Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Máster en Ingeniería Industrial

MASTER EN INGENIERÍA INDUSTRIAL

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

TRABAJO FIN DE MÁSTER

Aplicación de Deep Learning para la detección e identificación de fallos en una planta EDAR

Autor: D. Jaime Trigueros Suárez
Tutor: D. Gregorio Sanz Palmero

Valladolid, Febrero, 2022

RESUMEN

El actual crecimiento que está sufriendo la inteligencia artificial hace de esta técnica que sea un punto clave y estratégico en el mundo de la industria.

Con la evolución de la industria y el aumento de exigencia de la producción hacen tener en cuenta y desarrollar los diferentes métodos de control de la producción. Es por ello que es muy importante la detección e identificación de los fallos del proceso productivo.

Teniendo en cuenta lo citado anteriormente, el presente proyecto nace de la necesidad de proporcionar una respuesta y adaptación a las nuevos requisitos que exige la industria, ofreciendo técnicas innovadoras en la detección de fallos. La técnica utilizada, proveniente de la inteligencia artificial, es la de aprendizaje profundo, haciéndose uso de redes neuronales. El programa creado para la realización del proyecto se ha realizado de tal forma que corra de manera autónoma y automática, aprendiendo por sí solo.

Los resultados obtenidos en el este proyecto ponen de manifiesto la gran efectividad que dispone el aprendizaje profundo para la detección de fallos.

Palabras claves;

Aprendizaje profundo, redes neuronales, aprendizaje automatizado, Python, detección y clasificación de fallos.

ABSTRACT

The current growth that is hiding artificial intelligence makes this technique a key and strategic point in the world of industry.

With the evolution of the industry and the increase in production demands, the different production control methods are taken into account and developed. That is why it is very important to detect and identify failures in the production process.

Taking into account the aforementioned, this project arises from the need to provide a response and adaptation to the new requirements demanded by the industry, offering innovative techniques in fault detection. The technique used, coming from artificial intelligence, is deep learning, making use of neural networks. The program created for the realization of the project has been made in such a way that it runs autonomously and automatically, learning by itself.

The results obtained in this project show the great effectiveness of deep learning for fault detection.

Keywords;

Deep Learning, neural networks, Machine Learning, Python, Fault and isolations defaults.

AGRADECIMIENTOS

Este Trabajo Fin de Master es fruto y cosecha de los conocimientos y competencias adquiridos durante el Master de Ingeniería Industrial impartido en la escuela de Ingenieros Industriales de la Universidad de Valladolid.

Dar las gracias a la Universidad de Granada, así como al departamento de "Ingeniería de Sistemas y Automática" de la escuela de Ingenieros Industriales de la Universidad de Valladolid, por el apoyo, soporte y material aportado para la satisfactoria realización de este proyecto.

También agradecer a mis familiares y seres cercanos por el apoyo dado durante la realización del Master.

Jaime Trigueros Suárez

Valladolid, 14 de Febrero de 2022

ÍNDICE

Capítulo 1
INTRODUCCIÓN Y OBJETIVOS

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Introducción

En la actualidad en la que vivimos el mundo industrial ha sufrido de un gran avance en las últimas décadas. Es por ello que las exigencias de la industria se hayan ajustado a estos avances teniendo unas exigencias mayores. Estas exigencias actuales hacen que sea necesario una monitorización del proceso productivo industrial. En paralelo a este avance del mundo industrial también las tecnologías han sufrido un gran desarrollo. Este último avance de las tecnologías permite realizar un procesamiento de datos mayor, facilitando de esta forma la monitorización del propio proceso productivo.

Es por lo citado en el anterior párrafo que la detección de fallos en el proceso productivo toma un valor especial y es un punto táctico de toda fábrica, haciendo de la inteligencia artificial un buen candidato para la detección de estos fallos.

La inteligencia artificial cuenta con grandes métodos de identificación de fallos, destacando el *Deep Learning*, siendo una de las muchas técnicas del *Machine Learning*.

La base de este proyecto es asentar los conocimientos obtenidos en la asignatura de control del Master de Ingeniería Industrial cursado en la escuela de Ingenieros Industriales de la Universidad de Valladolid y adquirir conocimientos en la detección de fallos haciendo uso del *Deep Learning* (DL).

Para ello se ha analizado un proceso de una estación depuradora de aguas residuales (EDAR), aplicando el *Machine Learning* con la técnica de *Deep Convolutional Neural Networks* (DCNN) para la detección y clasificación de los posibles fallos que pueda tener la estación. La base de datos de esta depuradora será la proporcionada por la universidad y denominada *Benchmark Simulation Model no.2* (BSM2), se puede encontrar más informacion sobre el BSM2 en la ref [1]. No obstante el programa realizado puede ser de aplicación a otras bases de datos.

También se pone en conocimiento en esta memoria, otras posibles técnicas a emplear en la detección de fallos. Se adentrará también dentro de las redes neuronales artificial (ANN) y se explicará los posibles problemas a solventar en la técnica empleada de DCNN.

1.2. Objetivos

La intención del presente trabajo es la de adentrarse en el apasionante mundo de la inteligencia artificial, aprendiendo las técnicas del *Deep Learning*(DL).

Los objetivos a alcanzar con este proyecto son;

- Estudio y aplicación introductoria de las técnicas de detección y diagnóstico de fallos.
- Estudio y aprendizaje de las ideas, definiciones y formulaciones básicas relacionadas *Deep Learning* (DL).
- Estudio y aplicación de las técnicas de procesamiento de datos en entornos FDI & Dl.

- Aplicación y análisis de modelos DL para FDI en una planta compleja: *benchmark* estación de depuración de aguas residuales (EDAR).

A mayores de los objetivos generales del proyecto también se han establecido objetivos a la parte práctica del proyecto. Los objetivos de la parte práctica son los siguientes;

El objetivo principal es la detección automática del fallo, es decir, saber cuándo el proceso productivo está funcionando en modo normal o en modo fallo, a través de los datos que se introducen en el código.

A mayores del objetivo principal citado anteriormente también se ha establecido un objetivo secundario de clasificación del tipo de modo de fallo.

El primer objetivo de detección de fallo es el de mayor importancia, ya que nos dirá cuando el proceso entra en modo fallo, pudiendo ir el operario a intervenir en el mismo proceso, en cambio el segundo objetivo de menor importancia que el primero, servirá al operario para una primera valoración y análisis de donde puede venir el fallo del proceso.

Resumiendo:

1er Objetivo: Identificación funcionamiento normal o modo fallo.

2do Objetivo: Clasificación del modo de fallo.

1.3. Organización de la memoria

Para la satisfacción de los objetivos citados anteriormente, se ha realizado esta memoria la cual se ha organizado en un total de 7 capítulos. Estos capítulos son el siguiente;

- **Capítulo 2: Estado del Arte**

Este capítulo está dedicado a repasar las técnicas de detección de fallos y a si mismo se realiza un repaso de las técnicas de *Machine Learning* y *Deep Learning*, explicando toda la base de las redes neuronales.

- **Capítulo 3: Metodología**

Se dedica este tercer capítulo a presentar la metodología de trabajo empleada para la creación del algoritmo que se quiere obtener para cumplir con los objetivos del proyecto. En este capítulo se explica la base de datos empleada, el preprocesamiento de datos realizado, se explica el concepto de *overfitting* el cual es bastante temido en el aprendizaje, se muestra el pseudocodigo del trabajo, se realiza una breve descripción de la sintonización de la red y por último se explica el concepto de matriz de confusion.

- **Capítulo 4: Trabajo experimental**

Capítulo dedicado a mostrar los resultados obtenidos aplicando la metodología detallada en el capítulo anterior. Se verá como se ha realizado la parte práctica y los resultados obtenidos de forma global de cada modelo y de forma detallada de cada modo de fallo.

- **Capítulo 5: Estudio económico**

Se dedica este quinto capítulo a proporcionar una valoración económica en la realización del proyecto donde abarcará la planificación, horas empleadas y recursos utilizados.

- **Capítulo 6: Conclusión del proyecto**

Valoración del proyecto y satisfacción de los objetivos marcados. Teniendo en cuenta la técnica de *Machine Learning* utilizada y la detección de los fallos.

- **Capítulo 7: Líneas futuras del proyecto**

Posible continuación de investigación sobre el presente proyecto.

Capítulo 2
ESTADO DEL ARTE

2. ESTADO DEL ARTE

El propósito de este capítulo es el de describir la parte teórica que compone el proyecto, realizando un trabajo de análisis e investigación para una correcta realización de la parte práctica del mismo.

En este capítulo se hará hincapié en el contenido de la detección de fallos, conocida en inglés como *Fault Detection Isolation*(FDI) y las aplicaciones del *Deep Learning*(DL) para la detección y clasificación de estos fallos. Se irá viendo las diferentes técnicas que se pueden usar, asi como los aspectos más importantes a tener en cuenta en la realización de la parte práctica.

2.1. Detección de fallos FDI

En este apartado se va a proceder a explicar en qué consiste la detección de fallos en los que se basa el presente trabajo.

2.1.1 Introducción

Uno de los principales objetivos de cada industria es el de garantizar una producción óptima en base a la calidad, cantidad, plazos, seguridad, etc. Para poder hacer un correcto seguimiento de su producción es necesario la monitorización del proceso para poder verificar que no hay ninguna desviación no deseada que cause alguna pérdida de los objetivos de la industria.

En conclusión es imprescindible contar con un sistema de monitorización que analice el comportamiento de la planta y verifique que trabaja en óptimas condiciones.

Debido a todo esto es necesario realizar una correcta detección de los posibles fallos del proceso [2].

Para todo ello es necesario diferenciar entre; fallo, avería y mal funcionamiento.

- Fallo; desviación no permitida de al menos una característica del sistema respecto al estado usual, aceptable y estándar del mismo.
- Avería; interrupción permanente de la capacidad del sistema para realizar una determinada función según las condiciones de funcionamiento especificadas.
- Mal funcionamiento; irregularidad intermitente en el cumplimiento de las funciones exigidas al sistema

Las etapas de la detección de fallos son [3]:

- Detección de la existencia de fallos
- Identificación del fallo
- Diagnóstico del fallo
- Recuperación del sistema

2.1.2 Métodos FDI

En los métodos de la detección de fallos existen varios tipos de clasificación. En este apartado vamos a ver esa forma de clasificación, centrándonos en la clasificación de métodos de regresión lineal y no lineal. El objetivo de este apartado es conocer las diferentes técnicas de clasificación que existen y situar la técnica de *Deep Leaning*(DL).

Una de las posibles clasificaciones de los métodos de detección de fallos es la siguiente [2].

- Métodos basados en datos. Analizan los datos de la planta, esto no requiere gran conocimiento del ciclo productivo.
- Métodos basados en modelos del proceso. Es necesario un modelo de la planta. Analiza y compara los datos medidos con los estimados en el modelo.
- Métodos basados en el conocimiento. Basados en la experiencia y conocimiento de la planta.

A su vez en la categoría de métodos basados en datos se puede pueden clasificar en los siguientes métodos:

- Métodos estadísticos.
  - *Principle Component Analysis (PCA)*
  - *Independent Component Analysis*
  - *Partial Least Square (PLS)*
  - *Fisher Discriminant Analysis (FDA)*
  - *Quality Trend Analysis (QTA)*
- Método aprendizaje superficial.
  - *Support Vector Machine (SVM)*
  - *Artificial Immne System (AIS)*
  - *K-Nearest Neighbor (KNN)*
  - *Gaussian Mixture Model (GMM)*
  - *Artificial Neural Network (ANN)*
- Método aprendizaje profundo *(Deep learning).*
  - *Restricted Boltzman Machines (RBM)*
  - *Hierarchical Deep Neural Network (HDNN)*
  - *Deep Belief Network (DBN)*
  - *Deep Convolutional Neural Network (DCNN)*

[4]

También podemos realizar una clasificación de los métodos de detección de fallos haciendo referencia a su tipo de regresión, si esta regresión es lineal o no lineal.

A. Métodos de regresión lineal

A.1 Regresión LASSO

Este método de regresión LASSO(*Least Absolute Shrinkage an Selection Operator*) está basado en la aplicación de una penalización a los coeficientes de regresión, haciendo nulos

algunos de ellos. De esta forma lo que se consigue es un modelo con menos variables más sencillo. Solo aquellas variables representativas tendrán valor no nulo [5].

A.2 *Sparse* PCA

El método *Sparse* PCA o también llamado SPCA (*Sparse Principal Component Analysis*) es utilizado con bastante frecuencia en análisis multivariables ya que realiza una reducción de la dimensionalidad de los datos de entrada. Una de las grandes desventajas de este método es que los principales componentes suelen ser combinaciones lineales de los datos de entrada [6].

B. Métodos de regresión no lineal

B.1 Random forest

Este método se trata de árboles predictores, los cuales no están correlacionados. Dichos árboles se van promediando. Cada árbol de este método depende solo de un vector aleatorio que no tiene que ver con los demás árboles, aplicándose la misma distribución en cada uno de los árboles.

