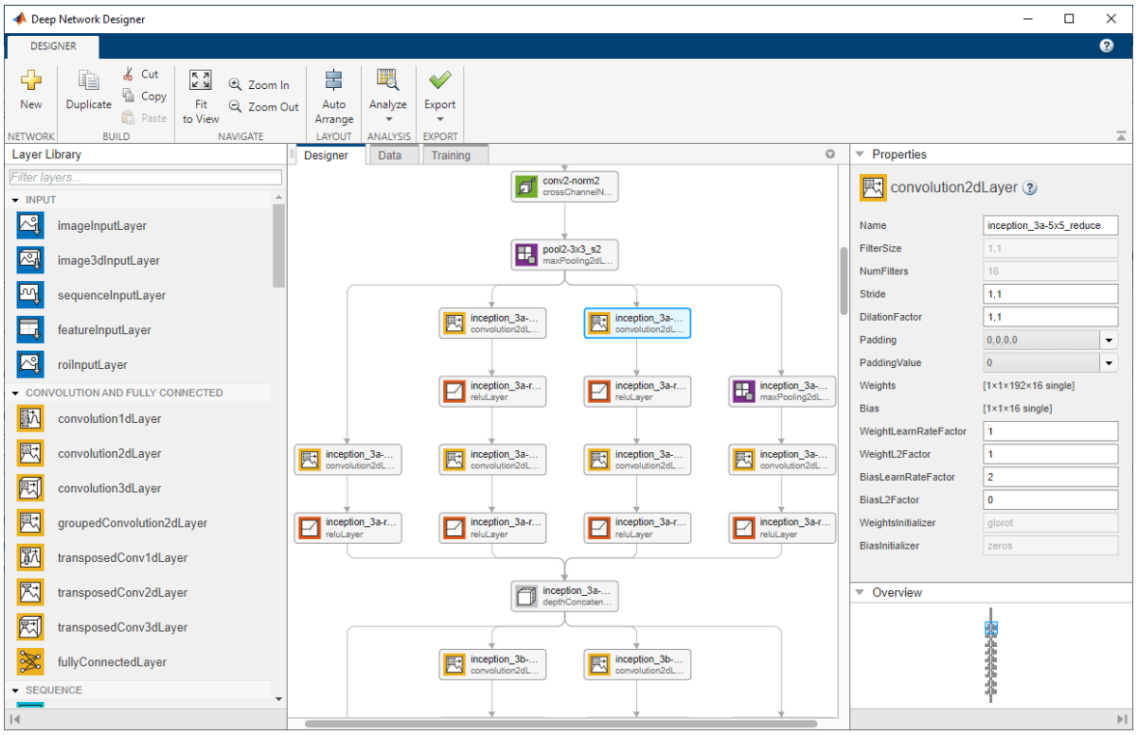


Figura 4-11. Interfaz gráfica *Deep Network Designer* [49]

5. Estructura de la solución propuesta

En este capítulo se detalla el proceso y los diferentes aspectos que han intervenido en el desarrollo de la solución propuesta para el problema tratado en este trabajo. Es por tanto que se recogen variadas tareas necesarias para poder desarrollar una solución de *DL*, tales como el preprocesado adecuado de los datos, el diseño de la estructura de la red, la optimización de hiperparámetros, así como la evaluación de la red neuronal desarrollada.

5.1. Análisis de la base de datos

Antes de zambullirnos en el desarrollo del algoritmo, una parte esencial es conocer y entender de la mejor forma posible la base de datos con la que vamos a trabajar. Esto puede englobar diversas tareas, como son ver el comportamiento de los sensores en cada caso, un análisis estadístico de los datos o cualquier tipo de actividad que consideremos que nos puede dar una información relevante sobre el problema al que nos estamos enfrentando. Este análisis se realizará por su parte para cada uno de los 4 subconjuntos de datos disponibles, comentados en la Sección 4.1.2.

5.1.1. FD001

Esta base de datos consta de información sobre un conjunto de 100 motores. Es la base de datos en teoría más sencilla, presentando un solo modo de fallo y un modo de operación. En el estado del arte es con diferencia la base de datos más estudiada, limitándose exclusivamente a ella numerosas publicaciones.

El tipo de señal que presentan los sensores en esta base de datos se presenta la Figura 5-1.

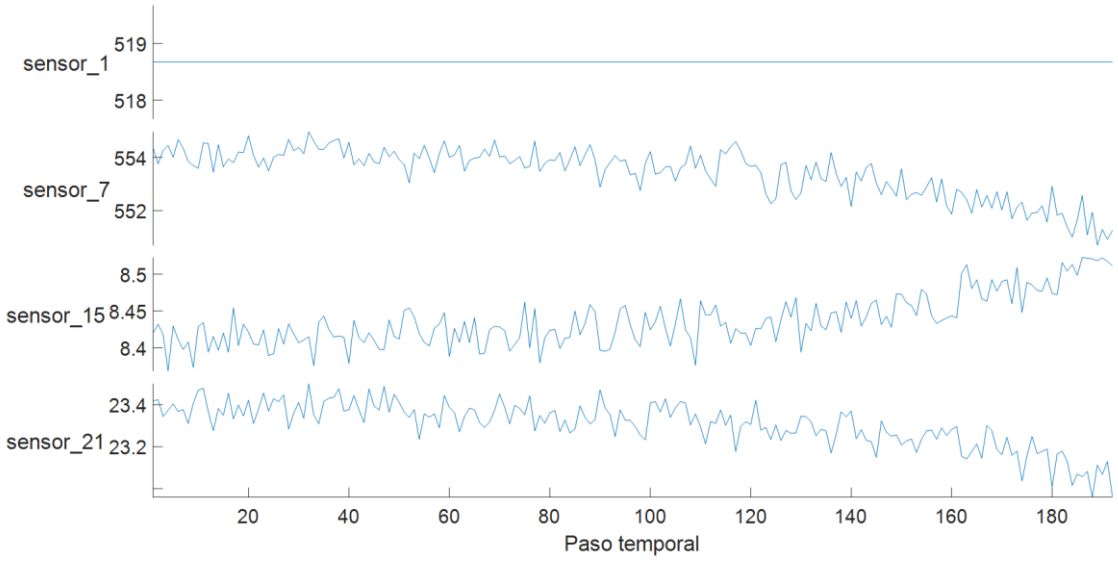


Figura 5-1. Señales sensores FD001

En la Tabla 5-1 se presenta un resumen de las longitudes de las secuencias de operación de los motores detallando la parte de entrenamiento, test y la longitud de las predicciones que se busca obtener. Por un lado, las longitudes de entrenamiento y test hacen referencia al número de pasos temporales desde el inicio de operación del motor hasta su fallo. En el caso de la longitud de las predicciones, nos referimos al número de pasos temporales que les quedan a los motores para fallar en el punto que se corta la secuencia de test, y cuyo valor será tarea de la red predecir.

	Media	**Varianza**	**Mínimo**	**25 %**	**50 %**	**75 %**	**Máximo**
Entrenamiento	206.31	46.34	128	174	199	229	362
Test	205.48	44.0419	140	173	198	226	340
Predicciones	75.52	41.76	7	29	85	112	145

Tabla 5-1. Resumen estadístico FD001

5.1.2. FD002

Esta base de datos consta de información sobre un conjunto de 260 motores. Sigue presentando un solo modo de fallo, pero 6 modos de operación. En el estado del arte esta base de datos no es frecuentemente estudiada y suele producir malos resultados.

El tipo de señal que presentan los sensores se presenta la Figura 5-2

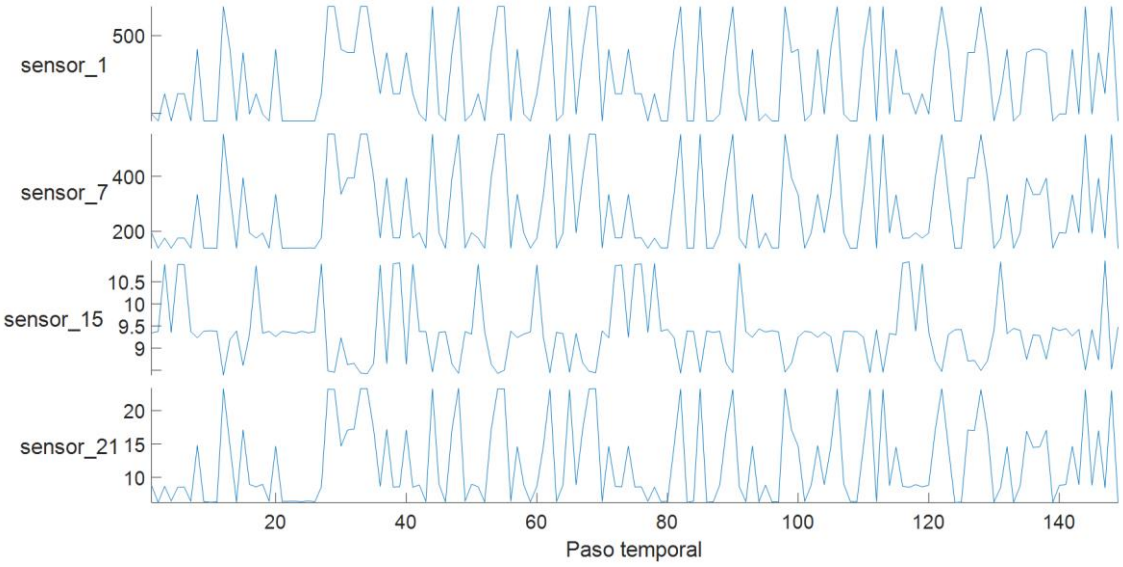


Figura 5-2. Señal sensores FD002

En la Tabla 5-2 se presenta un resumen estadístico de la longitud de las secuencias de operación de los motores.

	Media	Varianza	Mínimo	25 %	50 %	75 %	Máximo
Entrenamiento	206.7654	46.7822	128	174	199	230	378
Test	211.42	47.81	125	173	203	243	377
Predicciones	81.18	53.88	6	35	80	121	194

Tabla 5-2. Resumen estadístico FD002

5.1.3. FD003

Esta base de datos consta de información sobre un conjunto de 100 motores. Presenta 2 modos de fallo y solo 1 modo de operación. En el estado del arte esta base de datos es menos frecuente que la FD001 y suele producir resultados parecidos.

El tipo de señal que presentan los sensores en esta base de datos se presenta la Figura 5-3

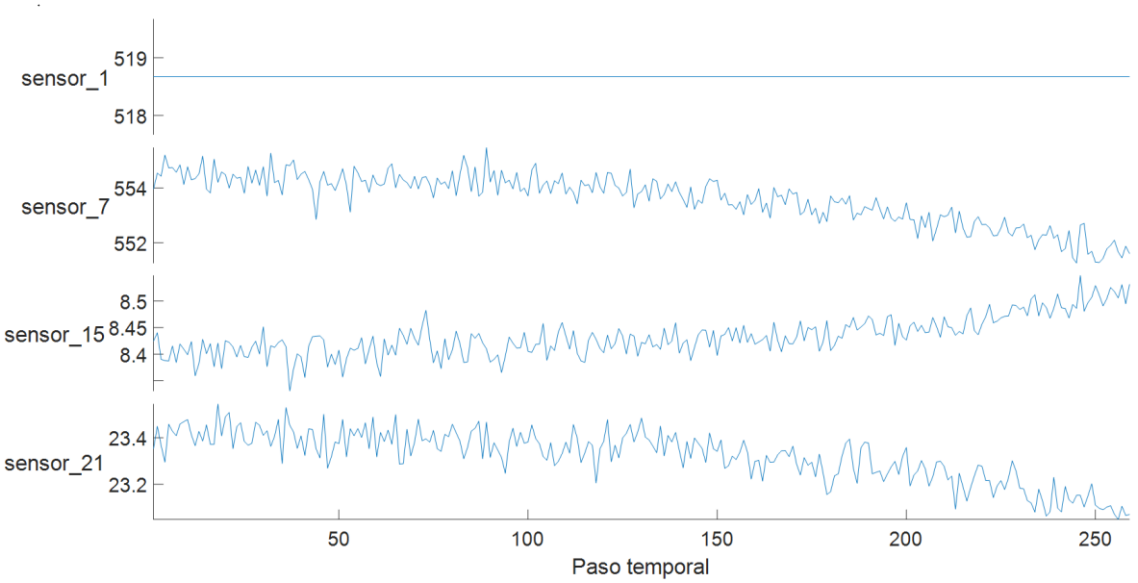


Figura 5-3. Señal sensores FD003

En la Tabla 5-3 se presenta un resumen estadístico de la longitud de las secuencias de operación de los motores.

	Media	Varianza	Mínimo	25 %	50 %	75 %	Máximo
Entrenamiento	247.2	86.48	145	189	220	278	525
Test	240.28	76.02	136	191	219	276	483
Predicciones	75.32	41.60	6	41	77	115	145

Tabla 5-3. Resumen estadístico FD003

5.1.4. FD004

Esta base de datos consta de información sobre un conjunto de 249 motores. Presenta 2 modos de fallo y 6 modo de operación. En el estado del arte esta base de datos no es frecuentemente estudiada y suele producir malos resultados.

El tipo de señal que presentan los sensores en esta base de datos se presenta la Figura 5-4.

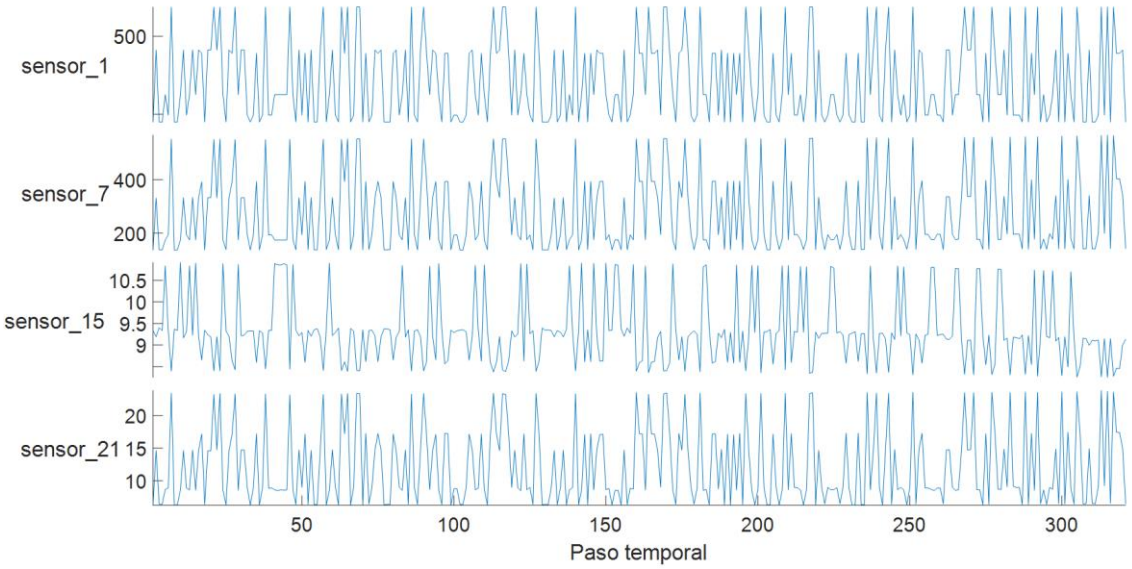


Figura 5-4. Señal sensores FD004

En la Tabla 5-4 se presenta un resumen estadístico de la longitud de las secuencias de operación de los motores.

	Media	Varianza	Mínimo	25 %	50 %	75 %	Máximo
Entrenamiento	245.97	73.11	128	187	234	290	543
Test	251.73	85.09	125	190	234	294	553
Predicciones	86.55	54.63	6	36	88	126	195

Tabla 5-4. Resumen estadístico FD004

5.1.5. Conclusiones

De los resultados obtenidos en el análisis preliminar de la base de datos podemos apreciar que estas presentan una clara similitud entre las impares por un lado y las pares por el otro. Además, se puede concluir de manera lógica que estas diferencias son principalmente debidas al número de condiciones de operación de cada base de datos.

Otra característica palpable es que las señales de los sensores en los subconjuntos impares presentan ostensiblemente menos ruido que en los pares, pudiéndose apreciar en estos claras tendencias a lo largo de la evolución del motor. Esto probablemente sea la razón por la cual los subconjuntos impares han dado generalmente menos problemas para ser analizados y por qué su uso es bastante más común en el estado del arte.

Para terminar, también se ha comprobado que la longitud de las secuencias es más o menos constante para todas las bases de datos tanto para entrenamiento y testeo, así como para el tamaño de las predicciones que la red se encargará de hacer.

5.2. Preprocesamiento de los datos

El preprocesamiento de los datos es un apartado crítico en todo proyecto que conlleve la utilización de técnicas que realicen un análisis de estos. La preparación de los datos es un proceso clave para conseguir que nuestro modelo funcione adecuadamente y, habitualmente, es una de las partes donde más tiempo se invierte.

Podemos diferenciar entre tres técnicas básicas:

- ***Data Cleaning*:** Hace referencia al proceso en el que rectificamos nuestra base de datos para completar valores vacíos, suavizar datos ruidosos, etc. La base de datos utilizada en el presente trabajo es provista por una institución que ya ha procurado que esta no presente este tipo de fallos, por lo que no será necesario realizar cambios relativos a este apartado.
- ***Data Transformation*:** Este apartado concierne al proceso en el que modificamos nuestra base de datos para que se encuentre en un formato idóneo con el que el algoritmo aprenda de ella. Generalmente se trata de una normalización de los datos y las posibilidades para realizarla son variadas.
- ***Data Reduction*:** Consiste en analizar el vector de características que se suministra al algoritmo, realizando una selección de cuales de ellas pueden ser eliminadas de cara a reducir el número de variables del problema.

5.2.1. Selección de variables

En este apartado se aborda la posibilidad de hacer una selección de las variables dentro de las bases de datos de secuencias de los motores. Esto básicamente responde a establecer una serie de criterios a través de los cuales evaluar las señales de los sensores y condiciones de operación, considerando en base a ello la utilidad de estas para el aprendizaje de la red y decidir si eliminarlas o no. De todas formas, tal y como se ha

visto en las publicaciones presentadas en las Sección 2.4, esta selección es un apartado controvertido ya que cada publicación parece seguir diferentes criterios o no realizar una selección en absoluto, obteniendo todas ellas buenos resultados.

En nuestro caso, hemos identificado que en la base de datos existen variables que producen una señal constante a lo largo de toda la secuencia. Se ha considerado que esta información no solo no aporta nada al proceso de entrenamiento, sino que, según algunas publicaciones, puede tener un impacto negativo en este.

Es por esto que se han buscado las señales que presentan un mínimo y un máximo iguales y se ha procedido a eliminar estas variables de la base de datos. El número de características restantes para cada subconjunto se recoge en la Tabla 5-5**.**

	FD001	**FD002**	**FD003**	**FD004**
N.º variables utilizadas	17	24	18	24

Tabla 5-5. Número de variables utilizadas

5.2.2. Normalización

La normalización de la base de datos en muchos casos ayuda a estabilizar el proceso de entrenamiento de la red, mejora la convergencia del método y reduce el riesgo de quedarse atascado en un óptimo local.

Dado que la base de datos presenta secuencias temporales multivariable en la que cada uno de los sensores presenta salidas en una escala diferente, resulta necesario escalar todos los datos a un rango común. En la Figura 5-5 pueden verse las diferencias en las señales de diferentes sensores para la base de datos FD001.

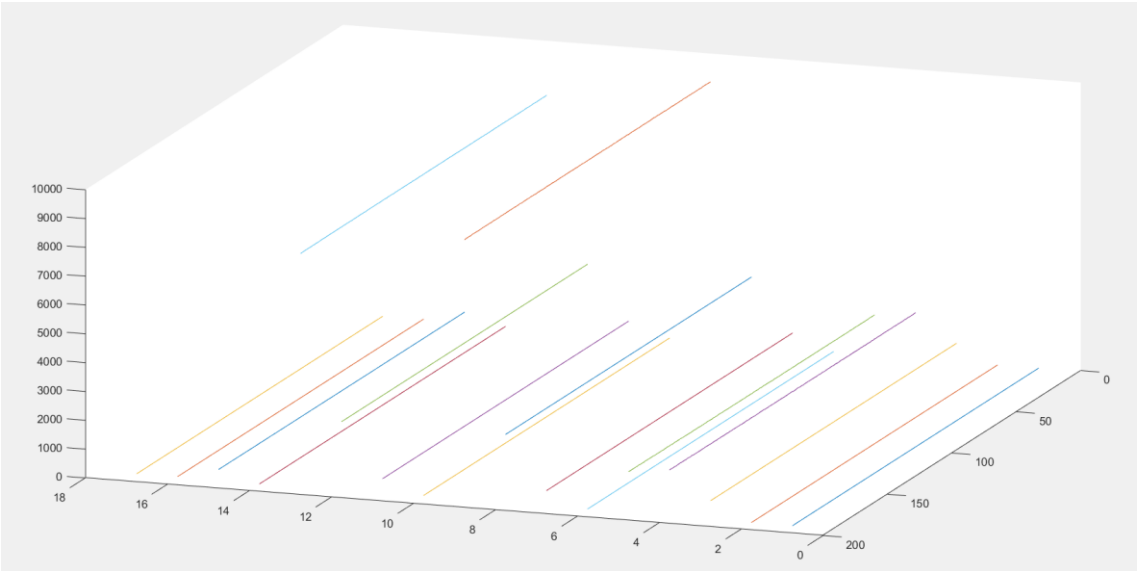


Figura 5-5. Diferente escala de los sensores para FD001

En particular en este trabajo se han considerado 2 tipos de normalizaciones:

- **Rango entre 0 y 1**

Este tipo de normalización consiste en escalar todos los datos para que estos se encuentren en un rango comprendido entre 0 y 1. Esto se consigue mediante la Ecuación (5-1).

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{5-1}$$

Las secuencias de datos resultantes presentan la forma recogida en la Figura 5-6 tras la transformación.

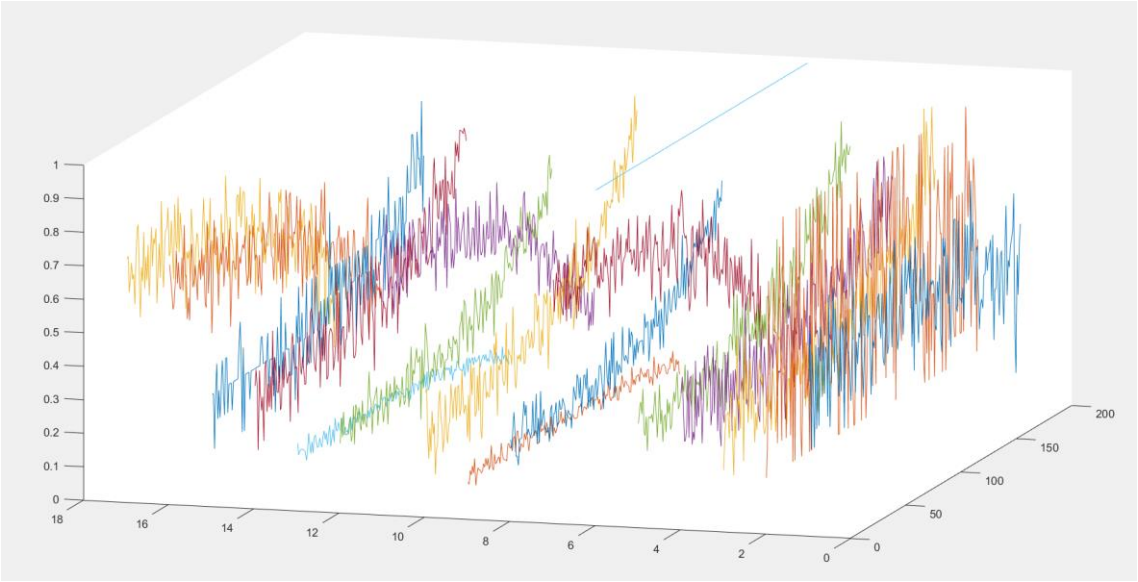


Figura 5-6. Sensores FD001 tras normalización entre 0 y 1

- **Media 0 y varianza 1**

Este tipo de normalización consiste en transformar todos los datos para que presenten media (μ) 0 y varianza (σ) unidad. Para esto primero debemos obtener la media y la varianza de la base de datos mediante la Ecuación (5-2) y Ecuación (5-3) respectivamente.

$$\mu = \frac{1}{n}\sum_{i=0}^{n} x \tag{5-2}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x-\mu)^2}{n}} \tag{5-3}$$

Siendo n el número de observaciones.

Con esto ya se puede realizar la normalización mediante la Ecuación (5-4).

$$z = \frac{x-\mu}{\sigma} \tag{5-4}$$

Las secuencias de datos resultantes presentan la forma recogida en la Figura 5-7 tras la transformación.

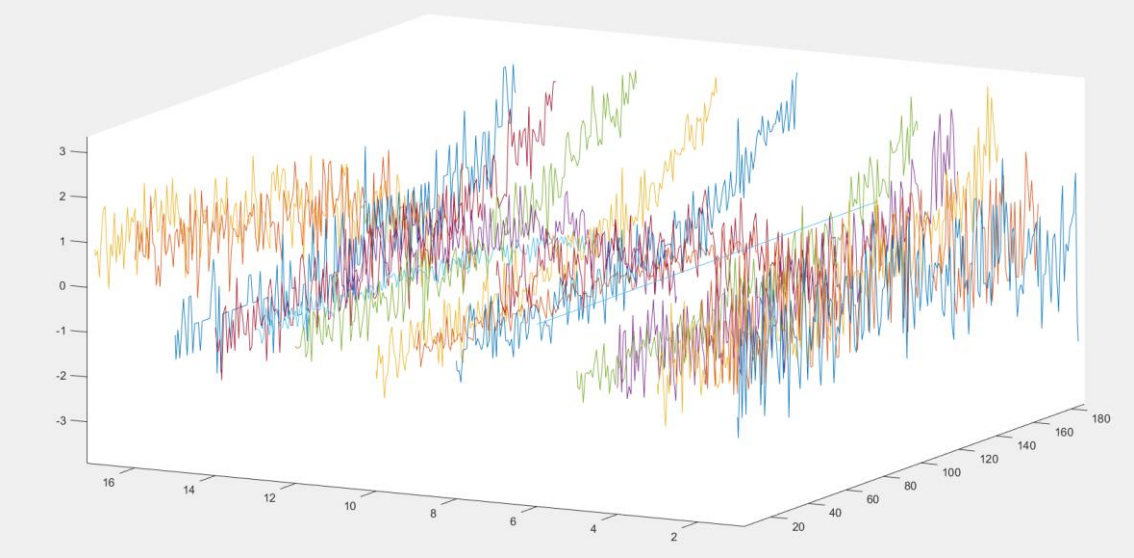


Figura 5-7. Sensores FD001 tras normalización media 0 y varianza 1

Dado que la red va a ser entrenada con los datos normalizados con una de las dos maneras, es necesario que a la hora estudiar el comportamiento de la red en la base de datos de testeo esta también este normalizada de la misma manera. Esto es debido a que la red espera los datos en el mismo formato con el que fue entrenada, no pudiendo funcionar adecuadamente si no se procede de esta manera.

5.2.3. Modelización de la degradación del motor

Para poder entrenar la red es necesario que esta tenga un valor con el que comparar su predicción a cada iteración de cara a optimizar sus parámetros. Dado que conocemos la longitud de las secuencias de cada uno de los motores, es posible crear una base de datos paralela en la que se recoja el número de ciclos restantes del motor en cada paso temporal para cada secuencia. Además, con esta información se puede manipular el

target de la red de diferentes maneras para tratar de optimizar la capacidad de aprendizaje del algoritmo.

- **RUL a trozos**

La base de datos de estudio consiste en secuencias de datos producidas por varios sensores que proporcionan información sobre la operación del motor. Estas secuencias van variando a lo largo de la operación del motor, degradándose este a lo largo de los ciclos hasta fallar.

En estas, los motores no presentan problemas desde un inicio, sino que pasan por diferentes fases a lo largo de su operación. En la Figura 5-8 vemos en las señales de uno de los sensores la evolución típica de un motor:

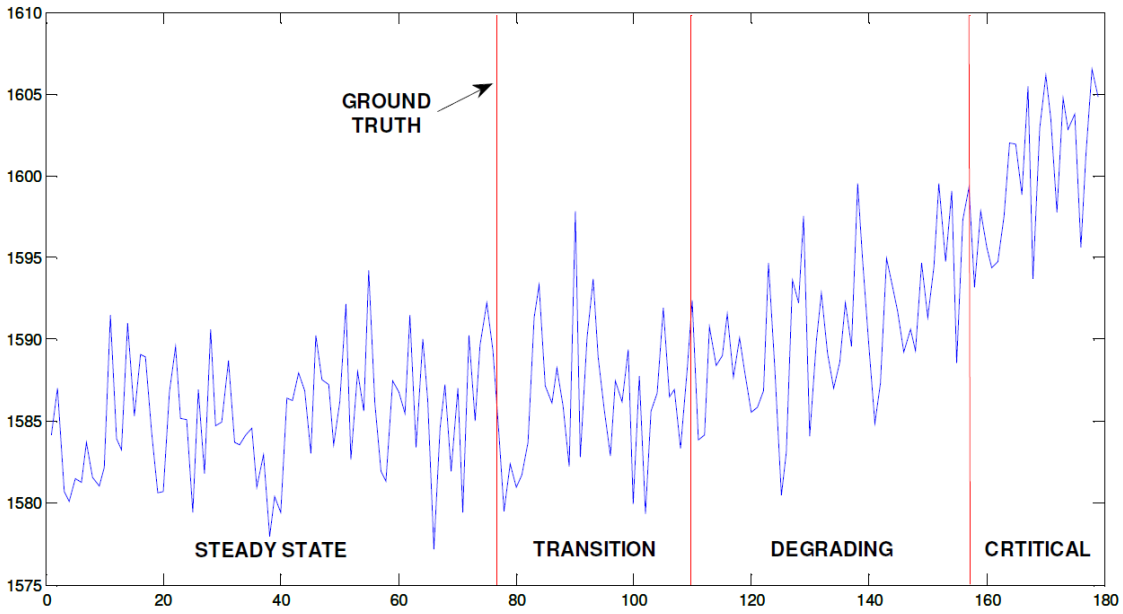


Figura 5-8. Diferentes fases de operación de un motor [50]

Tal y como se puede apreciar en la Figura 5-8 (ignorando el ruido que presentan las señales) las señales del motor presentan distintas tendencias a lo largo de la vida del motor.

- **Primera fase (estable):** al principio el motor no presenta problemas y la señal del sensor se encuentra estable. En esta fase, intentar realizar predicciones sobre la vida del motor no será muy concluyente, ya que este no presenta señales perceptibles de fallo.
- **Segunda fase (transición):** la pendiente de las señales empieza a cambiar levemente por lo que se puede intuir el inicio de un fallo en el motor. En esta fase las predicciones no serán las más precisas, pero ya se pueden empezar a apreciar variaciones.

- **Tercera fase (degradación):** en esta fase el problema en el motor es claro y este avanza rápidamente.
- **Cuarta fase (fallo):** el motor se encuentra en el momento de fallo en este momento.

Dado que podemos considerar entonces que la degradación del motor al inicio de su uso será negligible y esta se irá volviendo más evidente según avancemos en la secuencia, en la bibliografía es común ver como se adopta una estrategia en la que se modela la RUL del motor a trozos. En este método, establecemos un límite máximo por encima del cual la RUL será un valor constante y después de este se presentará la degradación lineal esperada. En la bibliografía este valor de umbral se establece generalmente entre 125 y 130, aunque el modo de llegar a este valor parece ser principalmente experimental ya que apenas aparecen razonamientos que lo sostengan. En la Figura 5-9 podemos ver una representación gráfica de este método.