Algunas ventajas de este método pueden ser la agilidad con la que se ejecutan grandes bases de datos, tiene gran certeza como clasificador, es posible manejar gran número de variables. No obstante este tipo de método también tiene ciertas desventajas como puede ser su gran sobreajuste y ruido en sistemas de clasificación. También tiene una gran dificultad de interpretación [7].

B.2 Regresion SVM

El método SVM (*Support Vector Machines*) es un algoritmo de aprendizaje supervisado que se usa muy a menudo para la clasificación.

No obstante para el aprendizaje no supervisado esta técnica queda un poco compleja.

Este método lo que realiza es una división espacial a través de un hiperplano o conjunto de hiperplanos que permiten dividir de forma correcta las diferentes clases [8].

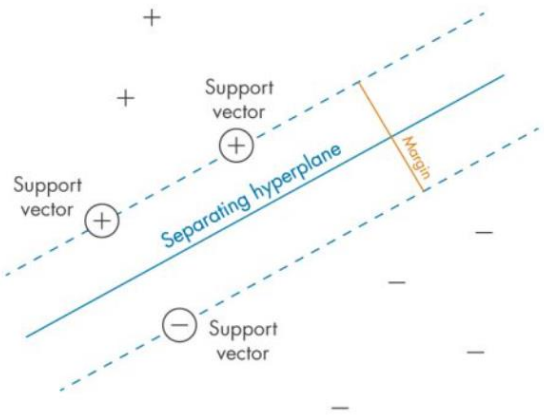


Figura 2.1: Regresión de SVM

B.3 Redes neuronales

Las redes neuronales son un modelo que realiza una combinación de los parámetros de entrada para predecir un cierto resultado. Las redes neuronales lo que realizan es la búsqueda de la mejor combinación para para poder predecir un resultado, a esto se le llama coloquialmente "entrenar" la red neuronal.

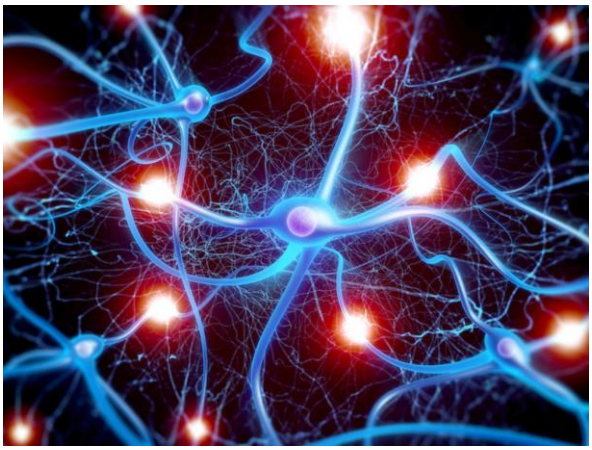


Figura 2.2: Red neuronal

Las redes neuronales son las familias de algoritmos de *Machine Learning* más utilizadas.

En la actualidad el uso de redes neuronales está en pleno desarrollo y auge debido al gran potencial que tienen las mismas.

Algunos ejemplos de uso de las redes neuronales para ver su gran potencial son los siguientes;

- Reconocimiento de caracteres.
- Reconocimiento de imágenes.
- Reconocimiento de voz.
- Predicción bursátil.
- Traducción de idiomas.
- Métodos de clasificación
- Conducción autónoma
- Pronóstico de enfermedades

El nombre de red neuronal viene dado de la cercanía de funcionamiento con las neuronas de las personas [8]

La técnica del *Deep Learnig* está incluida dentro del punto de redes neuronales.

Debido a la importancia que tiene la red neuronal en la realización del trabajo, en bloques posteriores se va a entrar más en detalle en la explicación de este concepto.

2.2. Machine Learning: Aplicación técnica Deep Learning

En primera instancia identificar el ámbito del *Deep Learning*(DL) dentro del *Machine Learning(*ML) también conocido en español como aprendizaje automático.

El *Deep Learning* es un metodo de aprendizaje que está incluido dentro del *Machine Learning*. A su vez el *Machine Learning* es una de las ramas de las que está compuesta la Inteligencia Artificial (IA). De esta manera estos tres conceptos de *Deep Learning, Machine Learning* e Inteligencia Artificial quedarían relacionados como se muestra en la figura XXX.

A continuación se va a entrar brevemente en detalle en cada uno de estos tres conceptos.

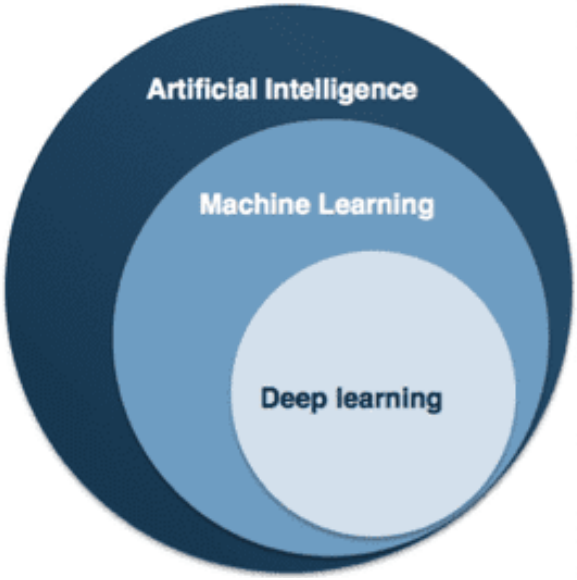


Figura 2.3: Ámbito de la Inteligencia Artificial, Machine Learning y Deep Learning.

2.2.1 Inteligencia Artificial (IA)

El concepto de inteligencia artificial es un concepto bastante amplio y complejo de explicar. Este concepto depende del concepto de inteligencia, el cual, dicho concepto a día de hoy no tiene una definición específica.

Este término nació en el año 1950 cuando una cantidad de pioneros del campo de la informática se realizaron la pregunta de que si un ordenador podía pensar.

Algunas de las definiciones que se han dado sobre este concepto son;

- Esfuerzo por automatizar las tareas intelectuales normalmente realizadas por humanos. [9].
- Capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano [10].

Haciendo un resumen de todas las definiciones encontradas en libros, artículos científicos, etc, se podria hacer una definición de inteligencia artificial como; el objetivo de esta ciencia es la de imitar al ser humano. Algunas de las características del humano para destacar en la inteligencia artificial pueden ser; la comunicación con otras personas, toma de decisiones de acuerdo a experiencias y datos recolectados, aprender de las experiencias y los errores, recordar gran parte de las experiencias vividas, etc. La inteligencia artificial es el conjunto de todas estas técnicas usadas para poder alcanzar la imitación del comportamiento de los seres humanos.

El concepto de inteligencia artificial engloba muchos campos, pudiéndose destacar;

- Visión
- Voz
- Robots
- Sistemas expertos
- Procesado de lenguaje natural
- Aprendizaje automático

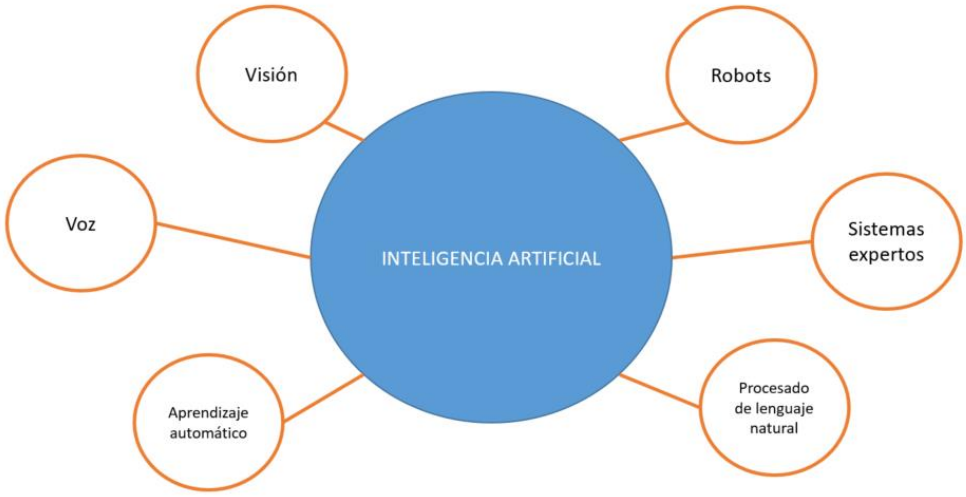


Figura 2.4: Campos de la Inteligencia Artificial

2.2.2 Machine Learning (ML)

El aprendizaje automático es el subconjunto de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan [11].

En este concepto, se usan algoritmos complejos para analizar una cantidad masiva de datos, reconociendo patrones entre los datos y de esta forma poder realizar una predicción sin necesidad de que se programen instrucciones específicas. Realiza estructuras estadísticas que eventualmente el sistema permite que elabore reglas para automatizar las tareas.

El aprendizaje automático empezó a florecer en la época de 1990, convirtiéndose rápidamente en el campo más popular y exitoso de la inteligencia artificial.

Para entender bien este concepto es necesario primero saber diferencias entre *Machine Learning y Deep Learning*. Para ello es necesario entender que es lo que hacen los algoritmos del *Machine Learning*. El *Machine Learning* lo que realiza es descubrir reglas para ejecutar tareas de procesamiento de datos. Para la ejecución del *Machine Learning* son necesario tres cosas;

- Input data points; Serán los datos de entrada en los que se quiere que el programa trabaje.
- Ejemplos del resultado esperado; Se tiene que introducir unos ejemplos del resultado que estamos esperando.
- Forma de medir si el algoritmo de la maquina está realizando bien su trabajo; Este punto es importante para determinar las distancias entre la salida actual del algoritmo y su salida, La medida de esta distancia es utilizada como una señal de retroalimentación para poder ir ajustando la forma en el que el algoritmo funciona. Este último punto es lo que se llama aprendizaje.

El algoritmo de aprendizaje automático consiste en encontrar automáticamente las trasformaciones convenientes que convierten los datos de entrada en representaciones más útiles para la tarea encomendada. Estos algoritmos no son creativos para encontrar estas transformaciones, sino que simplemente buscan en un conjunto predefinido de operaciones el cual se llama “espacio de hipótesis”.

Resumiendo, el *Machine Learning* es la búsqueda de esos útiles de representación de los datos de entrada dentro de un “espacio de posibilidades” ayudándose de la señal de retroalimentación.

El *Machine Learning* es todo el conjunto de métodos estadísticos que permiten que las maquinas mejoren las experiencias por sí mismos. El aprendizaje automático está estrechamente relacionado con las estadísticas matemáticas, no obstante se diferencia de estas en que el aprendizaje automático suele tratar con conjuntos de datos grandes y complejos en los que las estadísticas matemáticas serian poco prácticas [9].

2.2.3 Deep Learning

El *Deep Learning* lleva el concepto de la inteligencia aún más lejos. Este está formado por redes neuronales artificiales (ANN) capaces de adaptarse a nuevos datos, imitando de esta forma al ser humano y la conectividad de su cerebro, clasificando conjuntos de datos y buscan correlaciones entre ellos, obteniendo con ello un aprendizaje mucho más “profundo”. Esto lo realiza con las redes de aprendizaje que a menudo implica un gran número de capas, todas ellas jerarquizadas, aprendiendo automáticamente de la exposición de los datos de entrenamiento.

De esta forma, a través de las conexiones el *Deep Learning* adquiere un conocimiento que puede aplicar a otro conjunto de datos. De esta manera cada vez que se vayan introduciendo más datos, la maquina ira adquiriendo un mayor conocimiento y será más precisa en sus predicciones. Este aprendizaje es un marco matemático para aprender representaciones a partir de una base de datos.

Esta tecnología está teniendo un gran auge en estos últimos años debido al requerimiento de un gran número elevado de datos para realizar un correcto aprendizaje que esto se transmite a su vez en un requisito de tecnología de alto nivel para poder procesar los complejos algoritmos, es por este motivo que hasta estos últimos años el *Deep Learning* no había sufrido este gran desarrollo que tiene en la actualidad.

El *Deep Learning* es el conjunto que hace posible el cálculo de la red neuronal multicapa.

Funcionamiento del Deep Learning.

Como se ha explicado anteriormente, este modelo está formado por una red neuronal multicapa, la tarea que se realiza en cada capa con los datos de entrada se almacena en pesos de la capa. La transformación realizada en cada capa es parametrizada por sus pesos. De esta forma la aprende. Aprender significa encontrar el conjunto de valores para los pesos de todas las capas de la red, mapeando correctamente las entradas de ejemplos a sus objetivos asociado.