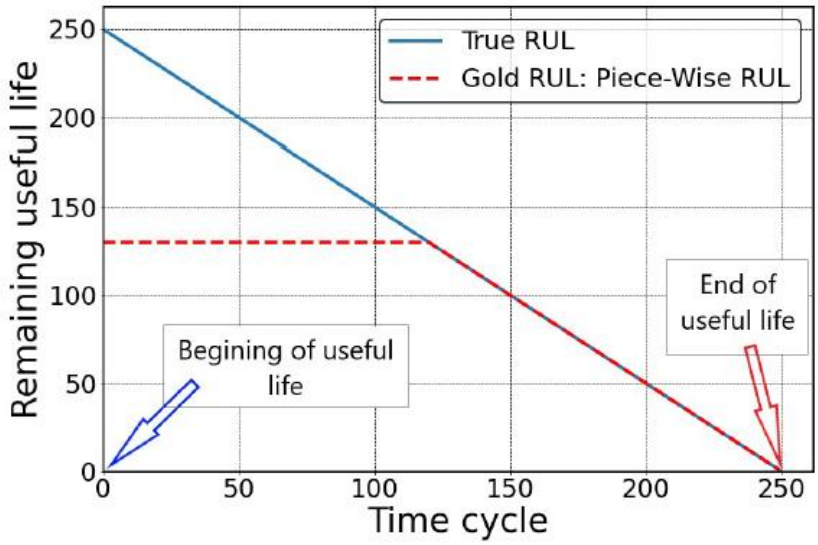


Figura 5-9. Modelado de la RUL a trozos [14]

La imposición de este valor de umbral viene a facilitar el proceso de entrenamiento de la red, dado que reduce el rango de valores a predecir, aquellos donde sería difícil dar resultados precisos. Por contraparte también provocará que la red sea incapaz de realizar predicciones por encima de este valor, aunque este caso es poco común en la base de datos de testeo.

- **<u>RUL a trozos con umbral variable</u>**

El método anterior, tal y como se aplica en el estado del arte, presenta algunos problemas a nuestro juicio.

  - Este valor de umbral se establece sin demasiado razonamiento previo. Muchas veces simplemente los autores se escudan en que es lo que otras

publicaciones hacen y otros comentan que experimentalmente han comprobado que es lo que mejores resultados les da.

- Como vimos en la Sección 5.1, la longitud de las secuencias de la base de datos es ciertamente variable. Se puede intuir de aquí que la tesis de establecer un umbral constante para todas ellas probablemente no se adapte correctamente a las características de cada una.
- El valor utilizado sirve exclusivamente para esta base de datos y la aplicación de este método en una base de datos de otras características presumiblemente necesitaría de otro proceso "experimental" para encontrar un valor idóneo.

Como se ejemplificó en la Figura 5-8, se puede suponer que las secuencias de los motores atraviesan fases diferenciadas, caracterizadas por cambios en las señales de los sensores. No es raro suponer entonces que un proceso de *clustering* podría ayudarnos a identificar estas fases.

Existen diversos métodos posibles para realizar una tarea de *clustering*. De todas maneras, en nuestro caso nos hemos decantado por el método *k-means* por ser el que mejores resultados nos ha ofrecido. La explicación del método se puede encontrar en la Sección 3.5.

Del estado del arte partimos de la idea de que 4 es un número idóneo, aunque una buena forma de elegir el número de *clusters* puede ser viendo como disminuye la suma de distancias entre datos y centroides, según aumentamos el número de *clusters.* Esto lo que nos indica es cuán significativo es el aumento de *clusters* respecto al error. En la Figura 5-10 vemos como varía el valor de esta suma.

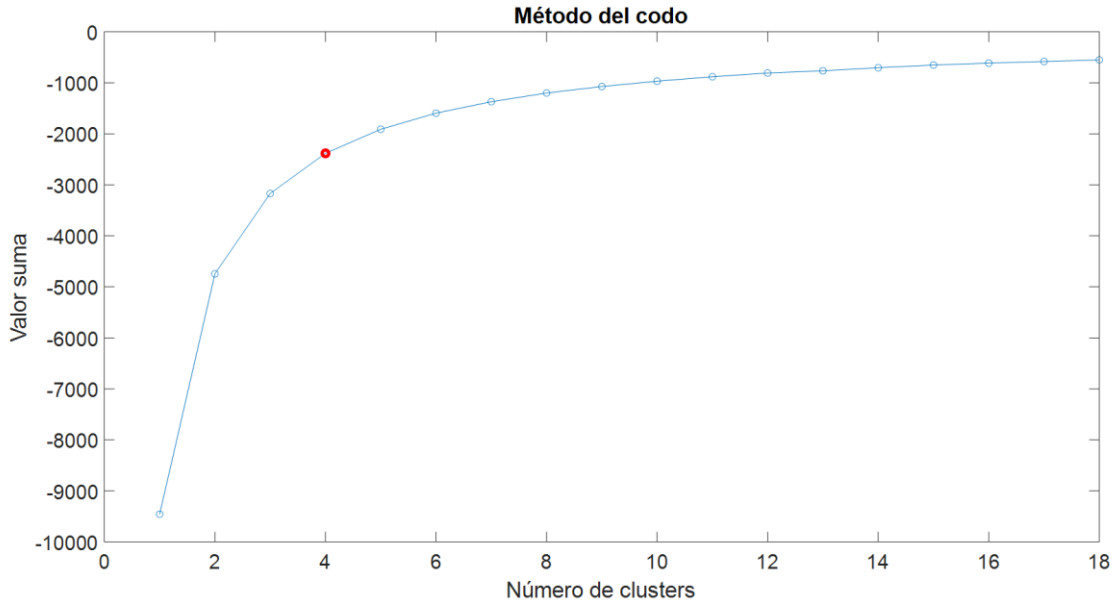


Figura 5-10. Método del codo *k-means*

Se puede apreciar que al principio cambia muy rápidamente y luego la bajada se estabiliza. Por lo general se establece que un buen número se encuentra en lo que se conoce como "codo" que es donde cambia la tendencia. En este caso, se considera en torno a 4 *clusters*.

Podemos ver los efectos sobre poner un umbral constante en la Figura 5-11, donde este afecta a cada motor en una fase diferente de su funcionamiento. Los motores se encuentran ordenados según la longitud de sus secuencias de mayor a menor. La línea vertical indica el umbral de 125.

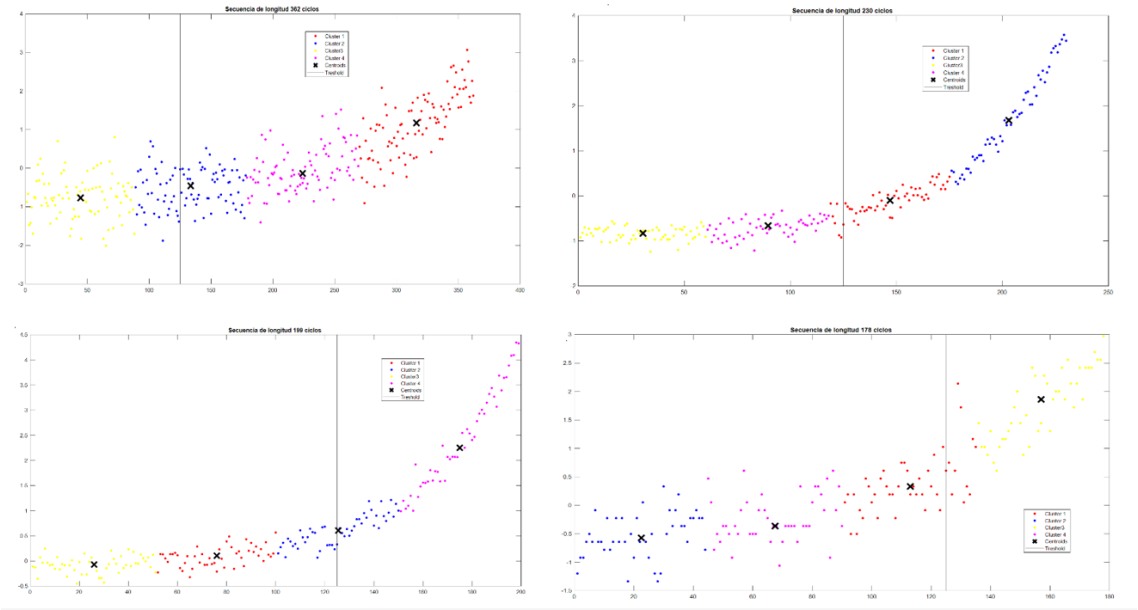


Figura 5-11. Aplicación de umbral constante a diferentes motores

Para intentar suplir este problema se ha establecido un método de umbral variable.

La idea básicamente consiste en aplicar un umbral variable, seleccionando para cada motor el punto donde empieza el tercer *cluster* ya que se considera a partir del cual los datos son representativos. En la Figura 5-12 se presenta cómo se establece el límite máximo.

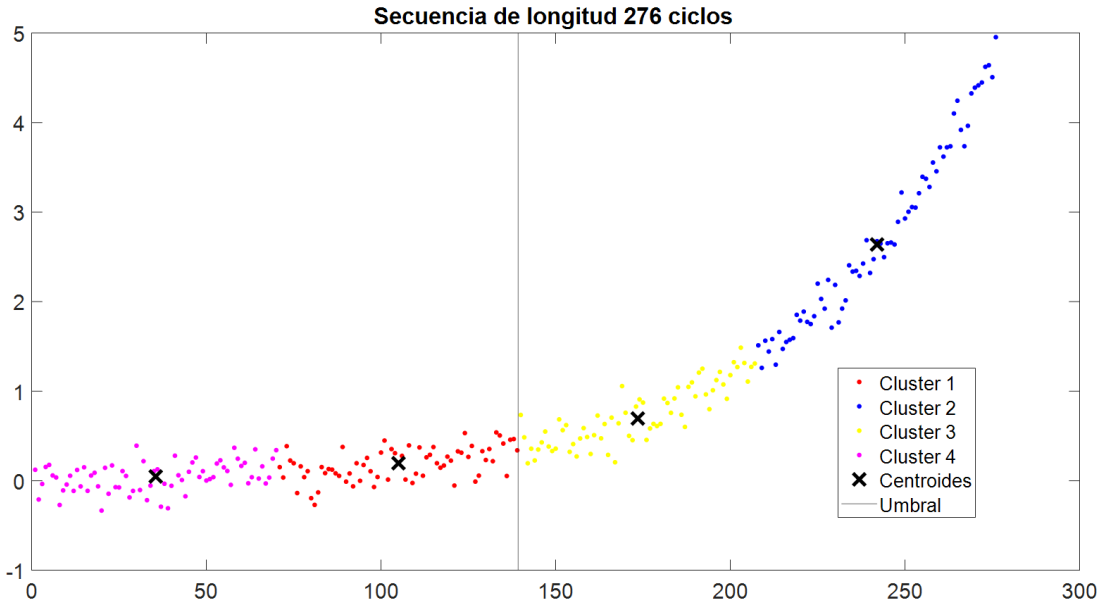


Figura 5-12. Aplicación de umbral variable

- **<u>Índice de salud exponencial</u>**

Como método alternativo tenemos la opción de cambiar la RUL por un índice de salud del motor. Esta transformación responde a la Ecuación (5-5) que establece una progresión exponencial. Esta forma de modelizar la evolución del motor se utiliza también en la publicación [19]. Cabe decir que esta función coincide también con la utilizada por los creadores de la base de datos para modelizar diferentes parámetros de degradación del motor, como se recoge en [15].

$$h(t) = 1 + d - exp\{a * t^b\} \tag{5-5}$$

Cuadrándose "a" para que cuando la secuencia de cada motor llegue a su fin el h(t) = 0.

$$a = \frac{\log(1+d)}{t} \tag{5-6}$$

En la Figura 5-13 se puede ver la comparación entre la modelización de la RUL a trozos de apartados anteriores con el índice de salud.

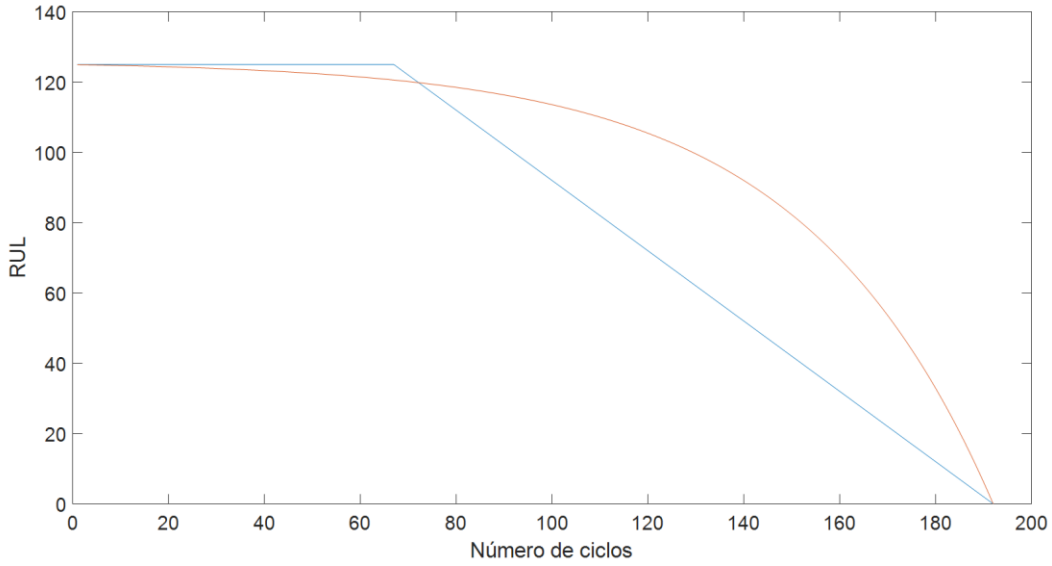


Figura 5-13. Comparación entre la modelización por RUL a trozos e índice de salud exponencial

5.2.4. División de la base de datos en entrenamiento y validación

La finalidad principal del desarrollo de un algoritmo de *DL* para estimar la vida de motores Turbofán es la de obtener predicciones fiables en datos nuevos, es decir, que nunca hayan sido vistos antes por el algoritmo. El error mostrado por el proceso de entrenamiento suele ser bastante optimista, por lo que es necesario utilizar la red sobre la base de datos de test o utilizar técnicas de validación.

Estos métodos de validación nos permiten obtener una estimación más fiable acerca de la precisión en las predicciones que nos puede ofrecer nuestra red únicamente con la base de datos de entrenamiento. Para esto, la base de datos de entrenamiento se divide en dos subconjuntos, utilizándose uno para el entrenamiento de la red y el resto de las observaciones se utilizan para evaluar su funcionamiento. Este método se basa en que los datos de validación no han sido "vistos" por la red, esta no ha aprendido a partir de ellos. La utilización de los datos de validación será un aspecto clave a la hora de optimizar los hiperparámetros de la red.

También resulta un parámetro a valorar propiamente durante el entrenamiento, ya que nos da información sobre si la red está cayendo en el *over-fitting* comentado en la Sección 3.2.3.

La división en datos de entrenamiento y validación se ha realizado eligiendo un porcentaje de secuencias de la base de datos de entrenamiento que van a formar parte del subconjunto de validación, siendo estas seleccionadas en un proceso aleatorio a través de este porcentaje.

Este método tiene su contraparte, ya que limitamos la cantidad de datos que puede usar la red en su aprendizaje. Dado que las 4 bases de datos disponibles no son del mismo

tamaño, se ha observado que utilizar el mismo porcentaje para todas afecta negativamente en el resultado final de la red. En la Tabla 5-6 se recogen los porcentajes utilizados finalmente en cada base de datos.

	FD001	**FD002**	**FD003**	**FD004**
% Entrenamiento	90	80	90	80
% Validación	10	20	10	20

Tabla 5-6. Porcentajes de entrenamiento y validación

5.3. Estructuras de red implementadas

En este apartado se recogen las diferentes estructuras de redes neuronales que se han evaluado a lo largo del estudio. Se especificará su composición y el razonamiento o referencias que se ha seguido para decidir construirlas de esta manera.

5.3.1. MLP

Por ser las MLP el tipo de red neuronal más básica, se ha considerado interesante iniciar el análisis utilizando este tipo de redes. Tal y como vimos en la Sección 2.4, del estudio de la bibliografía se podía extraer que este tipo de redes por su cuenta no son capaces de obtener buenos resultados ya que no son capaces de encontrar relaciones en secuencias de datos a diferencia de otras estructuras como se comentó en la Sección 3.1. De todas formas, se ha decidido comprobar estos resultados de la MLP como punto de partida y a partir de ahí tratar de mejorarlos implementando diferentes posibilidades.

La estructura que se ha elegido para la red MLP es la que aparece en la Figura 5-14. En la Tabla 5-7 se recogen los hiperparámetros elegidos para la red, siguiendo el proceso comentado en la Sección 5.4 tratando de optimizar en lo posible los resultados.

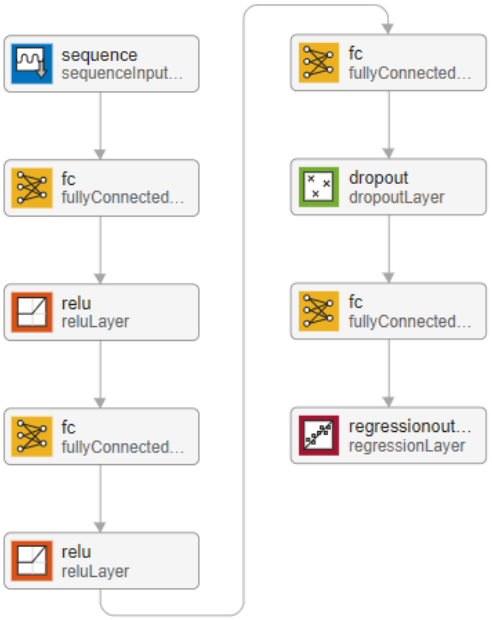


Figura 5-14. Estructura red MLP

Hiperparámetro	Valor
Learning rate	0.051
Número de capas	4
Número de neuronas por capa	71/57/57/1
Función de activación	ReLu
Batch size	116

Tabla 5-7. Hiperparámetros red MLP

5.3.2. CNN

Las redes neuronales convolucionales son una de las arquitecturas más preponderantes actualmente. Aunque su campo de aplicación más conocido son los relativos al tratado de imágenes, en el estudio del estado del arte pudimos comprobar que también gozan de

cierto protagonismo a la hora de estudiar secuencias, proponiendo una alternativa a la opción más común de las redes RNN. Esto se consigue mediante un proceso similar al comentado en la Sección 3.1.4, pero realizando el proceso de convolución de forma 1D, aplicándolo a cada serie temporal. Información más detallada sobre este método se puede encontrar en [51].

En la Figura 5-15 se representa la arquitectura de red utilizada. Para la construcción de esta nos hemos basado en [11]. En la Tabla 5-8 se recogen los hiperparámetros que se han seleccionado para la red.

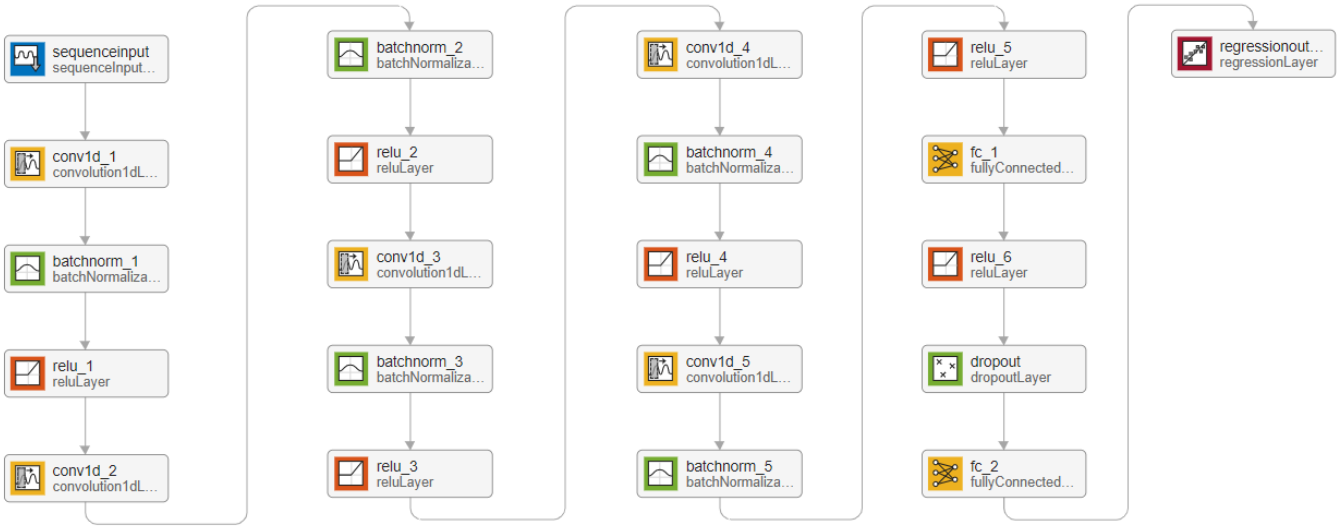


Figura 5-15. Estructura red CNN

Hiperparámetro	Valor
Learning rate	0.0043
Número de capas	7
Tamaño de filtros	5/7/11/13/15
Número de filtros	32/64/128/256/512
Número de neuronas capas MLP	100/1
Función de activación	ReLu
Batch size	5

Tabla 5-8. Hiperparámetros red CNN

5.3.3. LSTM básica

Las redes LSTM se empezaron a estudiar partiendo la arquitectura típica utilizada en este tipo de redes. Esta estructura se caracteriza por presentar una o varias redes LSTM seguidas para procesar las secuencias temporales, acompañadas de una red MLP para obtener la predicción de la RUL. Alguna publicación que presenta este tipo de estructura es por ejemplo [13].

En la Figura 5-16 se representa la arquitectura de red utilizada. En la Tabla 5-9 se recogen los hiperparámetros que se han seleccionado para la red.

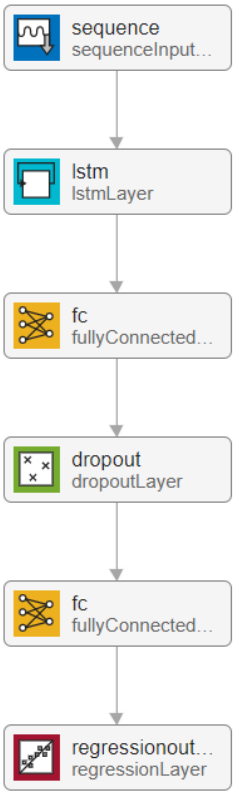


Figura 5-16. Estructura red LSTM

Hiperparámetro	**Valor**
Learning rate	0.0079
Número de capas	3
Número de neuronas por capa	200/50/1
Función de activación	Tanh
Batch size	32

Tabla 5-9. Hiperparámetros red LSTM

5.3.4. LSTM + MLP

Uno de los problemas que han resultado más llamativos en el estudio del problema ha sido la existencia de, por lo general, grandes diferencias de rendimiento en las propuestas del estado del arte a lo largo de los 4 subconjuntos de datos o la directa omisión en el estudio de los subgrupos más complicados. Es por esto que uno de los objetivos de este trabajo se estableció en proponer una estructura que pudiese ofrecer un rendimiento coherente en todas las partes del problema.

Las redes LSTM son una herramienta que ofrece grandes ventajas para el estudio de series temporales. Es por esto que, junto a la idea respecto a la utilización de redes MLP de [14] que ofrecía un funcionamiento más coherente sobre todos los subconjuntos, se ha desarrollado la arquitectura propuesta en la Figura 5-17. En la Tabla 5-10 se recogen los hiperparámetros que se han seleccionado para la red.

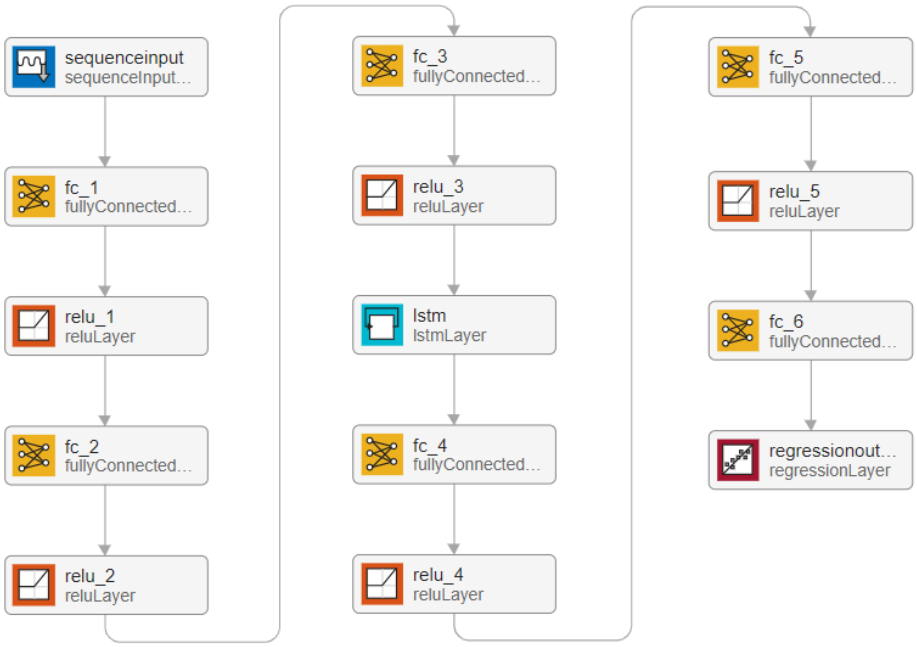


Figura 5-17. Estructura red LSTM con MLP

Hiperparámetro	Valor
Learning rate	0.0023
Número de capas	7
Número de neuronas por capa	100/50/50/60/60/30/1
Función de activación	ReLu
Batch size	44

Tabla 5-10. Hiperparámetros red LSTM con MLP

5.4. Optimización de hiperparámetros

Para obtener las estructuras finales presentadas en la Sección 5.3, una vez se ha decidido la idea de estructura de red que se quiere utilizar, hace falta buscar una combinación de hiperparámetros que proporcione los mejores resultados posibles en todas las bases de datos. Este trabajo es un proceso arduo y con mucha variabilidad ya que las posibilidades para combinar los parámetros de la red son prácticamente infinitas, sumado al hecho de que las 4 bases de datos presentan características diferentes entre sí. De todas maneras, se ha tendido a priorizar la optimización de los resultados en las bases de datos FD002 y FD004, que como vimos en la Sección 5.1 son las que presentan más problemas y dónde las propuestas recogidas en la bibliografía consiguen los resultados llamativamente más pobres o bien no las abordan.

Para realizar esta tarea se ha seguido un método mixto e iterativo donde se ha combinado la manipulación manual de los hiperparámetros y la optimización bayesiana recogida en la Sección 3.4, siendo para ello de gran ayuda la herramienta *Experiment Manager* de MATLAB, cuyas posibilidades vienen recogidas en la Sección 4.2.

Con esta metodología, se han tuneado los hiperparámetros guiándonos por los resultados sobre el subconjunto de validación, el cual nos da una idea de las capacidades predictivas reales de la red. Los parámetros que se han considerado para realizar la optimización son el *learning rate*, *batch size*, número de capas y neuronas en estas o funciones de activación utilizadas.

Se ha podido comprobar claramente que la elección de los hiperparámetros es un apartado determinante en el funcionamiento de la red y que determina en un grado muy elevado cual será el funcionamiento final de nuestro algoritmo.

5.5. Métricas de rendimiento

Para poder evaluar cómo se comporta la red neuronal hace falta obtener mediciones de su capacidad de predicción. Para esto básicamente se compara la predicción realizada por la red con el valor correcto. De todas maneras, existen diferentes métricas a la hora de realizar esta comparación, teniendo cada una de ellas características diferentes que hacen resaltar una información u otra del comportamiento de la red. Esta evaluación se lleva a cabo tanto en el proceso de entrenamiento de la red como posteriormente en el testeo de esta.

Las métricas existentes son muy diversas, pero en el presente trabajo se han utilizado las siguientes:

- **MSE (*Mean Squared Error*)**

Se trata de una métrica que calcula la media de las diferencias entre predicciones y soluciones al cuadrado. Matemáticamente responde a la Ecuación (5-7).

$$MSE = \frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N} \tag{5-7}$$

Esta métrica tiene la característica de que penaliza con más dureza a las diferencias entre valores elevadas que a las más pequeñas debido a que está elevada al cuadrado.

- **RMSE (*Root Mean Squared Error*)**

Esta métrica se puede ver como una extensión de la anterior dado que presenta la misma estructura salvo por que se realiza la raíz cuadrada de lo anterior. Esto lo que supone es que las unidades del valor RMSE son las mismas que las unidades originales de la variable predicha. Matemáticamente responde a la Ecuación (5-8).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}} \tag{5-8}$$

- ***Score Function***

Generalmente en el tipo de problemas que nos ocupa se valora como mejor una predicción adelantada que una tardía. Esto es debido a cuestiones de seguridad operativa, ya que una predicción tardía puede resultar catastrófica. Es por esto que se utiliza una función que simula este concepto, la función responde matemáticamente a la Ecuación (5-9).

$$Score = \begin{cases} e^{-\frac{d}{13}} - 1 & si \quad d < 0 \\ e^{\frac{d}{10}} - 1 & si \quad d \geq 0 \end{cases}$$

Dónde

$$d = \hat{y}_t - y_t$$

$$\hat{y}_t = RUL\ estimada$$

$$y_t = RUL\ real$$

(5-9)

La puntuación final será la suma de la puntuación de todas las predicciones.

Esta función viene representada gráficamente en la Figura 5-18.

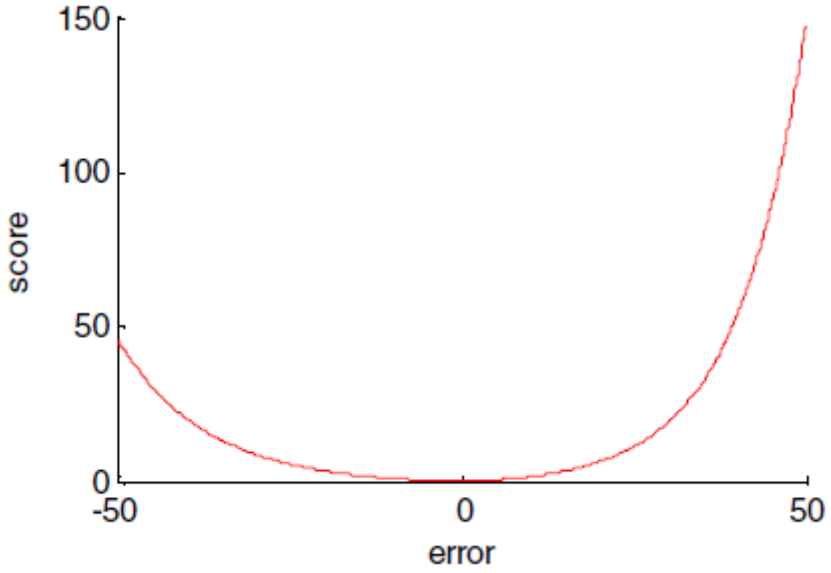


Figura 5-18. *Score function*

6. Resultados

En este apartado se recopilan los resultados obtenidos tras aplicar los conceptos y las estructuras propuestas en el Capítulo 5. La estructura a seguir será primero la de comprobar los resultados que nos aportan las diferentes arquitecturas de red neuronal propuestas utilizando el método de RUL a trozos comentado en la Sección 5.2.3. Esto se hará así por que es la metodología utilizada por la inmensa mayoría de las publicaciones estudiadas en el estado del arte, por lo que así podremos realizar una comparación más efectiva. Más adelante se estudiarán las otras posibilidades comentadas en la Sección 5.2.3 para conocer su aplicabilidad.

6.1. MLP

Utilizando la arquitectura especificada en la Sección 5.3.1 se han obtenido los resultados especificados en la Tabla 6-1 para todos los subconjuntos de datos.

	FD001	**FD002**	**FD003**	**FD004**
RMSE	20.65	21.84	18.6	30.63
Score	2128.7	9690	947.58	26621

Tabla 6-1. Resultados MLP

Como podemos ver en la Tabla 6-1 los resultados obtenidos no son muy precisos, siendo esto especialmente notable en el subconjunto FD004, que justamente coincide con ser la base de datos con más modos de fallo y condiciones de operación.

De manera más gráfica, podemos ver como en ejemplo en la Figura 6-1 una presentación de las predicciones de la base de datos FD002.