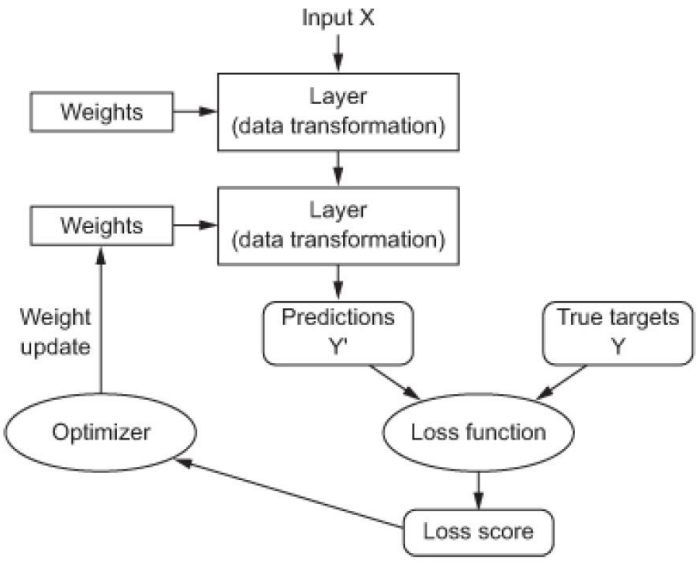


Figura 2.5: Sinóptico de funcionamiento del aprendizaje profundo [9].

La salida de la red neuronal es controlada y medida, comparando la lejanía al resultado que se esperaba. Esta medición la realiza la función de pérdida, también llamada función objeto. Estas funciones cogen la salida de red y el verdadero valor que debería dar y lo comparan, dando un valor de pérdida y calculando que tal bien lo ha hecho la red. El objetivo de esta función de perdida es retroalimentar la red para ir modificando los pesos de la red y de esta manera ir optimizándola según va aprendiendo. Este tipo de ajuste se realiza a través del algoritmo de retropropagación, que se realiza de atrás para adelante, es decir empieza ajustando primero las últimas capas de la red.

Inicialmente los pesos de la red se dan de manera aleatoria, pero a medida que se va entrenando la red con la introducción de datos de entrenamiento estos pesos se van ajustando y optimizando para obtener la mínima perdida lo que proporciona la máxima certeza en la predicción. A esto se le denomina aprendizaje profundo.

Algunas de las aplicaciones del *Deep Learning* son;

- Reconocimiento de caracteres.
- Reconocimiento de imágenes.
- Reconocimiento de voz.
- Predicción bursátil.
- Traducción de idiomas.
- Métodos de clasificación
- Conducción autónoma
- Pronóstico de enfermedades

Pasado-Futuro del Deep Learning.

En la actualidad el *Deep Learning* está en pleno desarrollo y se está haciendo mucho uso de él, como podemos ver en las aplicaciones de uso que se han citado anteriormente. Aunque se anuncia un prometer futuro a corto plazo, lleno de esperanza, hay que ser prudente. En el pasado ya sucedió dos veces, en el cual la inteligencia artificial sufrió un ciclo de optimismo pero que finalmente no acabo de despuntar. En 1960 comenzó este primer ciclo de optimismo y muchos expertos pensaban que el uso de la inteligencia artificial en la vida cotidiana estaba a la vuelta de la esquina. Años más tarde cuando estas altas expectativas no lograron materializarse, los fondos de los gobiernos y los investigadores fueron apartados. Aquí se marca el inicio del invierno de la inteligencia artificial.

No obstante este optimismo fracasado no fue el único. En la década de 1980 otra nueva visión de la inteligencia y optimismo por la misma surgió. Las empresas rápidamente invirtieron en estos sistemas, pero en la década de 1990 estas empresas vieron lo costoso que era mantener un desarrollo de este tipo y el alcance que tenía, por lo que fue disminuyendo el interés, esperando el segundo invierno de la inteligencia artificial.

En la actualidad estamos presenciando un claro desarrollo de esta tecnología y también podemos ver su resultado en todas las aplicaciones en las que se usa. No obstante se debe de ser cauteloso en este optimismo ya que podría presentarse un tercer ciclo similar a los dos anteriores.

La imagen a largo plazo del aprendizaje profundo es brillante. En estos últimos cinco años, este modelo se ha desarrollado bastante, por lo que nos permite dar una idea del rumbo que seguirá en el futuro. Aunque hay bastantes aplicaciones en el aprendizaje profundo, por el momento no se ha utilizado el gran potencial que tiene. La inteligencia artificial tendrá una transición que pasará a ser el elemento central en la forma en que trabajemos, pensemos y vivamos. Esta estará presente en todas las industrias y también será un asistente personal en la vida de los humanos [9].

2.2.4 Aprendizaje supervisado y no supervisado

Dentro del aprendizaje automático es necesario entender y diferenciar que tipos de aprendizaje podemos utilizar. Para ello diferenciamos dos tipos de aprendizaje; supervisado y no supervisado.

El aprendizaje supervisado es el metodo más común, ya que es el más efectivo. En este tipo de aprendizaje el programa enseña al algoritmo las conclusiones a las que debe de llegar, esto quiere decir que la salida del algoritmo ya es conocida. Esto requiere que los resultados del algoritmo ya sean conocidos y estén etiquetados, conociendo su salida. Para ello el algoritmo tiene que ser entrenado en un conjunto de datos que ya están previamente etiquetados, esto quiere decir que tendrá dos entradas, una primera con los datos y una segunda con las etiquetas que indican que tipo de dato se introduce. El aprendizaje no supervisado es usado pero por lo general es menos común que el anterior. En este tipo de aprendizaje no existe un conjunto de datos de entrenamiento y las salidas son desconocidas, no se dispone de datos de referencia. Solo se trabaja con el algoritmo y los datos de entrada, en este tipo de aprendizaje no se

introducen las etiquetas como en el anterior. Dicho algoritmo ira asociando las características de los datos y haciendo distinciones por el mismo, sabiendo que hay diferencias, pero sin conocer la salida [9].

En la gráfica 2.6 de este apartado se puede ver un ejemplo del funcionamiento de estos tipos de aprendizaje. Como se puede ver en la gráfica, los dos aprendizajes parten de la misma base de datos. En el aprendizaje supervisado, la introducción de datos es la base de datos y las etiquetas, es decir, cada dato que se introduce en el código se introduce con una etiqueta. Por ejemplo si se introduce el dato de "gato" se introduce una etiqueta para que el código sepa que ese dato es gato. De esta forma el programa clasifica y sabe cuál debe de ser la salida, el sistema en este caso conoce todas las salidas.

A diferencia con el aprendizaje supervisado, el aprendizaje no supervisado, la introducción de datos se realiza sin etiquetas, esto quiere decir que en el modelo se le introduce un dato por ejemplo "gato", pero no sabe el programa que es "gato". El algoritmo del programa lo que realiza es una clasificación y la salida determina que son diferentes y conoce que hay diferentes grupos pero no sabe que el grupo de "gatos" son "gatos".

En la parte práctica del proyecto se ha empleado un aprendizaje supervisado introduciendo una entrada con las etiquetas de los tipos de modos de funcionamiento de la base de datos.

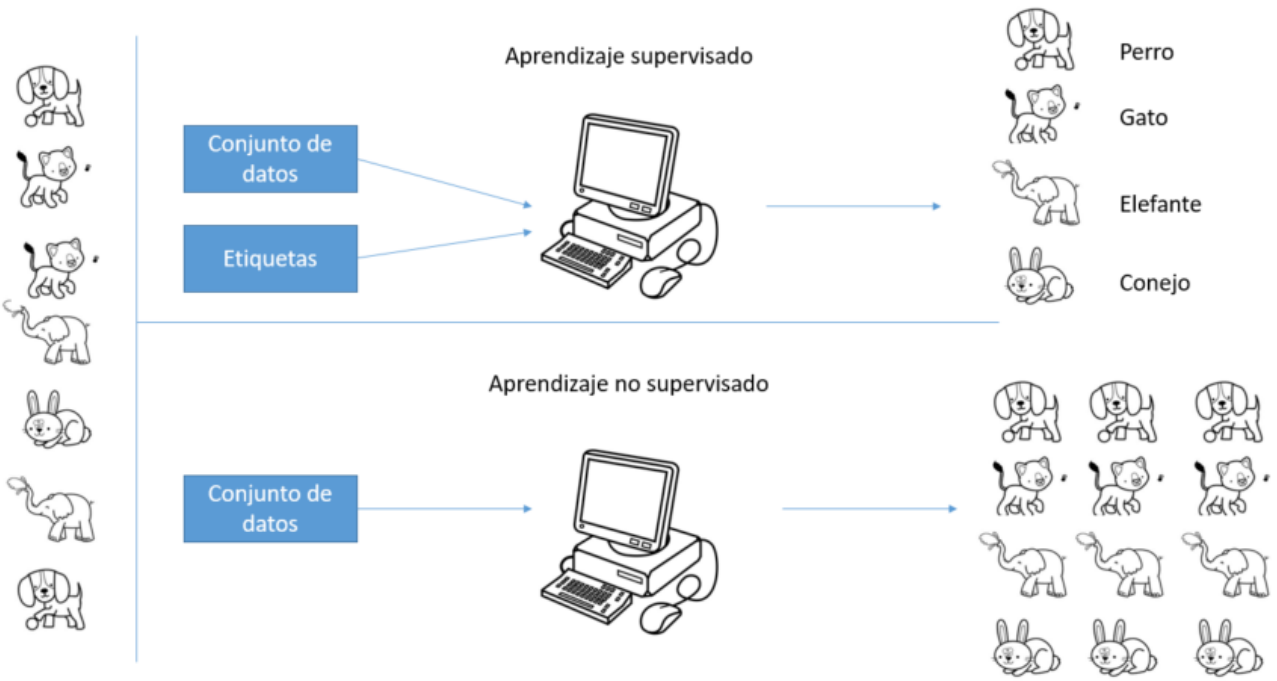


Figura 2.6: Detalle explicativo funcionamiento aprendizaje supervisado y no supervisado

Cada tipo de aprendizaje estará destinado a resolver los siguientes problemas [12];

- Aprendizaje supervisado
  - -Clasificación
  - -Regresión
- Aprendizaje no supervisado
  - -Clustering
  - -Reducción de dimensionalidad

A continuación se muestran gráficamente la clasificación de estos dos métodos de aprendizaje con los algoritmos más usados. Indicar que el *Deep Learning* estaría incluido dentro del sistema de clasificación de la red neuronal [13].

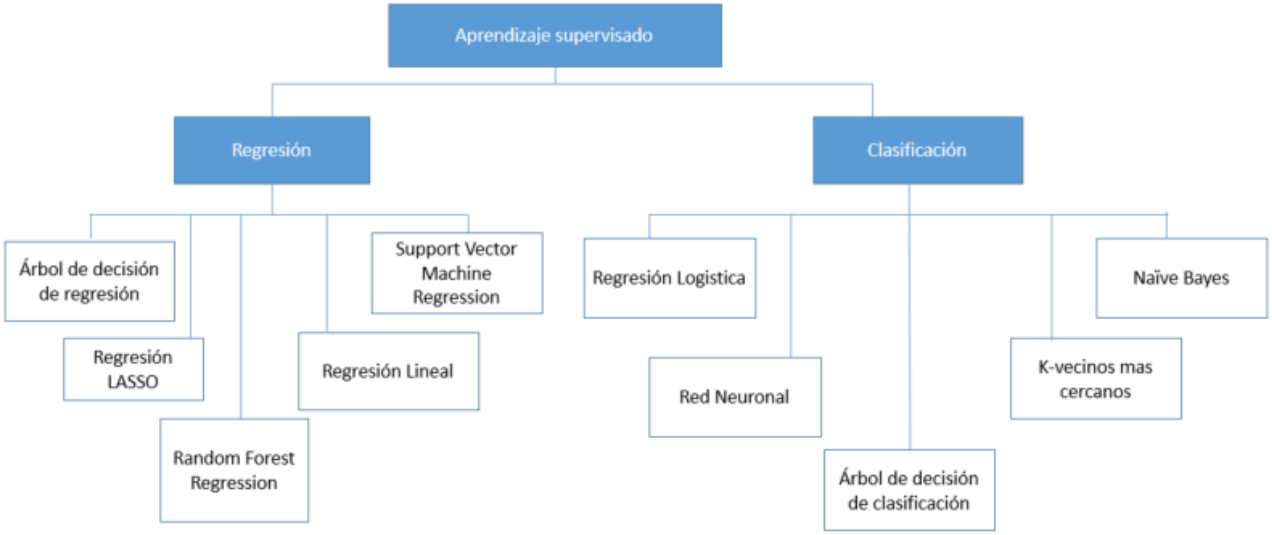


Figura 2.7: Algoritmos usados en el aprendizaje supervisado

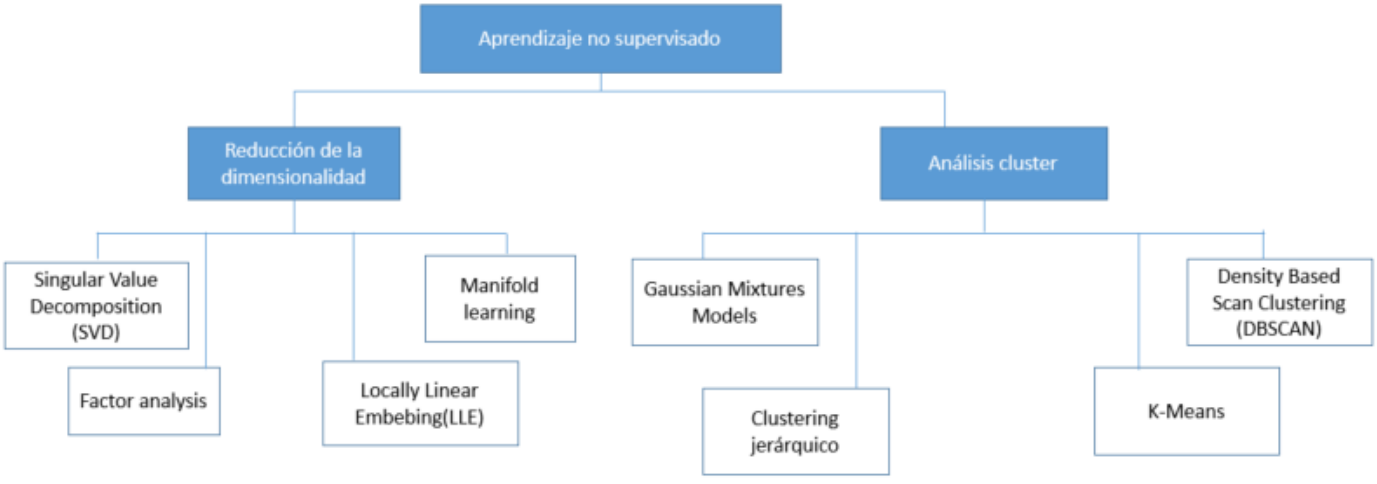


Figura 2.8: Algoritmos usados en el aprendizaje no supervisado

2.2.5 Red neuronal

Debido al gran impacto que tiene la red neuronal en la realización de este trabajo, se ha dedicado esta sección a la misma, profundizando más en su estudio.