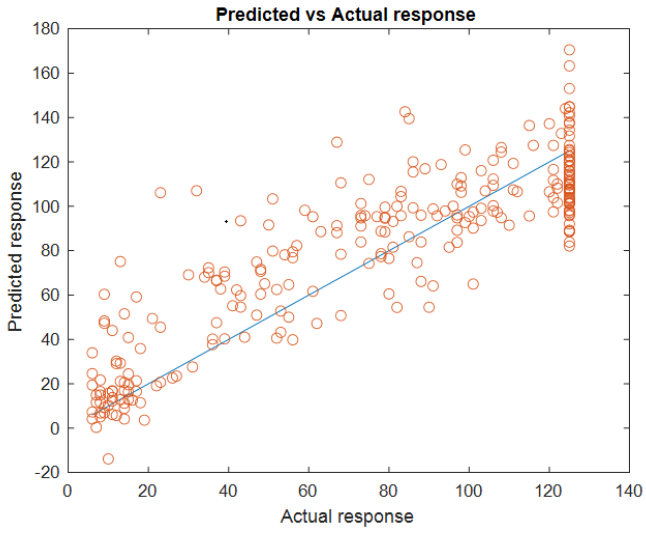


Figura 6-1. Representación gráfica predicciones MLP en FD002

En la Figura 6-1 cada punto presenta una predicción de la red. Si las predicciones fuesen perfectas estas deberían coincidir con la línea recta azul. Como se puede apreciar la red es capaz de ver la tendencia de los datos, pero goza de poca precisión.

En la Figura 6-2 se presentan las predicciones de la red a lo largo de distintas secuencias. Como podemos apreciar, las predicciones de la red a cada paso temporal presentan bastante ruido. En estas imágenes, la línea azul representa la RUL a trozos real, presentándose la parte constante seguida de la zona de degradación lineal como se comentó en la Sección 5.2.3.

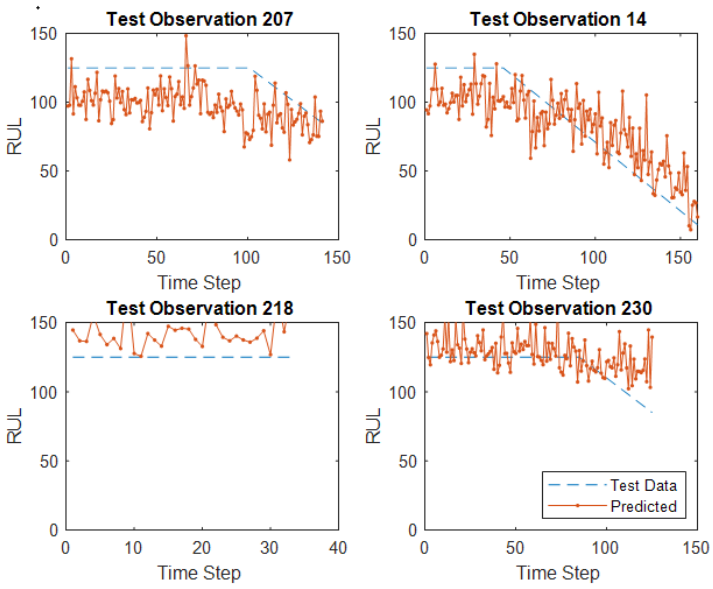


Figura 6-2. Predicción de secuencias por la red MLP para FD002

Por otra parte, un aspecto que se ha de destacar de este tipo de redes es el reducido coste computacional, siendo posible entrenar la red en breves períodos de tiempo. En la Tabla 6-2 se recogen los tiempos de entrenamiento utilizados para cada uno de los subconjuntos.

	FD001	**FD002**	**FD003**	**FD004**
Tiempo [min]	1: 51	3: 46	2: 10	2: 32

Tabla 6-2. Tiempo de entrenamiento red MLP

6.2. LSTM básica

Utilizando la arquitectura especificada en la Sección 5.3.3 se han obtenido los resultados especificados en la Tabla 6-3 para todos los subconjuntos de datos.

	FD001	FD002	FD003	FD004
RMSE	18.42	22.82	15.19	26.81
Score	769.80	2492.11	1230	3784

Tabla 6-3. Resultados LSTM

Con esta arquitectura los resultados son algo mejores en las bases de datos impares, aunque las predicciones en general distan de poseer la precisión que se buscaría en una herramienta sobre la que estructurar procedimientos empresariales.

A pesar de esto, una característica que sí que se aprecia en este tipo de redes es su capacidad de detectar secuencia. Esto se puede apreciar en la Figura 6-3.

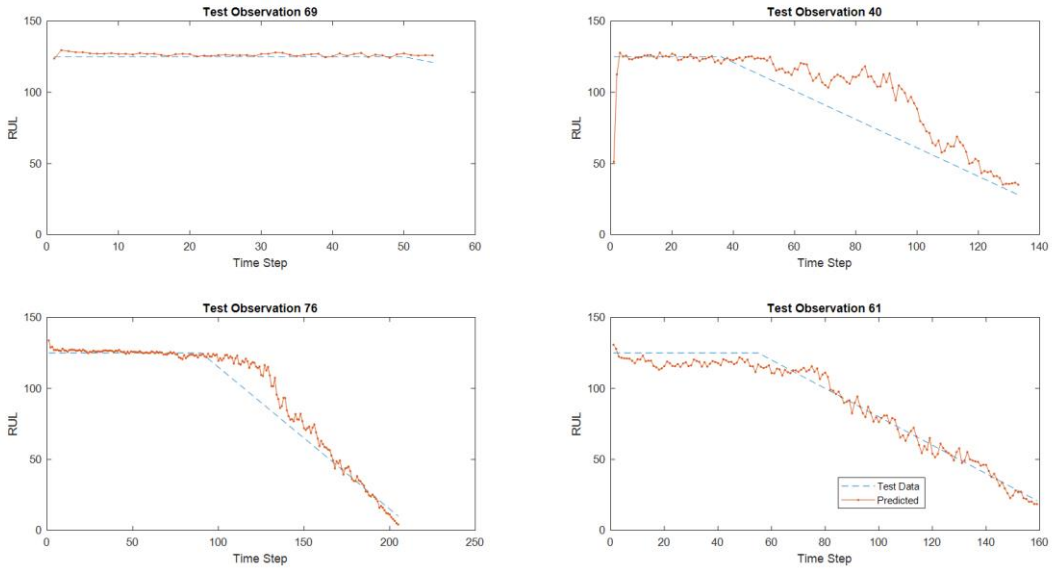


Figura 6-3. Predicción de secuencias por la red LSTM para FD001

En este caso se puede ver claramente como las predicciones de cada paso temporal presentan ostensiblemente menos ruido de unas a otras, siendo palpable que la red es capaz de captar las relaciones temporales de la evolución de cada secuencia.

Por su parte, utilizar este tipo de redes conlleva un mayor gasto computacional, tardando ostensiblemente más tiempo en ser entrenada. En la Tabla 6-4 se recogen los tiempos de entrenamiento utilizados para cada uno de los subconjuntos.

	FD001	FD002	FD003	FD004
Tiempo [min]	12: 45	27: 13	14: 20	35: 59

Tabla 6-4. Tiempo de entrenamiento red LSTM básica

6.3. CNN

Utilizando la arquitectura especificada en la Sección 5.3.2 se han obtenido los resultados especificados en la Tabla 6-5 para todos los subconjuntos de datos.

	FD001	FD002	FD003	FD004
RMSE	14.27	15.26	16.5336	19.18
Score	360.84	971.18	761.74	2312.20

Tabla 6-5. Resultados CNN

Con esta arquitectura obtenemos una notable mejoría en prácticamente todos los aspectos, siendo palpable la utilidad de las redes convolucionales también para el estudio de series temporales.

En la Figura 6-4 podemos apreciar como la red es capaz de seguir series temporales de manera coherente, aunque quizá presentando algo más de variabilidad en sus predicciones que en el caso de las LSTM.

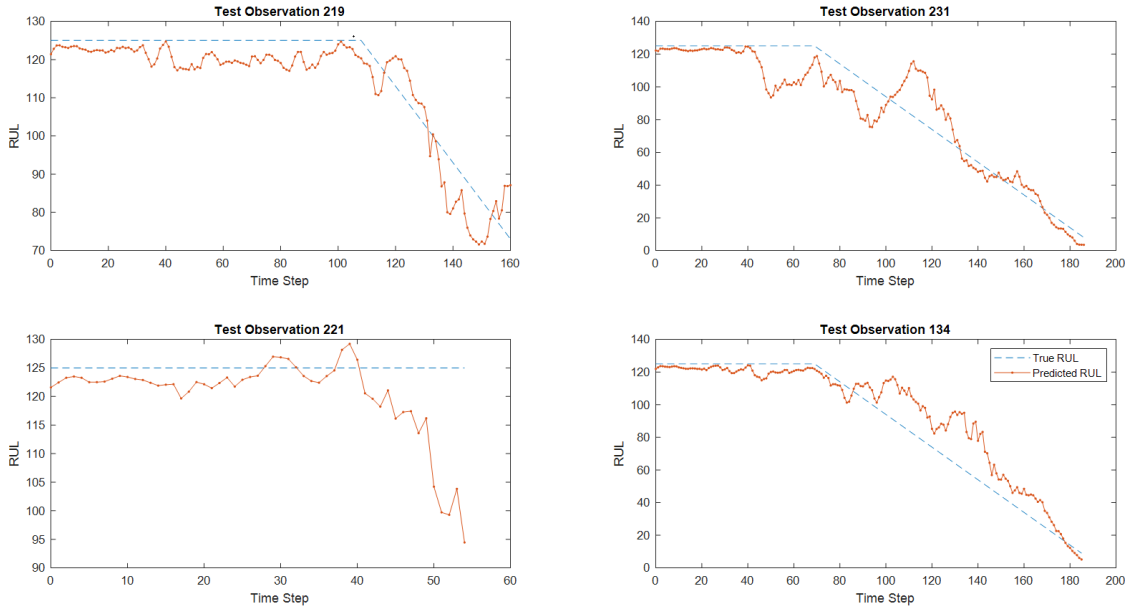


Figura 6-4. Predicción de secuencias por la red CNN para FD002

Cabe destacar también de esta arquitectura la rapidez del proceso de entrenamiento, el cual se pudo completar en ostensiblemente menos tiempo que para el caso de las LSTM. En la Tabla 6-6 se recogen los tiempos de entrenamiento utilizados para cada uno de los subconjuntos.

	FD001	FD002	FD003	FD004
Tiempo [min]	3: 09	9: 46	4: 23	10: 53

Tabla 6-6. Tiempo de entrenamiento red CNN

6.4. LSTM + MLP

Utilizando la arquitectura especificada en la Sección 5.3.4 se han obtenido los resultados especificados en la Tabla 6-7 para todos los subconjuntos de datos.

	FD001	FD002	FD003	FD004
RMSE	15.71	12.44	15.23	13.80
Score	596.66	716.33	673.33	1049.98

Tabla 6-7. Resultados LSTM con MLP

Como podemos apreciar, con esta arquitectura se ha conseguido mejorar considerablemente en las bases de datos FD002 y FD004 como era objetivo y dónde tradicionalmente se habían tenido más problemas. Esto nos sugiere por tanto la importancia de que las unidades LSTM reciban los inputs con cierto preprocesamiento para maximizar su rendimiento. Por el contrario, en las bases de datos FD001 y FD003 donde las entradas tenían menos ruido, los resultados nos se han visto especialmente mejorados.

Este hecho del preprocesamiento mediante una red MLP antes de la LSTM puede ser comprobado más visualmente si estudiamos el funcionamiento de la red desde su interior. En la Figura 6-5 tenemos las entradas que recibe la red neuronal para la base de datos FD002. Se puede apreciar claramente el ruido que presentan las señales.

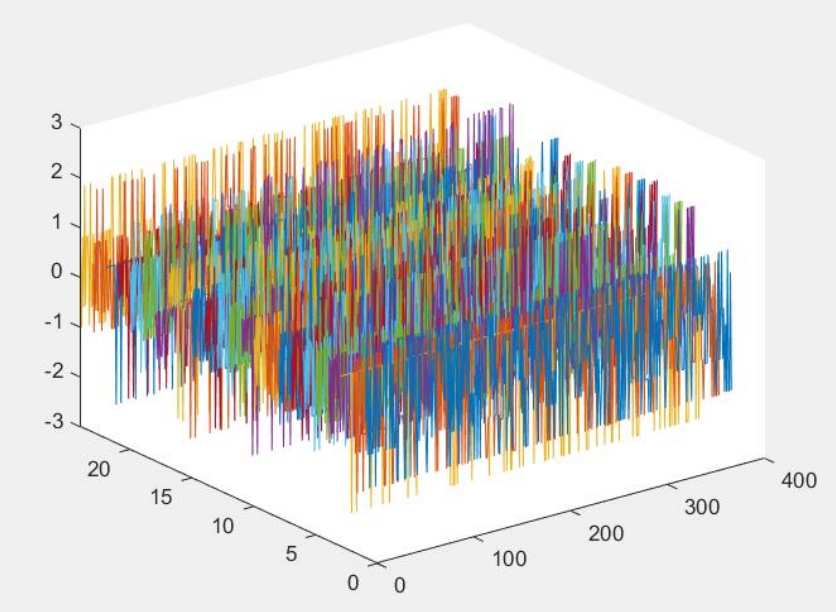


Figura 6-5. Entradas de la red neuronal para FD002

En la Figura 6-6 tenemos como serían las entradas para la red LSTM, una vez pasado el primer bloque de red MLP. En este caso ya se pueden observar tendencias en los datos que sugieren la degradación del motor. De todos modos, los valores son bastante variables y parecidos a lo que vimos en la Figura 6-2 para la red MLP.

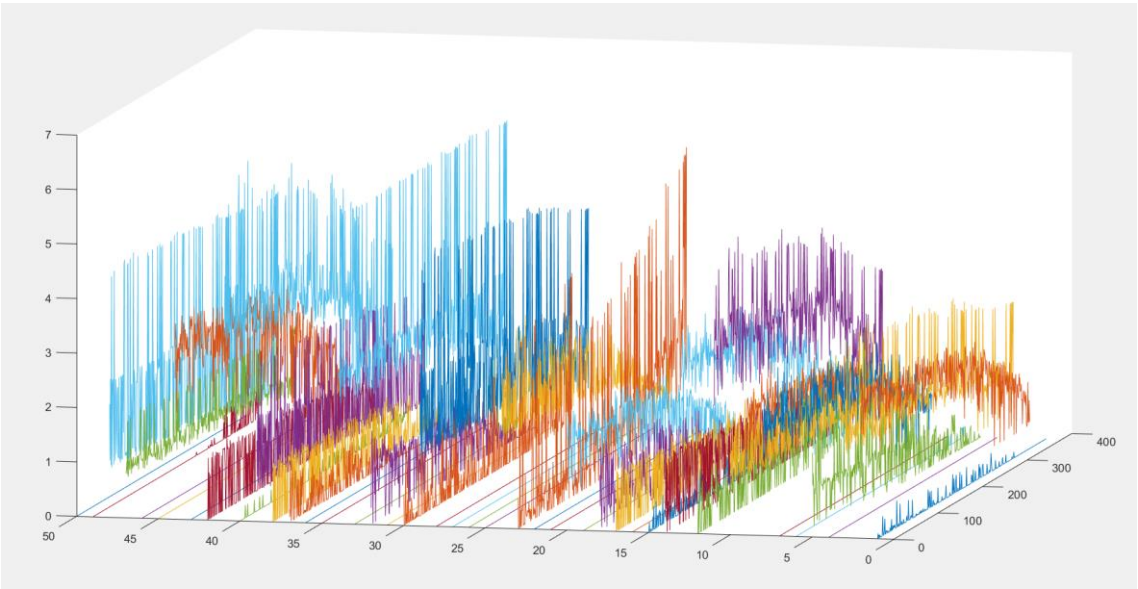


Figura 6-6. Entradas de la red LSTM para FD002

En la Figura 6-7 se presentan las salidas de la red LSTM. En este caso ya podemos ver secuencias mucho menos ruidosas, debido a la capacidad de las redes LSTM de tener en cuenta los pasos anteriores para realizar cada predicción.

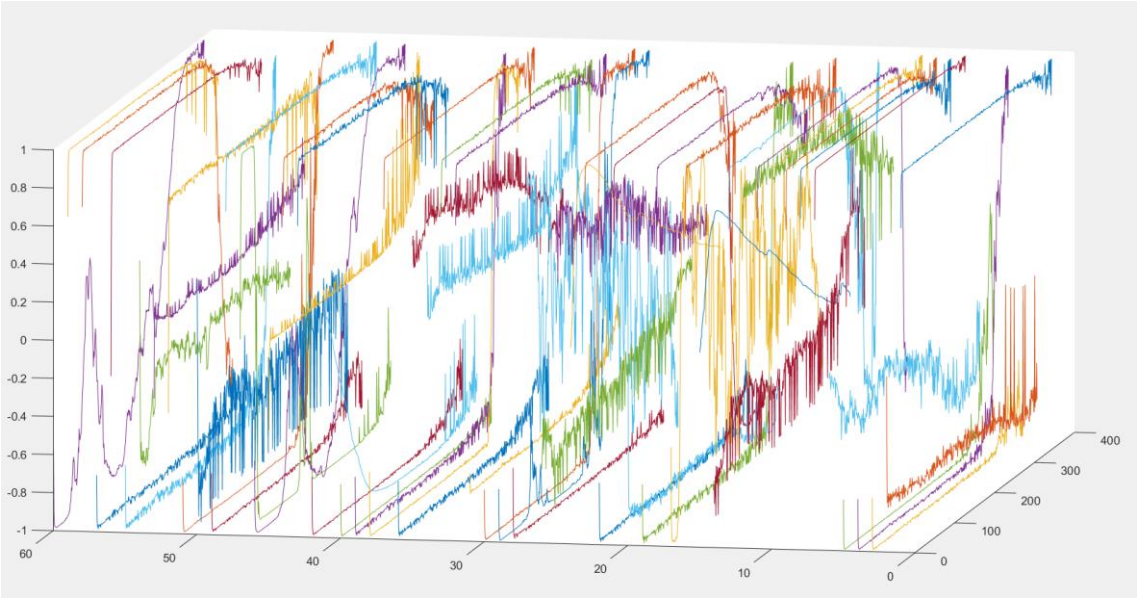


Figura 6-7. Salidas de la red LSTM para FD002

A parte de esto se ha testeado la red eliminando el primer bloque de MLP. Los resultados se recogen en la Tabla 6-8.

	FD001	**FD002**	**FD003**	**FD004**
RMSE	14.5886	17.08	14.32	27.55
Score	384	2522.25	458.33	6917.3

Tabla 6-8. Resultados red sin el primer bloque de MLP

Se puede observar claramente como los resultados de las bases de datos pares han empeorado notablemente sin esta primera capa de preprocesamiento. Por su parte los subconjuntos que no presentaban tanto ruido en sus entradas se han visto poco afectados, incluso se han obtenido resultados ligeramente mejores.

Cabe comentar que al igual que vimos en la anterior estructura de LSTM, en este caso también se trata de una arquitectura que tarda más entrenarse que las redes de otro tipo. En la Tabla 6-9 se recogen los tiempos de entrenamiento utilizados para cada uno de los subconjuntos.

	FD001	**FD002**	**FD003**	**FD004**
Tiempo [min]	8: 50	22: 12	14: 19	33: 37

Tabla 6-9. Tiempo de entrenamiento red LSTM+MLP

6.5. Comparación

Para tener una mejor perspectiva sobre los resultados obtenidos, en este apartado se realiza una comparación entre los diferentes métodos testeados en el presente trabajo, a la vez que una comparación de nuestra mejor solución con otras publicaciones recogidas en el estado del arte.

En la Tabla 6-10 se recoge la comparación de los resultados obtenidos junto a una media final de todos los valores para dar una idea del funcionamiento de la red en general, sobre todos los subconjuntos de datos.

Modelo	**FD001**		**FD002**		**FD003**		**FD004**		**Media**	
	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score
MLP	20.65	2128.7	21.84	9690	18.6	947.58	30.634	26621	22.931	9846.82
LSTM	18.42	769.80	22.82	2492.1	15.42	**402.39**	26.817	3784	20.81	2068.97
CNN	**14.27**	**360.84**	15.26	971.18	16.53	791.74	19.17	2312.20	16.31	1101.49
LSTM + MLP	15.71	596.66	**12.44**	**716.33**	**15.23**	673.33	**13.80**	**1049.98**	**14.30**	**759.08**

Tabla 6-10. Comparación entre las arquitecturas implementadas

Por otro lado, en la Tabla 6-11 se recoge la comparación entre diferentes publicaciones que hemos consultado en el estado del arte junto a la propuesta que mejores resultados ha obtenido en nuestro estudio.

Modelo	**FD001**		**FD002**		**FD003**		**FD004**		**Media**	
	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score
CNN [11]	12.61	273.7	22.36	10412	**12.64**	**284.1**	23.31	12466	17.73	5858.95
LSTM [14]	13.26	284.88	12.49	**571.4**	13.11	352.39	13.97	1252.32	**13.20**	**615.24**
imESN [20]	**10.41**	**197**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LSTM + MLP	15.71	596.66	**12.44**	716.33	15.23	673.33	**13.80**	**1049.98**	14.30	759.08

Tabla 6-11. Comparación con otras publicaciones

6.6. Método de umbral variable

Se ha aplicado el método del umbral variable explicado en la Sección 5.2.3, en un principio sobre la red LSTM + MLP sobre la que hemos obtenido los mejores resultados

en el caso anterior. Las pruebas realizadas nos han proporcionado los siguientes resultados.

- Se ha observado que, aunque no se establezca un umbral constante para todas las secuencias, la red tiende a establecer uno que coincide más o menos con la media de los umbrales variables que se han establecido a lo largo de las secuencias. Esto puede apreciarse en la Figura 6-8, donde para la base de datos FD002 las predicciones tienden a un valor de RUL constante a partir de 100, que sería la media de los umbrales para esta base de datos.

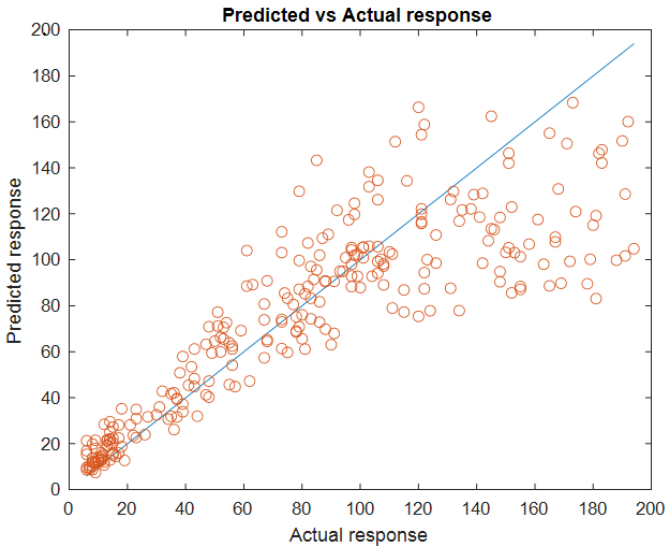


Figura 6-8. Predicciones de umbral variable con LSTM + MLP para FD002

Se ha contrastado este efecto en nuestra red CNN y esta tendencia se ha visto ostensiblemente más marcada. En la Figura 6-9 podemos ver este efecto.

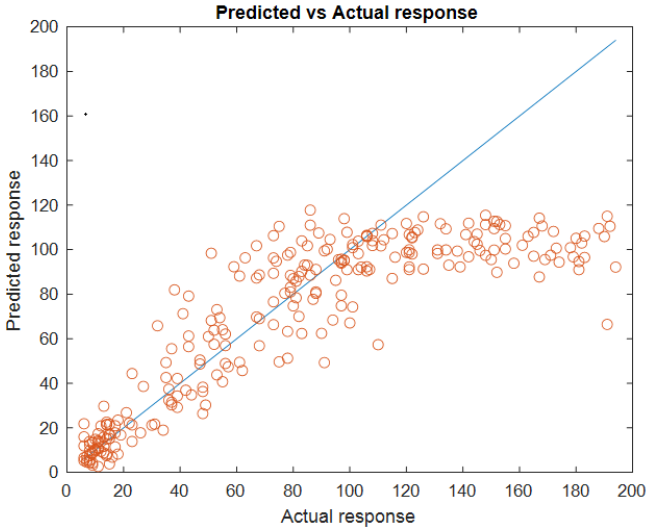


Figura 6-9. Predicciones de umbral variable con CNN para FD002

- Los resultados obtenidos con esta metodología no resultan en una mejora de los anteriores, aunque en la mayoría de las bases de datos se mantienen más o

menos similares. Cabe decir que para poder comparar los resultados ha sido necesario no tener en cuenta las predicciones por encima del umbral que hemos visto que establece la red. Se ha realizado una comparación con los resultados del apartado anterior recortándolos por el mismo punto. En la Tabla 6-12 se ve un resumen de los valores de RMSE obtenidos con esta metodología para la red LSTM + MLP.

	FD001	**FD002**	**FD003**	**FD004**
Umbral variable	14.86	13.709	15.123	17.383
RUL a trozos	15.61	12.53	15.19	14.84

Tabla 6-12. Comparación valores RMSE con el método de umbral variable

- Se ha comprobado también que el entrenamiento de la red con este método resulta sensiblemente más inestable.

6.7. Método de índice de salud exponencial

Se ha probado el funcionamiento de las redes modelizando la RUL del motor como una función exponencial según se explicaba en la Sección 5.2.3. Se han evaluado las predicciones obtenidas tanto en el formato exponencial como el de RUL tradicional utilizado. Esto se ha hecho simplemente despejando los ciclos en la Ecuación (5-5) con las predicciones obtenidas. Un problema que ha presentado esto es que, debido al logaritmo que aparece al despejar la Ecuación (5-5), existe la posibilidad de que cuando una predicción se va más allá del límite de degradación inicial establecido el despeje da números imaginarios. De todas maneras, en los resultados que se presentan en esta sección este hecho no ha ocurrido para ninguna base de datos en las redes entrenadas.

En la Tabla 6-13 se recogen los resultados obtenidos tanto para la red CNN como para la LSTM + MLP.

Modelo	**FD001**		**FD002**		**FD003**		**FD004**		**Media**	
	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score	RMSE	Score
CNN	17.76	429.52	24.772	5821.85	18.923	797.96	28.95	29150	22.60	9049.83
LSTM + MLP	12.609	208.62	10.68	400.31	24.17	65934.8	15.91	994.32	15.84	16884.3

Tabla 6-13. Comparación de predicciones de RUL con modelización exponencial

A primera vista algunos resultados no parecen muy impresionantes comparándolos con los recogidos en la Sección 6.5, pero la diferencia radica en que los anteriores recurrían a establecer un umbral para evitar las predicciones muy lejanas.

Destacan especialmente los resultados obtenidos para la red LSTM + MLP, donde se llega a resultados increíblemente precisos en bases de datos tradicionalmente complicadas y sin establecer ningún límite en la longitud máxima de las predicciones. Un ejemplo gráfico de esto puede verse en la Figura 6-10.

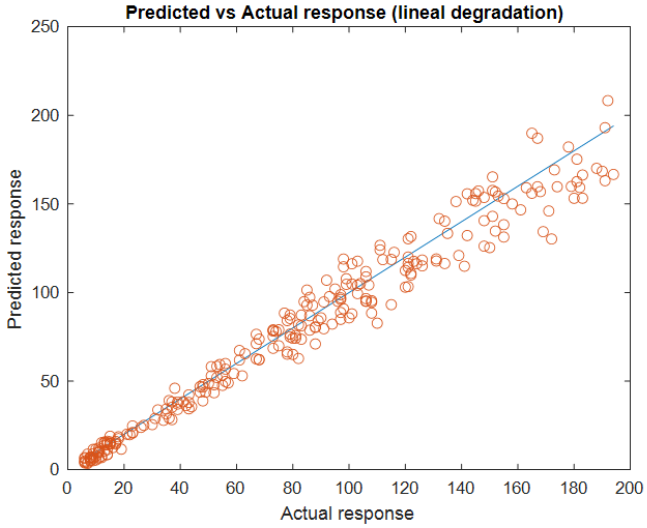


Figura 6-10. Predicciones con índice de salud exponencial mediante red LSTM + MLP para FD002

Se puede apreciar claramente como las predicciones se ajustan de manera llamativamente precisa a la línea azul, teniéndose la mejor predicción vista hasta el momento.

Un aspecto que se ha observado repetidamente en los resultados de ambas redes es la tendencia a realizar predicciones tempranas cuanto más lejana es la predicción. Esto se puede apreciar ligeramente en la Figura 6-10 y más claramente en la Figura 6-11.

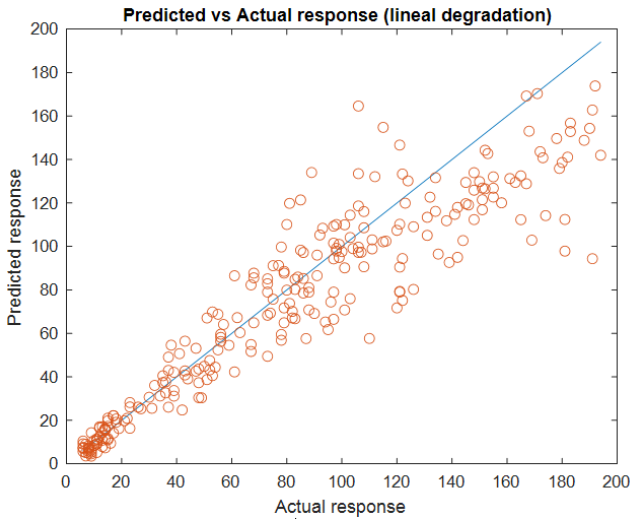


Figura 6-11. Predicciones con índice de salud exponencial mediante red CNN para FD002

Cabe comentar que no en todas las redes se han obtenido buenos resultados, las redes MLP y LSTM que habíamos usado anteriormente han presentado dificultades y unos resultados bastante pobres.

6.8. Discusión

De los resultados obtenidos en los apartados anteriores se pueden extraer diversas conclusiones.

Por un lado, afrontando el problema de la manera clásica modelizando la RUL a trozos, se han podido apreciar los buenos resultados que presenta el empleo de una red LSTM con cierto preprocesamiento. Esto presumiblemente también podría conseguirse modificando los datos con los que se va a entrenar la red, pero supondría llevar a cabo este tipo de preprocesamiento cada vez que los datos de los motores cambiasen, por lo que en principio el método de colocar una red MLP es preferible. Sin embargo, ha sido palpable que, para las bases de datos con datos menos ruidosos, este método de preprocesamiento no ha sido útil. Por otro lado, la red CNN, aunque con peores resultados también ha destacado por su precisión. Esto nos sugiere que el método "convolucional" es también muy capaz de trabajar con secuencias puras de datos y, tal como se ha visto en el estudio del arte, la popularidad que tiene actualmente su estudio es factible que lleve consigo grandes mejoras para este tipo de técnicas.

De la comparación de nuestra mejor solución con otras publicaciones del estado del arte podemos ver que nuestra propuesta ha conseguido, aunque sin ser la mejor globalmente, resultados competitivos. Existen numerosas publicaciones con las que era posible realizar la comparación, de todas formas, muchas de ellas son relativamente antiguas o con resultados poco competitivos por diversas razones. Para la comparación se ha tratado de utilizar propuestas relativamente actuales y que sean representativas de diferentes formas de afrontar el problema. Es por esto que se compara con una propuesta de redes RNN, otra de CNN y otra publicación que concentra su estudio en una sola base de datos obteniendo muy buenos resultados en ella, estrategia común en lo vistos en el estado del arte. Cabe decir que existirían más propuestas competitivas que podrían entrar en la comparación, pero que se han desestimado para no hacerla demasiado larga y por tampoco presentar mejoras relevantes en los resultados.

En el caso del método de umbral variable, es una idea que ya se comenta como una posibilidad en alguna publicación como [13]. No obstante, hemos visto como los resultados indican que quizá no es la mejor solución al problema que se presentaba. Cierto es que este método permite obtener un valor de umbral automáticamente y que este depende de una forma razonada de la base de datos que estamos estudiando. Aun así, no provoca mejoras en las predicciones del algoritmo y hace el entrenamiento más inestable. Por otro lado, mejorar las técnicas de *clustering* para conseguir una mejor caracterización de las fases del motor parece que tampoco repercutiría en la mejora de los resultados, al menos con este enfoque.