El fundamento básico del *Deep Learning* son las redes neuronales artificiales (ANN).

Anteriormente a 1986 se trabajaba solamente con una única neurona conocida como perceptron. Se creía que este algoritmo resolvería gran partes de los problemas expuestos pero pronto se dieron cuenta de las grandes limitaciones que tenía. Una única neurona solo es capaz de resolver problemas lineales. Este tipos de limitaciones vienen recogidas en el libro perceptrons de Marvin Minsky y Seymour A Papert [12]. Posteriormente, vistas las limitaciones del uso de una única neurona, se descubrió la gran capacidad que tenía usar una red neuronal con varias neuronas [12].

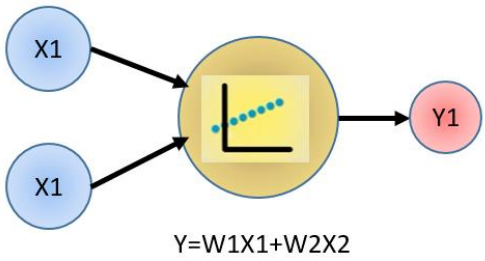


Figura 2.9: Perceptrón. Neurona

La neurona es la unidad básica de procesamiento dentro de la red neuronal. Esta neurona tiene unos valores de entrada los cuales procesarán y dará unos valores de salida. Este procesamiento interno de la neurona lo que realiza es una suma ponderada y regresión de los valores de entrada.

Para entender esto, se puede poner el ejemplo de las puertas AND, OR y XOR a través de la neurona.

Puerta AND

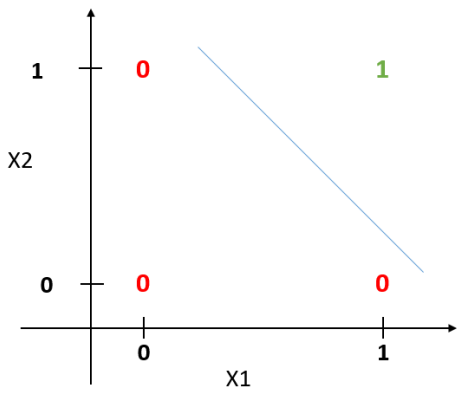


Figura 2.10: Regresión lineal de puerta AND

Puerta OR

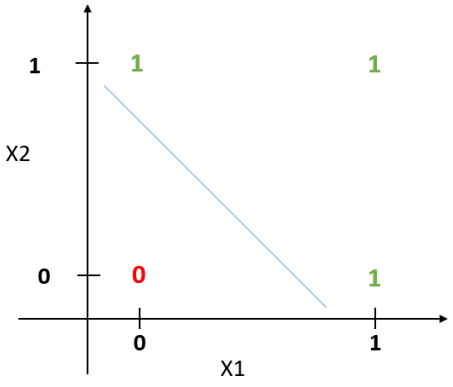


Figura 2.11: Regresión lineal de puerta OR

Se puede ver como con una regresión lineal dada por una única neurona podemos establecer las condiciones de las puertas AND y OR. En cambio el problema viene con la puerta XOR, la cual sería necesaria dos redes neuronales para cumplir con su condicional.

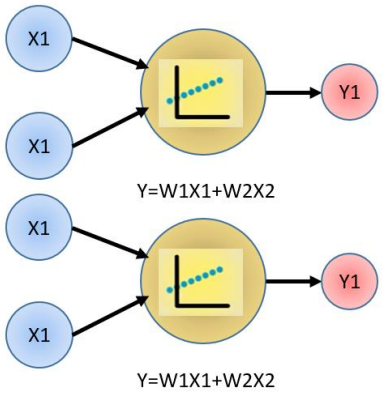


Figura 2.12: Dos neuronas para realización puerta XOR

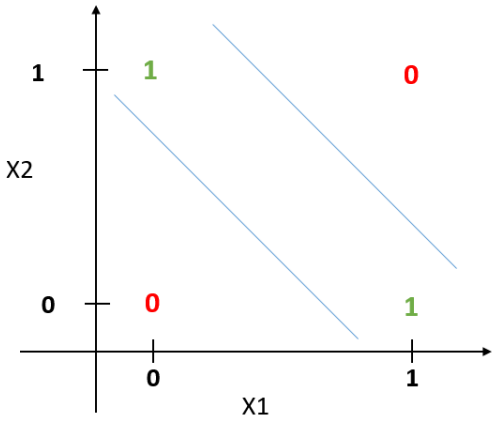


Figura 2.12: Doble regresión lineal para realización de puerta XOR

En este ejemplo hemos visto la necesidad de usar más de una neurona para temas complejos, esto uso de múltiples neuronas es lo que forma el concepto de la red neuronal.

La red neuronal es un conjunto de neuronas conectadas entre sí. Esto nos permite realizar un aprendizaje profundo más conocido como *Deep Learning*.

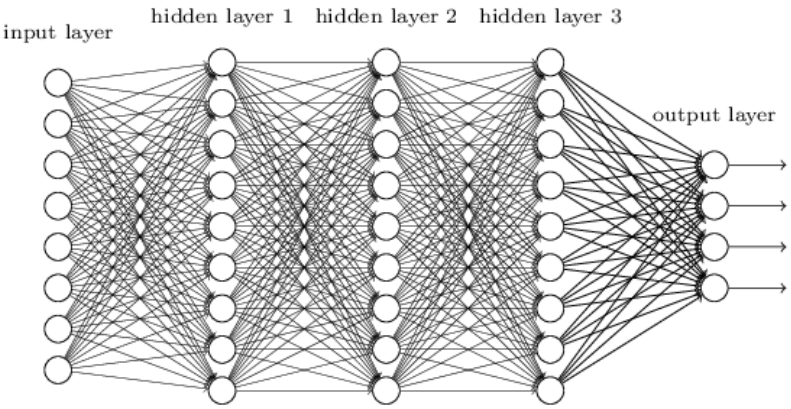


Figura 2.13: Red neuronal

Estas neuronas dentro de la red neuronal vienen organizadas por las diferentes capas que hay;

- Capa de entrada; son las neuronas que se encuentran al inicio y donde se introducen los datos de entrada.
- Capas ocultas; Estas capas son todas las capas intermedias que existen en la red y las cuales reciben datos de las capas de neuronas anteriores.
- Capa de salida. Se trata de la última capa de neuronas y son las que dan la información de salida.

Es importante introducir los conceptos de función de activación y *backpropagation*, ya que son elementos claves en la red neuronal.

Función de activación;

Para realizar una encadenación perfecta entre las diferentes capas de la red neuronal existen las funciones de activación. El objetivo de esta función de activación es la eliminación de la linealidad, permitiendo encadenar la computación de las neuronas.

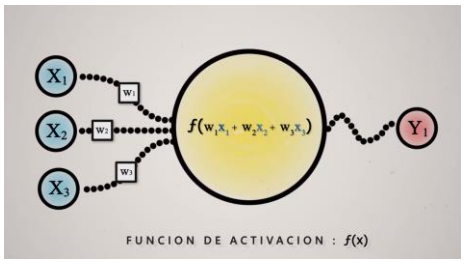


Figura 2.14: Función de activación en red neuronal

Los tipos de función de activación son los siguientes;

- Escalonada

Para un valor de entrada mayor que el umbral, la salida será 1 y para un valor menor que el del umbral, la salida será 0. El nombre de escalonada lo recibe de que el cambio de valor es instantáneo y no de forma gradual, produciendo un escalón. Este tipo de cambio brusco no favorece el aprendizaje por lo que este tipo de función no se suele usar para el *Deep Learning*.

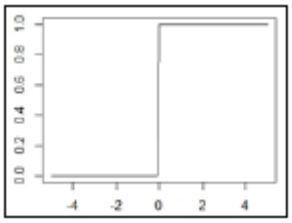


Figura 2.15: Función de activación escalonada

$$f(x) = \begin{cases} 0, & for\ x < 0 \\ 1, & for\ x \geq 0 \end{cases}$$

- Sigmoide

Esta función lo que realiza es que los valores muy grandes se saturen en 1 y los valores muy pequeños se saturen en 0. Esta función nos sirve para representar probabilidades.

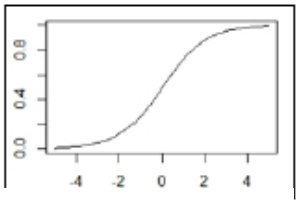


Figura 2.16: Función de activación sigmoid

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

- TANH

Muy similar a la función Sigmoide citada anteriormente pero a diferencia que el rango varia de (-1,1).

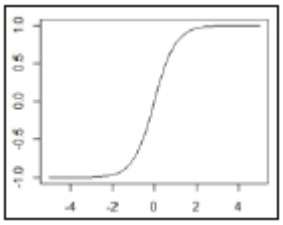


Figura 2.17: Función de activación TANH

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

- RELU

Esta función de activación es la unidad de rectificada lineal. Se comporta como una función lineal cuando es positiva y constante a cero cuando es negativa.

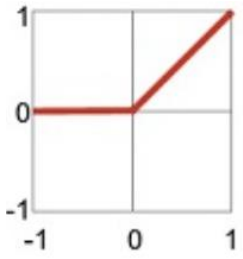


Figura 2.18: Función de activación RELU

$$f(x) = \begin{cases} 0, & for\ x < 0 \\ x, & for\ x \geq 0 \end{cases}$$

- Softmax

Esta función suele ser usada en redes neuronales para la clasificación de datos. Lo que realiza es dar un porcentaje de que probabilidad es de cada tipo de clasificación el dato a validar. La activación de esta función suele ser en la última capa de la red neuronal.

$$f(x) = \frac{e^{Yj}}{\sum_{k=1}^{K}(e^{Yk})}$$

Y-.Salida capas ocultas

k-.Nº clases en el modelo

[9]

Algoritmo de backpropagation

Como ya se comentó anteriormente, es la red neuronal la que aprende sola a través de los datos que se le van introduciendo, realizando un aprendizaje automático. Para poder entender este tipo de aprendizaje que realizan las redes neuronales, es necesario conocer el concepto del algoritmo de *backpropagation*.

Este algoritmo de aprendizaje autoajusta los pesos de la red de atrás hacia delante para de esta forma ir aprendiendo. La función principal de este algoritmo es analizar el error en la salida y de este punto ir retrocediendo hacia atrás para ir analizando los errores e ir viendo que neuronas han repercutido más en este tipo de error obtenido en la salida. Una vez obtenida esta evaluación se sabrá cuanto hay que modificar cada parámetro en dichas neuronas. Esto se ira realizando capa tras capa hasta llegar a la capa inicial. Esta forma de operar del algoritmo hace que se trate de un algoritmo bastante eficiente[15].

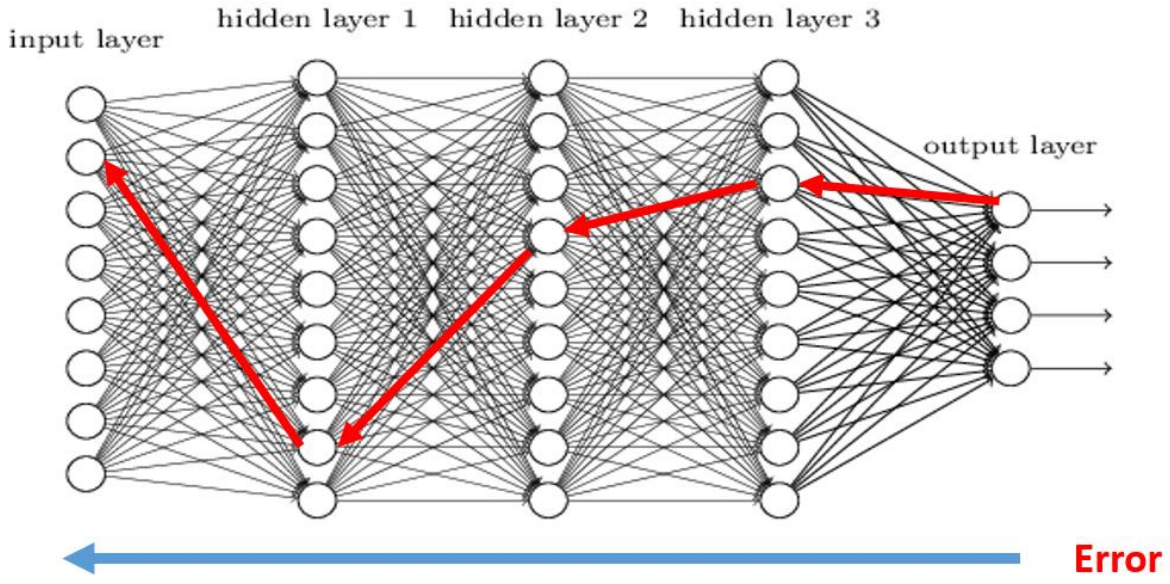


Figura 2.19: Algoritmo Backpropagation

Clasificación de redes neuronales

Aunque hay infinidad de estructuras de redes neuronales (ANN) a usar, las más importantes y con mayor éxito en la detección de fallos son las siguientes.

- DNN: *Deep Neural Network*

Esta es la red neuronal profunda, es una red versátil ya que se puede procesar texto, imágenes, datos numéricos. La estructura de estas redes está compuesta por capa de entrada, capa de salida y capas intermedias, también llamadas capas ocultas. Lo que hace que esta red sea profunda es que en la capa oculta vamos a tener varias capas. Se trata de redes neuronales con al menos 3 o 4 capas ocultas, aunque algunos expertos consideran como ejemplo que una red neuronal de 10 capas es poco profunda. En estas capas las conexiones entre capas es bastante elevado y para datos de entrada grande se vuelve muy pesado computacionalmente.

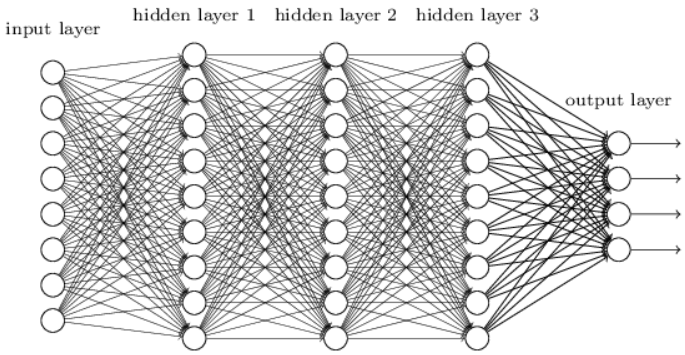


Figura 2.20: Red neuronal

- CNN: *Convolutional Neural Net*

Esta es una red neuronal convolucional. Su uso más común es con base de datos de texto e imágenes. Esta red también está compuesta por los tres tipos de capas citados anteriormente, capa de entrada, salida y oculta. Las capas ocultas son las capas que van a realizar convoluciones y *max pooling*. Más tarde en este proyecto se explicara esta técnica de convoluciones y *max pooling*. Esta técnica lo que realizara es una reducción de los datos de entrada según va avanzando por la estructura de nuestra red. Cada capa va reduciendo los datos, identifica los elementos y características más importantes.

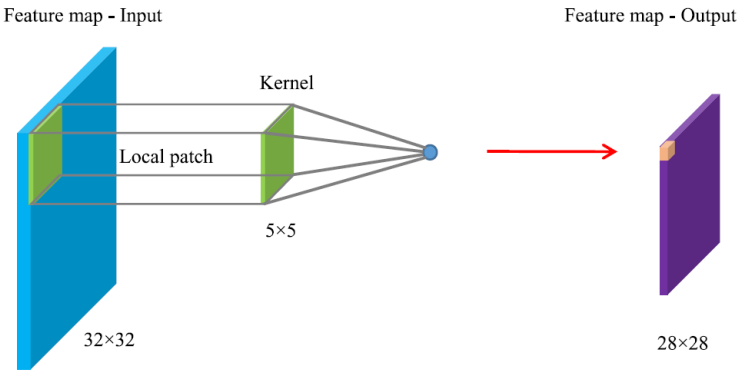


Figura 2.21: Funcionamiento red CNN

En la ilustración de este apartado se puede ver la explicación gráfica del funcionamiento de las redes convulacionales. La salida se compone de mapas de características dentro de las cuales cada unidad está conectada a un parche local en el mapa de entrada a través de un filtro compuesto por un conjunto de ponderaciones.

En conclusión lo que realiza las redes convulacionales es una filtración y simplificación a través de sus capas, creando patrones y obteniendo las características de los datos más representativas.

- RNN: *Recurrent Neural Net*

Estos tipos de redes, llamados redes neuronales recurrentes. Este tipo de red es utilizado para tipos de datos secuenciales. El valor de estos datos depende de los datos anteriores. Este tipo de estructura está formado por los tres tipos de capas citadas con anterioridad: Capa de entrada, salida y oculta. En estas redes, las capas ocultas reciben un dato que genera una predicción y la salida de la capa oculta se alimenta de nuevo, esto permite que la red tenga conocimiento de lo sucedido con anterioridad.

- DCNN: *Deep Convolutional Neuronal Network.*

Este tipo de redes es utilizado en la actualidad sobre todo para la detección de fallos debido a su gran éxito en los modelos experimentados. Esta estructura emplea una técnica conjunta de las dos estructuras explicadas anteriormente, DNN y CNN. La técnica DCNN tiene grandes ventajas ya que incluye la extracción y clasificación de las características de los datos. La extracción de las características se realiza a través de la técnica de la convolucion, mientras que la clasificación de los mismos se realiza a través de la técnica de fully conected, capas completamente conectadas que se explicara posteriormente en este proyecto. En estos tipos de capas se requiere de datos etiquetados en la fase de entrenamiento, lo que se consideró anteriormente como aprendizaje supervisado. Este algoritmo hace uso del concepto de retropropagacion [4].

Tipos de capas de una red neuronal DCNN.

Las redes neuronales (ANN) tienen gran tipo de capas a usar, en la técnica del *Deep Convolutional Neural Network*(DCNN) las capas más usadas son;

A. Capa convulacional

En esta capa se realiza la convolución de los datos de entrada, generando patrones y filtrando los datos de entrada, simplificando y quedándose con las características importantes de los datos según se va avanzando en la red. Este tipo de capa se ha explicado detalladamente con anterioridad.

Lo que realiza es que cada convolución va filtrando el mapa de datos inicial y lo va dividiendo en información más pequeña adquiriendo los patrones y características más relevantes de los datos, de esta forma se simplifica los datos iniciales introducidos en la red.

Las características más importantes en estas capas son;

- Tamaño de filtro
- Profundidad de la capa convulacional.
  Números de filtros que se le va a poner a la capa convulacional
- Stride.

Esta característica será el paso, es decir, como queremos que vaya avanzando el parche sobre nuestros datos. Cuanto más alto sea el paso, más se va a reducir los datos a la salida de la capa.

- Activación de funciones.
  Como se ha visto también anteriormente en la primera parte de este proyecto, para la realización de una perfecta encadenación entre las diferentes capas de la red neuronal y poder estandarizar la salida de los datos, las funciones de activación juegan un papel muy importante en el rendimiento del modelo.

B. Capa pooling

Esta capa *pooling* o también llamada en español agrupación, lo que realiza es una reducción mayor de los datos ya que produce versiones reducidas de los mapas de características de entrada. El objetivo de esta capa es fusionar datos locales similares en uno. Esto se realiza por dos razones, para quitar el número de conexiones, haciendo más ágil la red y otra evita el llamado y temido sobreajuste u *overfitting* que se explicará en el capítulo de metodología.

Existen por lo general dos tipos de capas principales de *pooling* que son;

- MaxPooling
  Lo que realiza es pasar el filtro de cierta altura y cierta longitud por la base de datos. En este esté filtro coge el valor más alto y lo pasa a una matriz más pequeña.

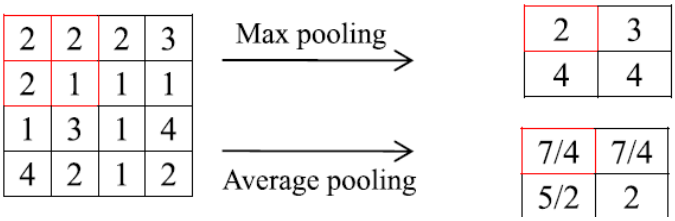


Figura 2.22: Técnicas de capa pooling

- Average pooling
  Procede de la misma forma que en el anterior, lo único que en vez de coger el máximo valor realiza la media de los valores.

En esta capa al igual que la anterior, se tendrá dos parámetros importantes;

- Tamaño del filtro
  Tamaño del parche del mapa que cogerá la parte de los datos que se establezca.
- Stride
  Es el tamaño del paso por el cual el filtro va avanzando a lo largo de los datos. En la ilustración se puede ver un *stride* de 2.

En la ilustración de este apartado se puede ver la forma de proceder de cada tipo.

En conclusión la capa de convolución se encarga de coger los datos e irlos recorriendo con los filtros para generar unos datos más pequeños en longitud y altura, pero más profunda. Después de la convolución se va a tener un max *pooling*, que se va a encargar de agrupar los datos en longitud y altura establecida.

Por este motivo las pacas de convolución seguidas de las capas de agrupación realizan una combinación óptima para el tratamiento de los datos y asi poder realizar una correcta clasificación de los datos, siendo este último el objetivo de este trabajo.

C. Capa dropout

Este tipo de capa es utilizado para combatir el temido sobre ajuste de la red neuronal, lo que nos ocasionara que la red aprenda de una manera errónea y no sea eficaz para el propósito de clasificación por el cual ha sido diseñado.

Lo que realiza esta capa en su funcionamiento es la desactivación de un número de neuronas de forma aleatoria, lo que obliga a las neuronas cercanas a no depender tanto de las neuronas desactivadas. De esta manera las neuronas de la red aprenden a trabajar de manera solitaria y no depender tanto de las neuronas vecinas.

D. Capa fully connected

La entrada de datos en esta capa debe de ser de un vector en una sola dimensión, es por eso que anteriormente a esta capa debe de haber una capa flatten para la preparación de los datos de entrada en la capa de fully connected.

El objetivo de esta capa es la de clasificar las características extraídas de la base de datos. La última salida de la red tendrá que tener el mismo tamaño que de clasificaciones se tengan, es decir, si se tienen en la clasificación 10 tipos de funcionamiento, el tamaño de la última capa tendrá que ser de 10.

E. Capa flatten

La función de la capa flatten es la reducción en una dimensionalidad de los datos de entrada. Como se ha comentado, la entrada de los datos en la capa fully conected debe de ser de una dimensión, es por ello que es necesario hacer uso de la capa flatten con anterioridad de la capa fully connected.

F. Función softmax

Esto es una función que se activará en la última capa de fully conected. Como se explicó anteriormente, esta función es utilizada en los modelos de clasificación de datos. Lo que realiza es ponderar y ver la probabilidad que tiene el dato de ser de un tipo de la clasificación establecida.

Capítulo 3
METODOLOGÍA

3 METODOLOGÍA

En este bloque denominado metodología, se ha puesto en práctica todo aquello que se ha explicado los capítulos anteriores. Este bloque forma parte de la parte práctica del proyecto.

Para el objetivo de este trabajo se ha hecho uso de una red neuronal convolucional profunda, también llamada en ingles *Deep Convolutional Neural Network(*DCNN) debido al gran éxito que tienen para la identificación de fallos en procesos. También dejar indicado que se tratará de un aprendizaje supervisado.

Se recuerda brevemente los objetivos prácticos de este proyecto;

- Objetivo principal: Detección automática del fallo. Saber cuándo se está trabajando en modo normal o en modo fallo.
- Objetivo secundario: Clasificación del tipo de fallo. De esta forma el operario tendrá una pequeña noción de lo que está pasando en el proceso.

A continuación en los siguientes apartados se va a proceder a explicar la base de datos usada, el pre-procesamiento de datos, punto importante antes de la introducción de los datos en la red neuronal, también se entrará en el detalle del temido concepto de *overfitting*, concepto que hay que tener muy en cuenta en el aprendizaje, posteriormente se verá el pseudocodigo utilizado que recoge un breve resumen del código creado, a continuación se entrará en la sintonización de la red para la obtención de los resultados optimizados y por último el concepto de la matriz de confusión.

3.1. Base de datos

Aunque el análisis de la obtención de la base de datos en este trabajo es secundaria, ya que el principal objetivo es el tratamiento de los datos sin entrar en la obtención de los mismos. En este apartado se va a detallar dicha base de datos usada para la realización de este proyecto.

La base de datos ha sido obtenida del *Benchmark Simulation Model No 2* (BSM2) [1], proporcionada por el departamento de Ingeniería de Sistemas y Automática de la Escuela de Ingenieros Industriales de la Universidad de Valladolid.