Finalmente, modelizando la RUL como una función exponencial se han obtenido unos resultados extraordinariamente precisos con ciertas arquitecturas. Por un lado, estos buenos resultados también se veían en [19] que hace uso de esta modelización, por lo que podría dar a suponer que este método es superior a la RUL a trozos. De todas maneras, como se establece en [15] por los creadores de la base de datos, esta función exponencial se utilizó como base para modelizar la degradación de distintas variables de los motores simulados en la base datos. Esto podría hacer suponer que estos buenos resultados pueden deberse más a la relación con el modelo de la simulación que de ser por sí mismo un método que presente ventajas claras.

7. Conclusiones y trabajos futuros

7.1. Conclusiones

A lo largo de este estudio se ha podido comprobar como los métodos de *ML*, y más específicamente de *DL*, se encuentran ya presentes en la mayoría de las áreas de la sociedad actual y son una de las puntas de lanza del avance tecnológico de nuestro tiempo.

Particularmente el mantenimiento predictivo se ha visto como un campo que despierta un gran interés para este tipo de métodos, siendo la base de datos C-MAPSS una herramienta extensamente conocida en el estudio de este tipo de problemas por investigadores de todo el mundo.

A lo largo de este trabajo se ha logrado consumar el ciclo completo que conlleva un estudio de *DL* hasta llegar a proponer una arquitectura que consigue ser competitiva dentro del extenso estado del arte que envuelve este problema. Se han estudiado diferentes posibilidades hasta llegar a una propuesta que destaca por su robustez a lo largo de las diferentes variantes de la base de datos como era el objetivo de partida. Más adelante desde un punto de vista más experimental se han estudiado posibilidades alternativas al enfoque común del problema que, aunque no presentan ventajas claras, aportan información de valor sobre líneas que se encuentran abiertas en el problema estudiado.

Desde un punto de vista más personal, la realización de este proyecto ha supuesto embarcarse en el estudio de un campo ajeno a mi formación académica pero que, como se ha podido ver a lo largo del trabajo, se encuentra ya en gran relación con la aeronáutica. Todo el proceso ha supuesto un aprendizaje apasionante y constante de una temática en gran parte desconocida a priori por lo que, más allá de los resultados obtenidos, el objetivo de crecer como profesional ampliando mi visión de la industria actual ha sido conseguido con creces.

7.2. Trabajo futuro

Aunque los resultados muestran la capacidad de las arquitecturas propuestas para realizar predicciones razonablemente precisas, se pueden entrever diversos caminos a recorrer en el desarrollo de este tipo de algoritmos.

Una de las opciones más claras es la de utilizar estas redes en datos de motores reales y no una base de datos preparada para la tarea. Aunque la base de datos se ha realizado buscando representar la realidad, las bases de datos reales pueden presentar nuevos retos como mayores longitudes de las secuencias, más y más complejos mecanismos de fallo, etc.

También sería interesante seguir explorando las técnicas de *clustering* en este tipo de datos, ya qué conocer en que fase se encuentra cada motor puede ser una información muy beneficiosa para el monitoreo de los motores.

Un poco unido a lo anterior, podría considerarse la posibilidad de que este tipo de algoritmos no sean totalmente supervisados como es el caso, de manera que puedan realizar el tratamiento de los datos que reciban por sí mismos. Para esto la utilización de mejores técnicas de *clustering* puede ser clave en el futuro.

8. Pliego de condiciones

8.1. Derechos y obligaciones de los trabajadores

Los derechos y obligaciones de los trabajadores se recogen en el Real Decreto 2/2015 del 23 de octubre. Se resumen en:

- Recibir las enseñanzas sobre materia en Seguridad e Higiene y sobre salvamento y socorrismo en los centros de trabajo que les sean facilitados por la empresa o en las instrucciones del Plan Nacional.
- Usar correctamente los medios de protección personal y cuidar de su perfecto estado de conservación.
- Dar cuenta inmediatamente a sus superiores de las averías y las deficiencias que puedan ocasionar peligros en cualquier centro o puesto de trabajo. Cuidar y mantener su higiene personal para evitar enfermedades contagiosas o molestias a los compañeros de trabajo.
- Someterse a los reconocimientos médicos preceptivos y vacunaciones o inmunizaciones ordenados por las Autoridades Sanitarias competentes o por el Servicio Médico de las Empresas.
- No introducir bebidas u otras sustancias no autorizadas en los centros de trabajo. Tampoco se podrá presentar o permanecer en los mismos en estado de embriaguez o de cualquier otro género de intoxicación.
- Cooperar en la extinción de siniestros y en el salvamento de las víctimas de accidentes de trabajo en las condiciones que, en cada caso, fueren racionalmente exigibles.
- Todo trabajador, después de solicitar de su inmediato superior medios de protección personal de carácter preceptivo para la realización de su trabajo, queda facultado para demostrar la ejecución de éste, en tanto no le sean facilitados dichos medios, si bien debería dar cuenta del hecho al Comité de Seguridad e Higiene o a uno de sus compañeros, sin perjuicio, además de ponerlo en conocimiento de la Inspección Provincial del Trabajo.

8.2. Condiciones generales de seguridad en los lugares de trabajo

8.2.1. Seguridad estructural

Los edificios y locales de los lugares de trabajo deberán poseer la estructura y solidez apropiadas a su tipo de utilización. Para las condiciones de uso previstas, todos sus elementos, estructurales o de servicio, incluidas las plataformas de trabajo, escaleras y escalas, deberán:

- Tener la solidez y la resistencia necesarias para soportar las cargas o esfuerzos a que sean sometidos.
- Disponer de un sistema de armado, sujeción o apoyo que asegure su estabilidad.

Se prohíbe sobrecargar los elementos citados en el apartado anterior. El acceso a techos o cubiertas que no ofrezcan suficientes garantías de resistencia sólo podrá autorizarse cuando se proporcionen los equipos necesarios para que el trabajo pueda realizarse de forma segura.

8.2.2. Espacios de trabajo y zonas peligrosas

Las dimensiones de los locales de trabajo deberán permitir que los trabajadores realicen su trabajo sin riesgos para su seguridad y salud y en condiciones ergonómicas aceptables. Sus dimensiones mínimas serán las siguientes:

- 3 metros de altura desde el piso hasta el techo. No obstante, en locales comerciales, de servicios, oficinas y despachos, la altura podrá reducirse a 2,5 metros.
- 2 metros cuadrados de superficie libre por trabajador.
- 10 metros cúbicos, no ocupados, por trabajador.

La separación entre los elementos materiales existentes en el puesto de trabajo será suficiente para que los trabajadores puedan ejecutar su labor en condiciones de seguridad, salud y bienestar. Cuando, por razones inherentes al puesto de trabajo, el espacio libre disponible no permita que el trabajador tenga la libertad de movimientos necesaria para desarrollar su actividad, deberá disponer de espacio adicional suficiente en las proximidades del puesto de trabajo.

Deberán tomarse las medidas adecuadas para la protección de los trabajadores autorizados a acceder a las zonas de los lugares de trabajo donde la seguridad de los trabajadores pueda verse afectada por riesgos de caída, caída de objetos y contacto o exposición a elementos agresivos. Asimismo, deberá disponerse, en la medida de lo posible, de un sistema que impida que los trabajadores no autorizados puedan acceder a dichas zonas.

Las zonas de los lugares de trabajo en las que exista riesgo de caída, de caída de objetos o de contacto o exposición a elementos agresivos, deberán estar claramente señalizadas.

8.2.3. Suelos, aberturas y desniveles, y barandillas

Los suelos de los locales de trabajo deberán ser fijos, estables y no resbaladizos, sin irregularidades ni pendientes peligrosas.

Las aberturas o desniveles que supongan un riesgo de caída de personas se protegerán mediante barandillas u otros sistemas de protección de seguridad equivalente, que podrán tener partes móviles cuando sea necesario disponer de acceso a la abertura.

Deberán protegerse, en particular:

- Las aberturas en los suelos.
- Las aberturas en paredes o tabiques, siempre que su situación y dimensiones suponga riesgo de caída de personas, y las plataformas, muelles o estructuras similares. La protección no será obligatoria, sin embargo, si la altura de caída es inferior a 2 metros.
- Los lados abiertos de las escaleras y rampas de más de 60 centímetros de altura. Los lados cerrados tendrán un pasamanos, a una altura mínima de 9O centímetros, si la anchura de la escalera es mayor de 1,2 metros; si es menor, pero ambos lados son cerrados, al menos uno de los dos llevará pasamanos.

Las barandillas serán de materiales rígidos, tendrán una altura mínima de 90 centímetros y dispondrán de una protección que impida el paso o deslizamiento por debajo de las mismas o la caída de objetos sobre personas.

8.2.4. Vías y salidas de evacuación

Las vías y salidas de evacuación, así como las vías de circulación y las puertas que den acceso a ellas, se ajustarán a lo dispuesto en su normativa específica.

En todo caso, y a salvo de disposiciones específicas de la normativa citada, dichas vías y salidas deberán satisfacer las condiciones que se establecen en los siguientes puntos de este apartado.

Las vías y salidas de evacuación deberán permanecer expeditas y desembocar lo más directamente posible en el exterior o en una zona de seguridad.

En caso de peligro, los trabajadores deberán poder evacuar todos los lugares de trabajo rápidamente y en condiciones de máxima seguridad.

El número, la distribución y las dimensiones de las vías y salidas de evacuación dependerán del uso, de los equipos y de las dimensiones de los lugares de trabajo, así como del número máximo de personas que puedan estar presentes en los mismos.

Las puertas de emergencia deberán abrirse hacia el exterior y no deberán estar cerradas, de forma que cualquier persona que necesite utilizarlas en caso de urgencia pueda abrirlas fácil e inmediatamente. Estarán prohibidas las puertas específicamente de emergencia que sean correderas o giratorias.

Las puertas situadas en los recorridos de las vías de evacuación deberán estar señalizadas de manera adecuada. Se deberán poder abrir en cualquier momento desde el interior sin ayuda especial. Cuando los lugares de trabajo estén ocupados, las puertas deberán poder abrirse.

Las vías y salidas específicas de evacuación deberán señalizarse conforme a lo establecido en el Real Decreto 485/1997, de 14 de abril, sobre disposiciones mínimas de

señalización de seguridad y salud en el trabajo. Esta señalización deberá fijarse en los lugares adecuados y ser duradera.

Las vías y salidas de evacuación, así como las vías de circulación que den acceso a ellas, no deberán estar obstruidas por ningún objeto de manera que puedan utilizarse sin trabas en cualquier momento. Las puertas de emergencia no deberán cerrarse con llave.

En caso de avería de la iluminación las vías y salidas de evacuación que requieran iluminación deberán estar equipadas con iluminación de seguridad de suficiente intensidad.

8.2.5. Condiciones de protección contra incendios

Los lugares de trabajo deberán ajustarse a lo dispuesto en la normativa que resulte de aplicación sobre condiciones de protección contra incendios.

En todo caso, y a salvo de disposiciones especificas de la normativa citada, dichos lugares deberán satisfacer las condiciones que se señalan en los siguientes puntos de este apartado.

Según las dimensiones y el uso de los edificios, los equipos, las características físicas y químicas de las sustancias existentes, así como el número máximo de personas que puedan estar presentes, los lugares de trabajo deberán estar equipados con dispositivos adecuados para combatir los incendios y, si fuere necesario, con detectores contra incendios y sistemas de alarma.

Los dispositivos no automáticos de lucha contra los incendios deberán ser de fácil acceso y manipulación. Dichos dispositivos deberán señalizarse conforme a lo dispuesto en el Real Decreto 485/1997, de 14 de abril, sobre disposiciones mínimas de señalización de seguridad y salud en el trabajo. Dicha señalización deberá fijarse en los lugares adecuados y ser duradera.

8.2.6. Disposiciones mínimas de equipo

En un puesto de trabajo con pantallas de visualización, es de vital importancia poner atención en el diseño de la disposición del material y la ergonomía del equipo del que se va a hacer uso a lo largo del mismo. Tal y como recoge el Real Decreto 488/1997, la utilización en sí misma del equipo no debe ser una fuente de riesgo para los trabajadores.

- Pantalla: Los caracteres de la pantalla deberán estar bien definidos y configurados de forma clara, y tener una dimensión suficiente, disponiendo de un espacio adecuado entre los caracteres y los renglones.

  La imagen de la pantalla deberá ser estable, sin fenómenos de destellos, centelleos u otras formas de inestabilidad.

El usuario de terminales con pantalla deberá poder ajustar fácilmente la luminosidad y el contraste entre los caracteres y el fondo de la pantalla, y adaptarlos fácilmente a las condiciones del entorno.

La pantalla deberá ser orientable e inclinable a voluntad, con facilidad para adaptarse a las necesidades del usuario.

Podrá utilizarse un pedestal independiente o una mesa regulable para la pantalla.

La pantalla no deberá tener reflejos ni reverberaciones que puedan molestar al usuario.

- Teclado: El teclado deberá ser inclinable e independiente de la pantalla para permitir que el trabajador adopte una postura cómoda que no provoque cansancio en los brazos o las manos.

  Tendrá que haber espacio suficiente delante del teclado para que el usuario pueda apoyar los brazos y las manos.

  La superficie del teclado deberá ser mate para evitar los reflejos.

  La disposición del teclado y las características de las teclas deberán tender a facilitar su utilización.

  Los símbolos de las teclas deberán resaltar suficientemente y ser legibles desde la posición normal de trabajo.

- Mesa o superficie de trabajo La mesa o superficie de trabajo deberán ser poco reflectantes, tener dimensiones suficientes y permitir una colocación flexible de la pantalla, del teclado, de los documentos y del material accesorio.

  El soporte de los documentos deberá ser estable y regulable y estará colocado de tal modo que se reduzcan al mínimo los movimientos incómodos de la cabeza y los ojos.

  El espacio deberá ser suficiente para permitir a los trabajadores una posición cómoda.

- Asiento de trabajo: El asiento de trabajo deberá ser estable, proporcionando al usuario libertad de movimiento y procurándole una postura confortable.

  La altura del mismo deberá ser regulable.

  El respaldo deberá ser reclinable y su altura ajustable.

  Se pondrá un reposapiés a disposición de quienes lo deseen.

8.2.7. Disposiciones mínimas del entorno

Al igual que ocurre con el equipo de trabajo, deberá existir unas disposiciones mínimas para el entorno, atendiendo al Real Decreto 488/1997, de 14 de abril, el entorno deberá atender a:

- Espacio: El puesto de trabajo deberá tener una dimensión suficiente y estar acondicionado de tal manera que haya espacio suficiente para permitir los cambios de postura y movimientos de trabajo.
- Iluminación: La iluminación general y la iluminación especial (lámparas de trabajo), cuando sea necesaria, deberán garantizar unos niveles adecuados de iluminación y unas relaciones adecuadas de luminancias entre la pantalla y su entorno, habida cuenta del carácter del trabajo, de las necesidades visuales del usuario y del tipo de pantalla utilizado.

  El acondicionamiento del lugar de trabajo y del puesto de trabajo, así como la situación y las características técnicas de las fuentes de luz artificial, deberán coordinarse de tal manera que se eviten los deslumbramientos y los reflejos molestos en la pantalla u otras partes del equipo.
- Reflejos y deslumbramientos: Los puestos de trabajo deberán instalarse de tal forma que las fuentes de luz, tales como ventanas y otras aberturas, los tabiques transparentes o translúcidos y los equipos o tabiques de color claro no provoquen deslumbramiento directo ni produzcan reflejos molestos en la pantalla. Las ventanas deberán ir equipadas con un dispositivo de cobertura adecuado y regulable para atenuar la luz del día que ilumine el puesto de trabajo.
- Ruido: El ruido producido por los equipos instalados en el puesto de trabajo deberá tenerse en cuenta al diseñar el mismo, en especial para que no se perturbe la atención ni la palabra.
- Calor: Los equipos instalados en el puesto de trabajo no deberán producir un calor adicional que pueda ocasionar molestias a los trabajadores.
- Emisiones: Toda radiación, excepción hecha de la parte visible del espectro electromagnético, deberá reducirse a niveles insignificantes desde el punto de vista de la protección de la seguridad y de la salud de los trabajadores.
- Humedad: Deberá crearse y mantenerse una humedad aceptable.