Se trata de una base de datos de un proceso de una estación de depuración de aguas residuales (EDAR), la cual consta de diez tipos de modo de funcionamiento.

Un primer modo de funcionamiento que corresponde a funcionamiento normal del proceso, sin ningún tipo de fallo. Más tarde se han introducido nueve tipos de funcionamiento que corresponden con nueve tipos de fallos diferentes. A continuación se muestran el modo de funcionamiento y los tipos de fallo;

Tabla 3.1: Tipos de modo de funcionamiento del sistema

Modo	Descripción
0	Funcionamiento Normal
1	Fallo Salk reac - Fallo de alcalinidad en el reactor
2	Fallo Salk influent - Fallo de alcalinidad en el afluente
3	Fallo CO2 - Fallo de Oxigeno
4	Fallo Biomasa influ - Fallo biomasa del afluente
5	Fallo Biomasa reac - Fallo biomasa del reactor
6	Fallo Qr y Qw - Fallo caudales en el circuito de refrigeración
7	Fallo Caudales storage - Fallo en caudales de almacenamiento
8	Fallo Caudales primary - Fallo en caudales primarios
9	Fallo Kla - Fallo en el controlador

Cada entrada de datos cuenta con 141 variables, de las cuales 8 son constantes con valor cero, no teniendose en cuenta debido a que no dan valor añadido en la diferenciación de modos de fallo, sino todo lo contrario, añadiendo ruido al análisis, por estos motivos se ha configurado el programa para eliminar de manera automática toda variable constante igual a cero, optimizando de esta forma el resultado obtenido.

En la ilustración que se muestra a continuación, se puede ver un pequeño croquis del proceso del *Benchmark Simulation Model No 2* (BSM2) [1].

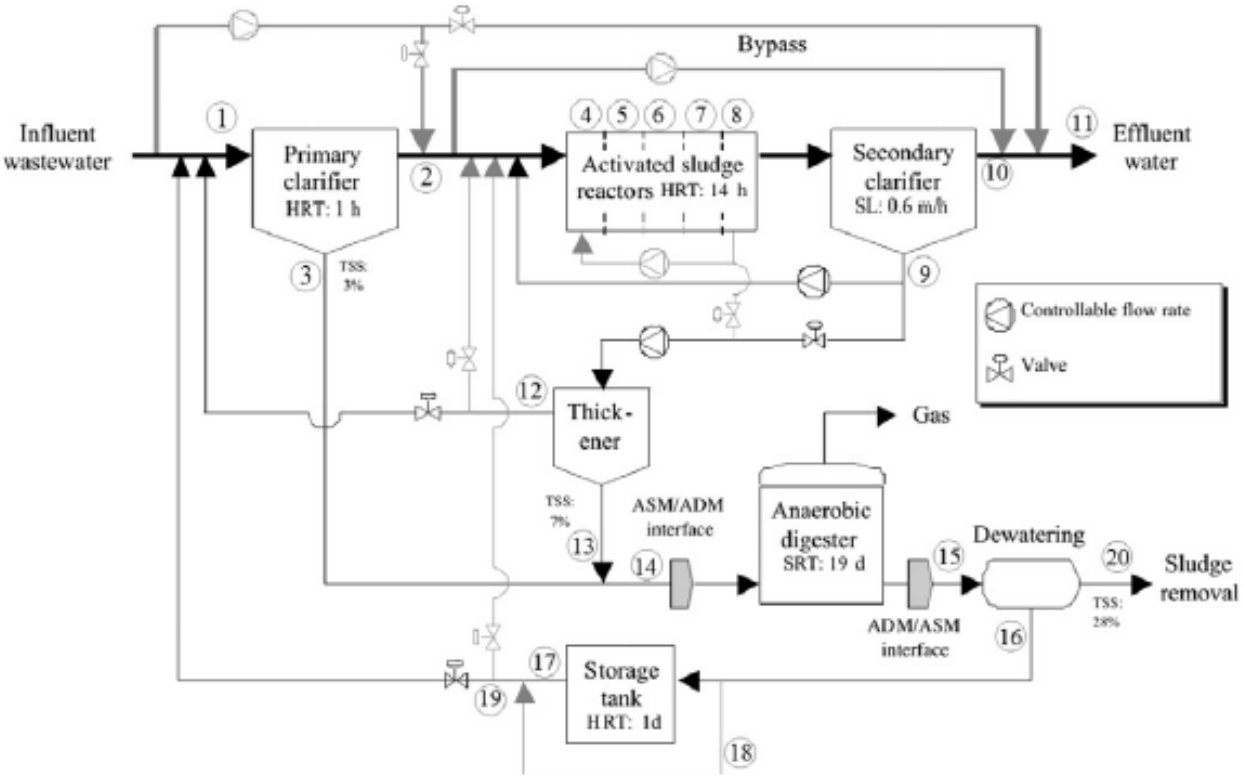


Figura 3.1: Modelo BSM2 [1]

Los datos se han agrupado por lectura de las variables del proceso cada tres dias. La recolección de datos es de cada 15 minutos, es decir, cuatro tomas de datos a la hora, 96 tomas de datos al día, 288 tomas de datos cada tres dias.

Se han introducido un total de 641664 tomas de datos de los diferentes modos de funcionamiento en la base de datos de este trabajo.

3.2. Preprocesamiento de datos

La base de datos juega un punto muy importante, ya que es el punto de partida del el análisis de los datos. El preprocesamiento de datos es el primer paso que hay que realizar, antes de introducir los datos al programa. Un buen preprocesamiento de datos va a jugar una parte estratégica y primordial en los resultados deseados del código. Es por eso que en este punto se debe de gastar gran parte de los recursos.

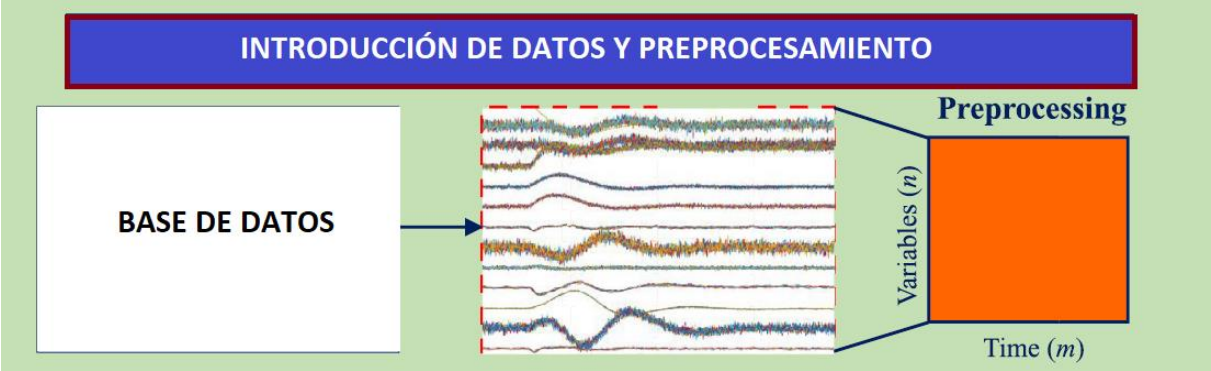


Figura 3.2: Esquema de preprocesamiento de datos.

Primeramente en el preprocesamiento de datos se debe de filtrar y mantener aquellos datos que den valor añadido a la red. En algunos casos las variables con valor cero y las variables con valor constante no dan valor añadido, debiéndose quitar para optimizar el tratamiento de los datos.

También se debe de pensar de qué forma se introducirán los datos a la red neuronal y que información necesitamos de los datos, viendo la dimensión de los tensores a introducir en la red.

3.2.1 Dimensión de datos

Para entender la dimensión de los datos usados, se va a explicar la dimensionalidad, detallada en la ref [9]. Dependiendo del tipo de dato que se trabaje y la informacion que se quiera analizar, la dimensión del mismo cambiará. A continuación se explica la dimensionalidad de los datos.

- Escalar (0 Dimensiones).

  Este tensor contiene solo un número.

- Vector (1 Dimensión)

  Este tensor tiene un solo eje.

- Matrices (2 Dimensiones)

  Este tensor tiene dos ejes.

  La entrada del primer eje son las llamadas filas mientras que las entradas del segundo eje son las llamadas columnas.

- Tensores 3 dimensiones y más dimensiones
  Este tensor tiene tantos ejes como dimensiones se disponga.

Los parámetros de los tensores a tener en cuenta son los siguientes:

- Numero de ejes: Como se ha visto anteriormente, esto marca la dimensión del tensor.
- Forma: Es la dimensión que tiene cada eje.
- Tipo de dato: Puede ser de diferente tipo. float32, uint8, float64.

Algunos ejemplos de datos y la dimensión de sus tensores.

- Datos de vector – Tensor 2D con la forma (muestras, características).
- Series temporales de datos – Tensor 3D con la forma (muestras, tamaño-muestra, características).
- Imágenes – Tensor 4D con la forma (muestras, alto, ancho, canales) o (muestras, canales, alto, ancho).
- Video – Tensor 5D con la forma (muestras, frames, alto, ancho, canales) o (muestras, frames, canales, alto, ancho).

Series temporales:

En este caso se va a explicar los datos de las series temporales de forma más detallada, ya que son los datos usados en la parte experimental de este proyecto.

Como se ha explicado anteriormente la dimensión de los datos que se introducen en las series temporales tienen que ser de una dimensión 3D, ya que se tienen que registras los parámetros más importantes de esta serie de datos que son [9];

- Características (*Features*)
- Muestras (*Samples*)
- Tamaño muestra (*Timesteps*)

Figura 3.3: Tensor 3D en serie temporal [9]

En la metodología de la parte práctica, los datos que se han tratado han sido series temporales con una dimensión 3D.

Explicando los datos que se han trabajado en el proyecto. Los sensores del proceso recogen datos de las variables de entrada. La recogida de datos es de 4 muestras cada hora. El lote se realiza cada tres días, por lo que tendremos cada día 96 muestras por los tres días, un lote de 288 muestras + 1 del título de los datos.

Por otra parte todos los sensores registran 141 variables. Por lo que por cada tres días se registrara como un vector 2D. (289, 141). No obstante hay que introducir también el número de muestras que va a tener la serie temporal, quedándonos un vector 3D. (2228, 289, 141).

3.2.2 Normalización.

Una vez analizando, evaluando y realizando la primera preparación básica de los datos, viene la etapa de normalizarlos o también llamada estandarización.

Esta etapa de normalización es muy importante. Alimentar una red neuronal con datos que toman rangos tremendamente diferentes puede ser bastante problemático. Esta diferencia de rangos en los valores hace que las variables que trabajan en un rango pequeño no sean significativas con las que trabajan en un rango mayor. Es por todo esto que es necesario realizar una normalización a los datos de entrada para que todas las variables trabajen sobre el mismo rango, haciendo más fácil la comparación de caracteres y más ágil el aprendizaje.

Los tipos de estandarización más usados son [13];

- Z-Score

Este método lo que realiza es la transformación uniforme de datos de diferentes magnitudes en una misma, gracias al valor calculado con su fórmula, garantizando la comparabilidad de datos. Para conseguir esto, toma el número de desviaciones típicas con respecto a la media de su población.

Su fórmula es;

$$z = \frac{\mathrm{x} - \mathrm{mean(x)}}{\sigma(x)}$$

z-. Valor normalizado
x-. Valor a normalizar
mean-. Media
σ-. Desviación típica

- MinMax

También es otro metodo bastante usado de normalización. Lo que realiza es un escalado entre [0, 1], usando los valores mínimos y máximos de los datos.

$$z = \frac{\mathrm{x} - \mathrm{min(x)}}{[max(x) - \mathrm{min}(x)]}$$

z-. Valor normalizado
x-. Valor a normalizar
min-. Valor mínimo de la población de datos

- Sigmoide

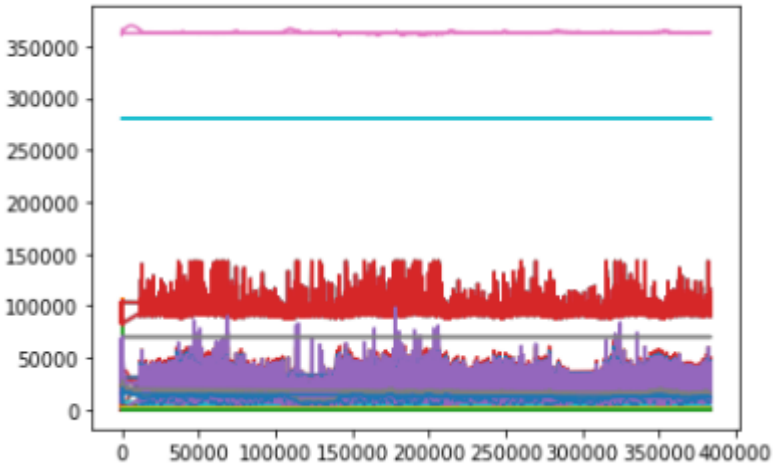
Este modelo al igual que el anterior realiza un escalado entre el rango de [0,1]. Para ello realiza una exponencial de los valores a normalizar.

$$z = \frac{1}{1 + \exp(-x)}$$

z-. Valor normalizado
x-. Valor a normalizar

Se muestra a continuación un ejemplo de la normalización de los datos de entrada con cada método, haciendo uso de la base de datos de la parte experimental. Estas gráficas se han sacado del propio código creado para la realización del proyecto.

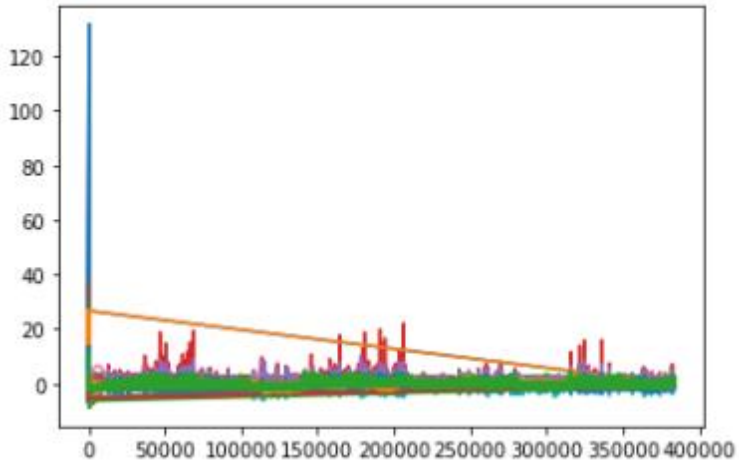
Base de datos **sin normalizar**



El primer caso que se muestra son los parámetros de la base de datos sin normalizar, vemos como el rango de los valores de las variables son muy distintos, haciendo que los grandes valores hagan insignificantes las desviaciones de los pequeños valores, no permitiendo realizar un buen análisis y sacar las características necesarias para realizar una buena clasificación del metodo de funcionamiento.

Figura 3.4: Variables base de datos sin normalizar

Base de datos con normalización **z-scoore**



En este segundo caso, aplicando el método de normalización z-scoore, aunque se ve que se ha mejorado el problema de escala en comparación con la base de datos sin normalizar, este problema persiste en menor rango, mejora la situación pero no es óptimo para los datos de los que se dispone.

Figura 3.5: Variables base de dotas con normalización Z-Score

Base de datos con normalización **MinMax**

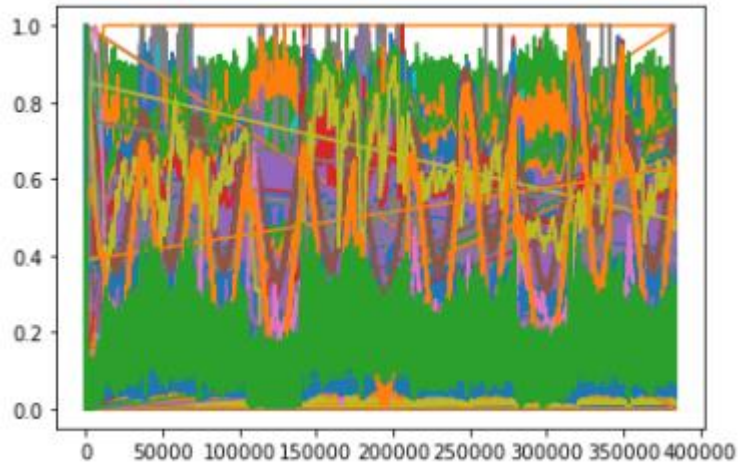


Figura 3.6: Variables base de datos con normalización Min-Max

En este tercer caso, donde aplicamos la normalización MinMax, resolvemos el problema de escala, presentado en los dos casos anteriores, ya que los valores de las variables trabajan en un rango [0, 1]. En este caso cualquier diferencia de valor de cada variable será más significativo, debido a la reducción del rango de escala y permitirá una mejor clasificación del modo de fallo. Todo ello hace que este método sí que se óptimo para la realización de este proyecto.

Base de datos con normalización **Sigmoide**

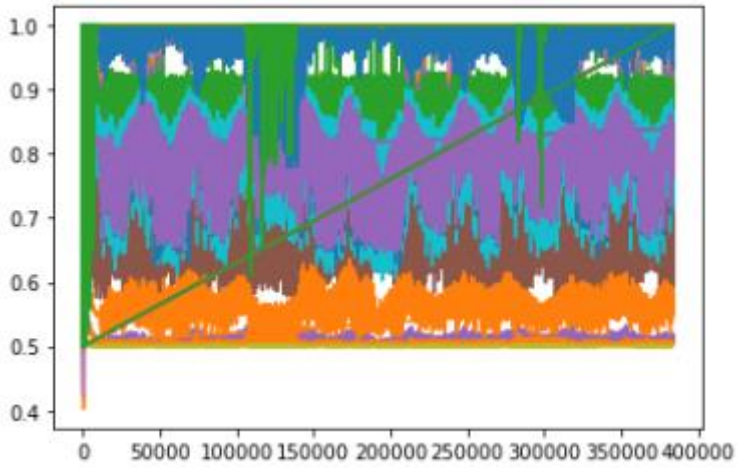


Figura 3.7: Variables base de datos con normalización Sigmoide

En este último caso la normalización es realizada con el método sigmoide, lo que al igual que la normalización MinMax, permite trabajar en un rango de escala [0, 1], permitiendo resolver este problema. Por ello también hace de este metodo, un metodo óptimo de uso. En este caso con la base de datos que disponemos vemos que con la normalización sigmoide se trabaja en un rango inferior de aproximadamente un 0,5, teniendo a parte una mayor homogenización de los valores de las variables, permitiendo esto, hacer más fácil la clasificación de los modos de fallos. Es por ello que el metodo de normalización sigmoide es el ideal para la normalización de este proyecto.

A mayores de lo explicado, se ha realizado verificaciones experimentales en el código con los diferentes métodos de normalización, obteniendo el mejor resultado con la normalización sigmoide.

3.2.3 Cross validation

El funcionamiento de la red neuronal es aprender con una parte de los datos, a este tipo de datos se llamaran datos de entrenamiento y validar ese aprendizaje con otra parte de datos denominados datos de validación.

En una red neuronal es muy importante dejar indicado que cantidad de los datos de entrada van a ser dedicados para entrenar y que cantidad para validar la red. De este punto partirá la eficacia del aprendizaje de la red.

Aunque se puede coger el valor que se quiera, normalmente la proporción óptima suele ser un 80% de los datos para entrenar y el 20% restante para testear la propia red, no obstante dependiendo de las circunstancias se puede coger otros valores.

Para garantizar un correcto reparto de datos de entrenamiento y validación, se usa la técnica del *cross validation* o también llamada validación cruzada en español.

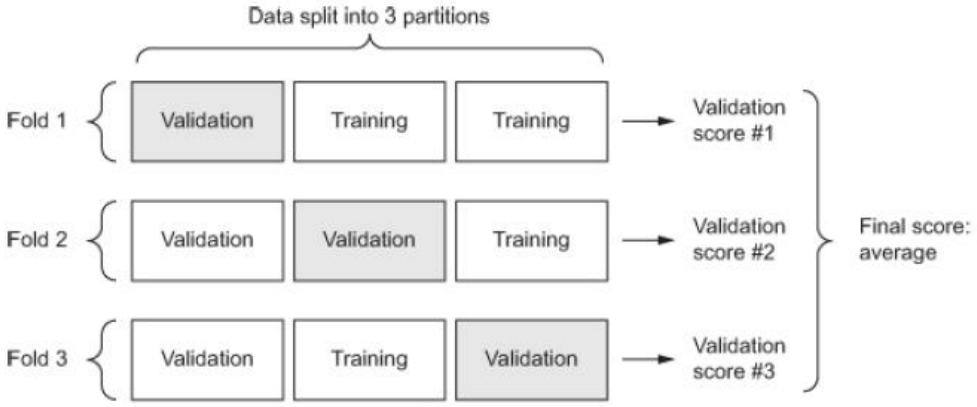


Figura 3.8: Técnica Cross-Validation [9]

Esta técnica divide los datos en varias partes, una parte de testeo y las partes restantes de entrenamiento. Realiza varias iteraciones cambiando la parte de testeo. Una vez finalizada, hace la media aritmética de todas las iteraciones que ha realizado. En la figura de la derecha, se puede ver gráficamente el funcionamiento de esta técnica.

Tipos de validación cruzada [17]:

- K-Iteraciones

Esta validación cruzada llamada K-iteraciones o en ingles *K-Fold* es la validación más usada. Consiste en dividir todos los datos en subconjuntos. Un subconjunto se utiliza como datos de testeo y el resto de subconjuntos como datos de entrenamiento. El proceso es repetido K-iteradas veces con cada uno de subconjuntos de datos, hasta completar todos. Posteriormente se realiza la media aritmética de todas las k-iteraciones realizadas para obtener un único dato. Este método es muy práctico y preciso debido a que se evalúan todas las posibilidades con los datos de entrada.

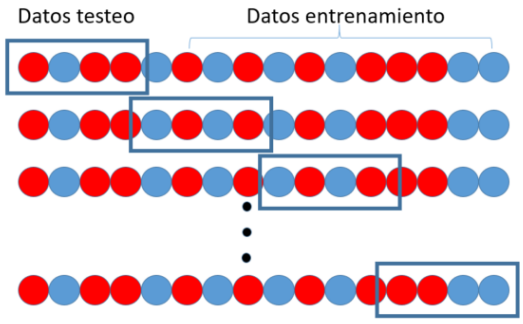


Figura 3.9: Validación cruzada K-Fold

- Aleatoria

En este tipo de validación, el conjunto de entrenamiento y testeo se divide de manera aleatoria. Se realiza varias iteraciones, siendo el resultado final la suma de las medias aritméticas de los valores obtenidos en cada iteración. La ventaja de este método es su sencillez y agilidad, no obstante, la desventaja reside en que puede haber algunos datos que queden sin evaluar y por el contrario, otros datos que se evalúen más de una vez.

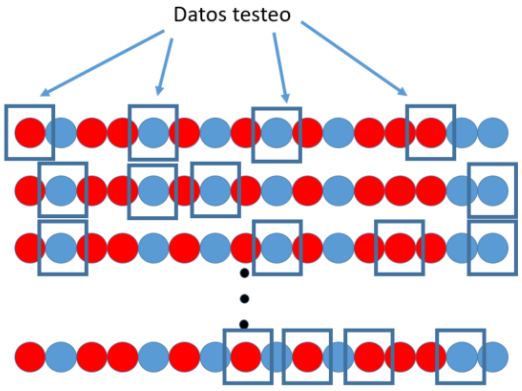


Figura 3.10: Validación cruzada aleatoria

- LOOCV

Este modelo llamado LOOCV, *Leave-One-Out-Cross-Validation*, consiste en que cada iteración que se realiza se tenga una solo muestra de dato de testeo y el resto de datos de entrenamiento. En este tipo de validación cruzada obtenemos un error muy bajo y es muy precisa pero no suele ser usado, ya que es de gran complejidad y exige un número elevado de iteraciones [14].

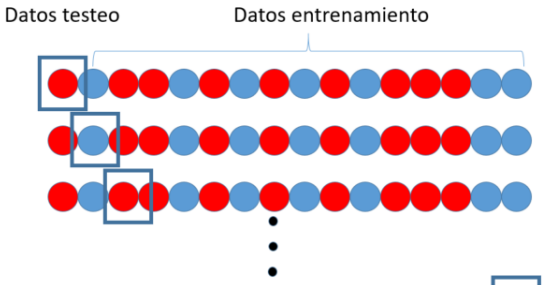


Figura 3.11: Validación cruzada LOOCV

En la parte experimental de este proyecto, se ha dividido la parte de entrenamiento y la parte de testeo por modos de funcionamiento, de esta forma garantizamos una mejor distribución, es decir, si se coge globalmente todos los datos de entrada, podría darse la circunstancia que para testear se cogiesen casualmente solo fallos del tipo uno. Por este motivo se han separado la parte de testeo y la parte de entrenamiento con la técnica de validación cruzada con K-iteraciones modo por modo y posteriormente se han sumado las partes de entrenamiento y testeo de cada modo de funcionamiento, garantizando la homogeneidad de todos los modos de funcionamiento.

3.3. Overfitting

Este término de *overfitting* o también llamado sobreajuste en español es un término muy importante en el ámbito del aprendizaje automático. Este término es una característica a tener en cuenta a la hora de realizar un modelo de *Machine Learning* ya que influirá directamente en el éxito, directamente en la exactitud y rendimiento del modelo.

La red debe de estar ajustada de una manera óptima, sin que este poco ajustada pero tampoco debe de estar sobreajustada ya que esto también ocasionará problemas.