8.2.8. Disposición mínima en la interconexión ordenador/persona

Para la elaboración, la elección, la compra y la modificación de programas, así como para la definición de las tareas que requieran pantallas de visualización, el empresario tendrá en cuenta los siguientes factores:

- El programa habrá de estar adaptado a la tarea que deba realizarse.

- El programa habrá de ser fácil de utilizar y deberá en su caso, poder adaptarse al nivel de conocimientos y de experiencia del usuario; no deberá utilizarse ningún dispositivo cuantitativo o cualitativo de control sin que los trabajadores hayan sido informados y previa consulta con sus representantes.
- Los sistemas deberán proporcionar a los trabajadores indicaciones sobre su desarrollo.
- Los sistemas deberán mostrar la información en un formato y a un ritmo adaptados a los operadores.
- Los principios de ergonomía deberán aplicarse en particular al tratamiento de la información por parte de la persona.

9. Presupuesto

Para finalizar, en el presente capítulo se recogen los costes que ha implicado la realización del presente Trabajo de Fin de Máster. La divisa de referencia utilizada en este presupuesto ha sido el euro (EUR).

Dada la naturaleza del proyecto podemos realizar un desglose de los costes en dos categorías:

- **Costes humanos:** hace referencia a los costes percibidos por la participación del personal asignado al proyecto.
- **Costes materiales:** hace referencia a los costes derivados de las herramientas tanto físicas como informáticas necesarias para la realización del presente proyecto.

9.1. Costes humanos

Este coste hace referencia al salario que deberían percibir los participantes del proyecto según su grado de formación y tiempo empleado.

- **Ingeniero Aeronáutico Junior:** encargado de la búsqueda de información, desarrollo de los algoritmos, realización de las pruebas y creación del documento.
- **Doctor Ingeniero Aeronáutico:** su tarea consiste en participar en el establecimiento de los objetivos y metodología de realización del proyecto, supervisión del proceso de desarrollo y revisión de los resultados finales.

En la Tabla 9-1 se recogen los gastos asociados al trabajo realizado por cada agente junto al valor total de los gastos humanos en el proyecto.

Categoría	Precio [EUR/h]	Tiempo [h]	Coste [EUR]
Ing. Aeronáutico Junior	10	600	6000
Doctor Ing. Aeronáutico	25	20	500
TOTAL			**6500**

Tabla 9-1. Desglose de los costes humanos

9.2. Costes materiales

Dentro de este apartado se requiere distinguir entre Hardware y Software.

9.2.1. Costes de Hardware

Este apartado recoge los costes asociados al equipo físico utilizado para la realización del proyecto. El desglose de gastos se recoge en la Tabla 9-2. Se ha considerado que el uso de este equipo ha sido por un tiempo limitado por lo que se tiene en cuenta el coste de amortización.

Producto	Coste [EUR]	Periodo de Amortización [meses]	Tiempo [meses]	Coeficiente de amortización [-]	Subtotal [EUR]
PC	1300	72	6	0.2	86.67
Pantalla Samsung	120	72	6	0.2	8
TOTAL					**94.67**

Tabla 9-2. Desglose de los costes de Hardware

9.2.2. Costes de Software

Este apartado recoge los gastos asociados a las licencias de los programas empleados en el proyecto. Al igual que en el hardware también se ha tenido en cuenta el coste de amortización. La Tabla 9-3 recoge el desglose de los gastos.

Producto	Coste [EUR]	Periodo de Amortización [meses]	Tiempo [meses]	Subtotal [EUR]
MATLAB R2022a	800	72	6	400
Microsoft Office	579	72	6	289.5
TOTAL				**689.5**

Tabla 9-3. Desglose de los costes de Software

9.3. Presupuesto total

Para finalizar se adjunta el desglose total en la Tabla 9-4 uniendo todos los costes recogidos en los apartados anteriores. A los gastos previos se ha unido una entrada de costes generales que hace referencia a gastos imprevistos y medios auxiliares, como energía o limpieza, que se ha estimado en un 5% de la suma de costes humanos y hardware. Además, también se han de incluir los costes asociados a los impuestos en el presupuesto final del proyecto.

Categoría	Coste [EUR]
Costes humanos	6500
Costes de Hardware	94.67
Costes de Software	689.5
Costes generales	329.73
Total sin impuestos	7613.9
IVA (21%)	1598.9
TOTAL	**9212.8**

Tabla 9-4. Desglose general

10. Bibliografía

[1] «Big Data in Aviation - Reduce Costs trough Predictive Maintenance — EXSYN», *EXSYN Aviation Solutions*. https://www.exsyn.com/blog/big-data-in-aviation-predictive-maintenance (accedido 2 de julio de 2022).
[2] «Big Data: ¿En qué consiste? Su importancia, desafíos y gobernabilidad». https://www.powerdata.es/big-data (accedido 23 de agosto de 2022).
[3] «FOQA - Flight Data Analysis of Aircraft for Flight Safety». https://www.theairlinepilots.com/forumarchive/flightsafety/foqaflightdataanalysis.php (accedido 17 de marzo de 2022).
[4] «Big Data in Aviation - Reduce Costs trough Predictive Maintenance», *EXSYN Aviation Solutions*. https://www.exsyn.com/blog/big-data-in-aviation-predictive-maintenance (accedido 17 de marzo de 2022).
[5] «Turbofan engine degradation simulation data set | NASA Open Data Portal». https://data.nasa.gov/Aerospace/Turbofan-engine-degradation-simulation-data-set/vrks-gjie (accedido 17 de marzo de 2022).
[6] I. Sanz Berbegal, «Diseño de métodos de detección y diagnóstico de fallas en rodamientos de aerogeneradores mediante el análisis de vibraciones por medio de algoritmos de machine learning», Proyecto/Trabajo fin de carrera/grado, Universitat Politècnica de València, 2021. Accedido: 9 de julio de 2022. [En línea]. Disponible en: https://riunet.upv.es/handle/10251/174309
[7] P. Muñoz y J. Segundo, «Wind Turbine Blade Damage Identification using Deep Learning Algorithms», ene. 2020, Accedido: 9 de julio de 2022. [En línea]. Disponible en: https://riunet.upv.es/handle/10251/133954
[8] «Deep Learning in Aeronautics: Air Traffic Trajectory Classification Based on Weather Reports». https://riunet.upv.es/handle/10251/160616 (accedido 9 de julio de 2022).
[9] I. CORPORATIVA, «Mantenimiento predictivo: la técnica basada en datos clave para anticipar errores», *Iberdrola*. https://www.iberdrola.com/innovacion/mantenimiento-predictivo (accedido 9 de julio de 2022).
[10] I. Goodfellow, Y. Bengio, y A. Courville, *Deep Learning*. Cambridge, Massachusetts, 2016.
[11] X. Li, Q. Ding, y J.-Q. Sun, «Remaining useful life estimation in prognostics using deep convolution neural networks», *Reliab. Eng. Syst. Saf.*, vol. 172, pp. 1-11, abr. 2018, doi: 10.1016/j.ress.2017.11.021.
[12] C. W. Hong, C. Lee, K. Lee, M.-S. Ko, D. E. Kim, y K. Hur, «Remaining Useful Life Prognosis for Turbofan Engine Using Explainable Deep Neural Networks with Dimensionality Reduction», *Sensors*, vol. 20, n.º 22, Art. n.º 22, ene. 2020, doi: 10.3390/s20226626.
[13] S. Zheng, K. Ristovski, A. Farahat, y C. Gupta, «Long Short-Term Memory Network for Remaining Useful Life estimation», en *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, jun. 2017, pp. 88-95. doi: 10.1109/ICPHM.2017.7998311.
[14] A. Chaoub, A. Voisin, C. Cerisara, y B. Iung, «Learning representations with end-to-end models for improved remaining useful life prognostics». arXiv, 26 de mayo de 2021. doi: 10.48550/arXiv.2104.05049.
[15] «Damage propagation modeling for aircraft engine run-to-failure simulation | IEEE Conference Publication | IEEE Xplore». https://ieeexplore.ieee.org/document/4711414 (accedido 18 de marzo de 2022).
[16] C.-G. Huang, H.-Z. Huang, y Y.-F. Li, «A Bidirectional LSTM Prognostics Method Under Multiple Operational Conditions», *IEEE Trans. Ind. Electron.*, vol. 66, n.º 11, pp. 8792-8802, nov. 2019, doi: 10.1109/TIE.2019.2891463.

[17] E. Zvornicanin, «Differences Between Bidirectional and Unidirectional LSTM | Baeldung on Computer Science», 25 de enero de 2022. https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm (accedido 2 de agosto de 2022).
[18] C. Peng, Y. Chen, Q. Chen, Z. Tang, L. Li, y W. Gui, «A Remaining Useful Life Prognosis of Turbofan Engine Using Temporal and Spatial Feature Fusion», *Sensors*, vol. 21, n.º 2, Art. n.º 2, ene. 2021, doi: 10.3390/s21020418.
[19] U. Thakkar y H. Chaoui, «Remaining Useful Life Prediction of an Aircraft Turbofan Engine Using Deep Layer Recurrent Neural Networks», *Actuators*, vol. 11, n.º 3, Art. n.º 3, mar. 2022, doi: 10.3390/act11030067.
[20] C. Peng, Y. Chen, W. Gui, Z. Tang, y C. Li, «Remaining useful life prognosis of turbofan engines based on deep feature extraction and fusion», *Sci. Rep.*, vol. 12, n.º 1, Art. n.º 1, abr. 2022, doi: 10.1038/s41598-022-10191-2.
[21] K. Javed, R. Gouriveau, y N. Zerhouni, «Novel failure prognostics approach with dynamic thresholds for machine degradation», en *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, nov. 2013, pp. 4404-4409. doi: 10.1109/IECON.2013.6699844.
[22] L. González, «¿Qué es el Perceptrón? Perceptrón Simple y Multicapa», *Aprende IA*, 5 de octubre de 2021. https://aprendeia.com/que-es-el-perceptron-simple-y-multicapa/ (accedido 19 de julio de 2022).
[23] «Introducción Práctica a Deep Learning (Taller 1)». https://gustavoaguilar.io/data-science-el-salvador-taller-intro-deep-learning/ (accedido 6 de septiembre de 2022).
[24] «ANN vs CNN vs RNN | Types of Neural Networks», *Analytics Vidhya*, 17 de febrero de 2020. https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/ (accedido 21 de marzo de 2022).
[25] «Recurrent Neural Network (RNN): ¿de qué se trata?», *Formación en ciencia de datos | DataScientest.com*, 14 de diciembre de 2021. https://datascientest.com/es/recurrent-neural-network-rnn-de-que-se-trata (accedido 22 de marzo de 2022).
[26] A. Eliasy y J. Przychodzen, «The role of AI in capital structure to enhance corporate funding strategies», *Array*, vol. 6, p. 100017, jul. 2020, doi: 10.1016/j.array.2020.100017.
[27] «Understanding LSTM Networks -- colah's blog». http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accedido 22 de marzo de 2022).
[28] «What are Convolutional Neural Networks?», 6 de enero de 2021. https://www.ibm.com/cloud/learn/convolutional-neural-networks (accedido 22 de marzo de 2022).
[29] J. Brownlee, «Overfitting and Underfitting With Machine Learning Algorithms», *Machine Learning Mastery*, 20 de marzo de 2016. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ (accedido 26 de agosto de 2022).
[30] P. Radhakrishnan, «What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?», *Medium*, 18 de octubre de 2017. https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a (accedido 18 de abril de 2022).
[31] J. Brownlee, «A Gentle Introduction to Dropout for Regularizing Deep Neural Networks», *Machine Learning Mastery*, 2 de diciembre de 2018. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/ (accedido 18 de abril de 2022).
[32] J. Brownlee, «How to Choose an Activation Function for Deep Learning», *Machine Learning Mastery*, 17 de enero de 2021. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (accedido 18 de abril de 2022).
[33] «Activation Functions in Neural Networks [12 Types & Use Cases]». https://www.v7labs.com/blog/neural-networks-activation-functions, https://www.v7labs.com/blog/neural-networks-activation-functions (accedido 23 de julio de 2022).
[34] «A Practical Guide To Hyperparameter Optimization», *AI & Machine Learning Blog*, 19 de mayo de 2021. https://nanonets.com/blog/hyperparameter-optimization/ (accedido 18 de abril de 2022).

[35] W. Koehrsen, «A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning», *Medium*, 2 de julio de 2018. https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f (accedido 18 de abril de 2022).
[36] Orlando, «¿Qué es el Clustering? | Detección de Comunidades», *GraphEverywhere*, 21 de agosto de 2020. https://www.grapheverywhere.com/que-es-el-clustering/ (accedido 7 de agosto de 2022).
[37] «Machine Learning: Qué es y para qué sirve el Clustering • Kueski Blog», 12 de mayo de 2018. https://kueski.com/blog/company/life-at-kueski/machine-learning-que-es-clustering/ (accedido 6 de septiembre de 2022).
[38] «Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), Version 2(LEW-18315-2) | NASA Software Catalog». https://software.nasa.gov/software/LEW-18315-2 (accedido 17 de marzo de 2022).
[39] D. Frederick, J. DeCastro, y J. Litt, «User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)», *NASA Tech. Manuscr.*, vol. 2007–215026, ene. 2007.
[40] «MATLAB - El lenguaje del cálculo técnico». https://es.mathworks.com/products/matlab.html (accedido 4 de septiembre de 2022).
[41] «List of Deep Learning Layers - MATLAB & Simulink - MathWorks España». https://es.mathworks.com/help/deeplearning/ug/list-of-deep-learning-layers.html (accedido 3 de agosto de 2022).
[42] «Define Custom Deep Learning Layers - MATLAB & Simulink - MathWorks España». https://es.mathworks.com/help/deeplearning/ug/define-custom-deep-learning-layers.html (accedido 3 de agosto de 2022).
[43] «Options for training deep learning neural network - MATLAB trainingOptions - MathWorks España». https://es.mathworks.com/help/deeplearning/ref/trainingoptions.html (accedido 3 de agosto de 2022).
[44] «TensorFlow», *TensorFlow*. https://www.tensorflow.org/?hl=es-419 (accedido 23 de agosto de 2022).
[45] «PyTorch». https://www.pytorch.org (accedido 23 de agosto de 2022).
[46] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, y K. Keutzer, «SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size». arXiv, 4 de noviembre de 2016. Accedido: 23 de agosto de 2022. [En línea]. Disponible en: http://arxiv.org/abs/1602.07360
[47] C. Szegedy *et al.*, «Going Deeper with Convolutions». arXiv, 16 de septiembre de 2014. doi: 10.48550/arXiv.1409.4842.
[48] «Design and run experiments to train and compare deep learning networks - MATLAB - MathWorks España». https://es.mathworks.com/help/deeplearning/ref/experimentmanager-app.html (accedido 3 de agosto de 2022).
[49] «Build Networks with Deep Network Designer - MATLAB & Simulink - MathWorks España». https://es.mathworks.com/help/deeplearning/ug/build-networks-with-deep-network-designer.html (accedido 3 de agosto de 2022).
[50] «(1) (PDF) From real data to remaining useful life estimation : an approach combining neuro-fuzzy predictions and evidential Markovian classifications.» https://www.researchgate.net/publication/44709653_From_real_data_to_remaining_useful_life_estimation_an_approach_combining_neuro-fuzzy_predictions_and_evidential_Markovian_classifications?enrichId=rgreq-08187e6d017d5887be63f1e5e8dfb463-XXX&enrichSource=Y292ZXJQYWdlOzQ0NzA5NjUzO0FTOjEwMjEwMDEyMTk0ODE2M0AxNDAxMzUzOTc0NTc4&el=1_x_3&_esc=publicationCoverPdf (accedido 4 de septiembre de 2022).
[51] M. Morty, «Convolutional Neural Networks for Sequence Processing: Part 1», *Medium*, 28 de septiembre de 2018. https://tech-morty.medium.com/convolutional-neural-networks-for-sequence-processing-part-1-420dd9b500 (accedido 6 de agosto de 2022).