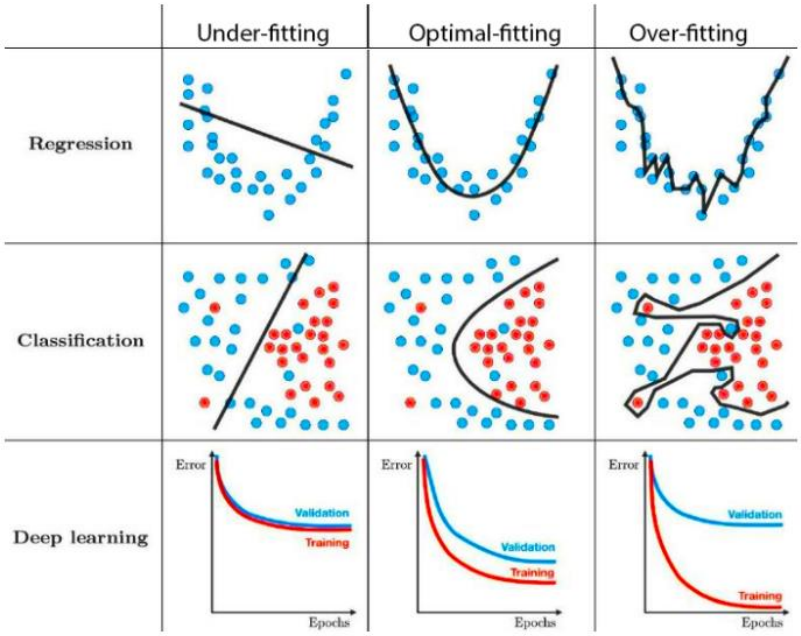


Figura 3.12: Representación ajuste de aprendizaje

En la ilustración podemos ver como una red poco ajustada tiene un error de entrenamiento y de validación elevado. Una red ajustada en condiciones óptimas, el error de validación será el más bajo. En cambio realizando un sobreajuste de la red, el error de entrenamiento se podrá bajar, no obstante al bajar este error, el error de validación irá aumentando. Esto es debido a que aunque se piense que realizar un aprendizaje con los datos de entrenamiento con el mínimo error es lo más óptimo, en verdad no es así, ya que si la red aprende de todos los datos de entrenamiento, estará aprendiendo mal, debido a que también aprende de los datos extraordinarios. Al final lo que se busca es un aprendizaje óptimo intermedio, el objetivo es conseguir que el error de validación sea lo menor posible. En la figura 3.12 vemos como el error de validación más bajo se produce en el *optimal-fitting*.

En general las causas que producen el fenómeno de sobreajuste suelen ser las siguientes;

- Aprendizaje de ruido en el entrenamiento
  Esto ocurre cuando en la etapa del aprendizaje, la base de datos que introducimos es bastante pequeña o con demasiado ruido. La red aprende de igual forma de los datos correctos y de los datos con ruido. Por ese motivo, optimizando el algoritmo de aprendizaje, se puede realizar que dicho algoritmo diferencie de los datos verdaderamente representativos de los datos de ruido.
- Baja cantidad de datos
  Es necesario una cantidad mínima de datos tanto de entrenamiento como de validación. Si no se dispone de datos suficientes el modelo no podrá realizar un aprendizaje óptimo.
- Desequilibrio entre modos
  Debe de estar equilibrado los datos que se introduzcan de cada modo para que se realice correctamente el aprendizaje de cada modo.

- Mala separación de datos de entrenamiento y validación
  Un mal reparto entre los datos de validación y entrenamiento puede producir que se quede algun modo de clasificación sin suficientes datos tanto para validar como para entrenar.
- Estructura de red
  El uso de capas ocultas en exceso también puedo favorecer a la aparición del *overfitting*.

En la parte práctica se han tenido en cuenta estas causas, realizando las siguientes acciones:

- Filtrado de datos de entrada eliminando valores constantes igual a cero. Consiguiendo de esta manera eliminar ruido.
- Realización de *cross-validation* con metodologia de *k-fold* para garantizar una correcta división de datos de entrenamiento y validación.
- División de datos de entrenamiento y validación por modos de fallo. La técnica del *cross-validation* citada en el anterior punto, ha sido aplicada por modos de fallo, con ello se garantiza a mayores un buen reparto de todos los datos por modos de fallo. Una vez obtenido los datos diferenciados por datos de entrenamiento y validación en cada modo, se han agrupado todos los datos de todos los modos para obtener los datos de entrenamiento y validación de la base de datos entera, garantizando un perfecto reparto.
- Capa *dropout*. Se ha hecho uso de esta función en la estructura de la red neuronal para actuar sobre la conexión de las capas. Esta función se verá más adelante.

3.4. Pseudocódigo

El Pseudocódigo es una descripción del código usado en la programación del modelo. Esta descripción es de alto nivel y básica en la cual se deja registrado los pasos a programar y los objetivos que se quieren llegar a obtener. Este pseudocódigo se realiza antes de la programación del código, para tener una noción de que es lo que se quiere realizar y el método de proceder antes de empezar a programar.

El pseudocódigo usado en la parte experimental de este trabajo es el que se detalla a continuación.

Primeramente se realiza un cargado de librerías sobre el código a utilizar. Este paso es importante, ya que un registro incorrecto de las librerías a usar ocasionará una inhabilitación de la función.

Tabla 3.2: Pseudocódigo introducción librerías

Procedimiento 1 Introducción de librerías
Requisitos: Funciones a usar durante la programación // Cargar librerías a usar **Import** librerías requeridas: **From** librería **import** Colección de funciones:

El segundo paso del procedimiento será utilizar una correcta lectura de la base de datos. Es importante tener de antemano una estrategia clara y saber de qué forma se quieren registrar los datos, ya que este punto condicionara los puntos sucesivos.

Primeramente se ha realizado una función de lectura de datos en la cual ha ido clasificando el tipo de funcionamiento leyendo la designación del fichero, en este caso en formato .csv. De esta manera se generara una lista con las etiquetas de la lectura y otra lista con todos los *dataframes* leídos.

A mayores se ha implementado un filtro para un correcto tratamiento de datos. La función de este filtro es la de eliminar aquellas variables que sean cero, eliminando el ruido que pueda producir esta variable en la diferenciación de los modos de fallo. En particular para este proyecto se ha iniciado de una toma de datos de 141 variables, teniendo 8 variables de ellas constantes con valor cero, quedando un total de 133 variables tras la aplicación del filtro explicado anteriormente en este párrafo.

Tabla 3.3: Pseudocódigo lectura datos de entrada

Procedimiento 2 Lectura de datos de entrada

Requisitos: Datos de entrada formato Excel .csv

```
                                        // Estrategia de datos a leer
  def función etiquetas in fichero:
    if nombre_fallo ⊂ nombre_fichero:
        Etiquetar i
    Else:
        Etiquetar -1

  read datos in fichero:
    for i in fichero:
        loc sumacolumnas != 0 in datos_i
                                        //localiza variables diferente a cero
        datostotal = datostotal + datos i

        etiquetadatostotal = función etiquetas
    end for
```

El tercer paso será la realización de un filtro automático que garantice la dimensión de los datos, ya que si hubiese algun dato con diferente dimensión a la considerada, esto ocasionará un fallo en la lectura del programa. La dimensión de cada dataframe ha sido considerada en este proyecto de 289 recogidas de datos por 133 variables.

Tabla 3.4: Pseudocódigo filtro tamaño de datos de entrada

Procedimiento 3 Filtro tamaño de datos de entrada

Requisitos: Datos de entrada filtrados y registrados en un único dataframe

```
                                        // Estrategia de datos a leer
  for dataframel in datostotal:
    if datosframe != tamaño_lotel:
        dataframe save in posición_fuera_rango
    Else:
   end for                              //Se identifican las posiciones que están fuera de rango
```

```
for i in posición_fuera_rango:
    del posición_fuera_rango in datostotal
end for
                                        //Se borra las posiciones que son fuera de rango
                                          identificadas sobre el registro de la lista de dataframes
```

Después de realizar el correcto tratamiento y filtrado de la base de datos otro punto muy importante, como se ha visto anteriormente en este documento, es realizar la normalización de los datos, para tratar todos en la misma escala y asi permitir una correcta clasificación entre los diferentes modos de funcionamiento.

Tabla 3.5: Pseudocódigo normalización de datos

Procedimiento 4 Normalización de datos
Requisitos: Datos de entrada filtrados y registrados en un único dataframe // Estrategia de datos a leer **for** *datostotal*: **algoritmo** normalización //aplicación del algoritmo de normalización que mejor resultado tenga, en el caso del proyecto, algoritmo sigmoide.

Una vez obtenido los datos, filtrados y correctamente normalizados se deben de realizar la técnica del crossvalidation, explicada también anteriormente, para obtener los datos de entrenamiento y los datos de validación. El porcentaje de datos se ha escogido con la proporcionalidad óptima de 80% para datos de entrenamiento y 20% para datos de validación. En este proyecto se ha realizado la técnica del crossvalidation para cada tipo de modo de funcionamiento, sin realizarlo al conjunto entero de datos, es decir, se ha ido modo por modo realizando la partición de datos de entrada y datos de validación, unificándolos posteriormente, de esta forma se ha podido garantizar una correcta repartición y homogenización de todos los datos.

Tabla 3.6: Pseudocódigo Crossvalidation

Procedimiento 5 Crossvalidation
Requisitos: Datos de entrada filtrados y registrados en un único dataframe // Estrategia de datos a leer **for** *i* **in** *numero_modos*: **if** *i* = *modo_funcionamiento* **algoritmo** crossvalidation (0,2 validación, 0,8 entrenamiento). **else** **end for**

Con la realización de esta última etapa se acabaría la parte del preprocesamiento de datos. La siguiente etapa seria la realización de la red neuronal. En esta parte, se genera la estructura de la red neuronal. A mayores se ha incluido un contador de tiempo de procesamiento para realizar una evaluación de tiempos de cada modelo.

Tabla 3.7: Pseudocódigo Red Neuronal

Procedimiento 6 Red Neuronal

Requisitos: Datos de entrada generados en entrenamiento y validación.

```
Generate model:
        imput layer                    //Se establece el tamaño de la capa de entrada
        intermediate layer             //Insertamos las capas intermedias que requiera
                                       la estructura a realizar
        ……
        Output layer                   //la capa de salida tiene que tener la misma
                                       dimensión que clasificaciones se quiera realizar
```

Una vez generado el modelo de estructura de red neuronal a realizar se pasará por un bucle para que corra el programa en todos los parámetros que se desee, de esta manera se encontrará el parámetro óptimo de funcionamiento de la estructura de red neuronal generada. También dentro de este bucle se guardarán los datos generados para el posterior análisis, es importante hacer una buena estrategia de guardado de los datos que se quieran analizar para hacer una correcta explotación de los mismos, se tendrá que tener en cuenta la forma y el formato del guardado de estos datos.

Tabla 3.8: Pseudocódigo bucle procesamiento

Procedimiento 7 Bucle de procesamiento

Requisitos: Modelo de red neuronal creado

```
Generate contador_tiempo:

For i in epochs
    For j in batch_size
        For k in Learning_rate
            Run model in (i,j,k)
            Evaluate model
            Save results (contador_tiempo, matriz_confusión, resultados red,
                          Gráfica_aprendizaje)
        End for
    End for
End for
```

3.5. Sintonización de la red neuronal.

La sintonización de la red neuronal es la acción de modificación de los parámetros internos de la red para de esta forma optimizar el resultado final consiguiendo el valor requerido.

Dicha sintonización se ha realizado tras un estudio profundo de las redes neuronales y de forma experimental.

Parámetros principales de entrada a tener en cuenta para realizar una buena sintonización de la red.

- Estructura de la red neuronal.
  La estructura de la red neuronal puede ser muy diversa. En apartados anteriores se ha hecho una explicación de los posibles tipos de capas a usar. Posteriores se realizará una explicación sobre las diferentes estructuras de red usadas para optimizar los resultados de la mejor forma posible.
- Ratio de aprendizaje.
  También llamado en inglés *learning rate*. Este es un parámetro importante dentro de las redes neuronales, ya que deja indicado el tamaño del camino que tome el algoritmo de optimización. Inicialmente cada neurona tiene un peso arbitrario establecido, posteriormente los datos de entrenamiento van pasando por dichas neuronas hasta llegar a la capa de salida. En la capa de salida la neurona predice la clase a la que pertenecen los datos de entrenamiento. Esta predicción es usada por la función de pérdida para establecer el nivel de acierto de la red neuronal. Este ciclo se repite varias veces actualizando los pesos de las redes neuronales para de esta forma ir optimizando la red neuronal y reduciendo la pérdida. Aquí es donde entra el parámetro de tasa de aprendizaje para la actualización de los pesos. Los gradientes de pérdida se calcularan realizando la relación de la perdida entre el peso de cada neurona multiplicado por la tasa de aprendizaje, como se muestra a continuación.

$$y = \frac{loss}{w}\ x\ learning\ rate$$

y-. Gradiente de pérdida
loss-. Pérdida de la neurona
w-. Peso prestablecido de la neurona

  Haciendo uso de un ratio de aprendizaje bastante elevado tenemos el riesgo de sobre pasar el mínimo de optimización, en cambio haciendo uso de un ratio muy pequeño, la red neuronal tardará más en aprender debido a que la actualización de los pesos de las redes se hace de forma más corta.
- Tamaño de lote.
  Este parámetro también es de importancia para hacer una correcta sintonización. La cantidad total de datos a entrenar se debe de dividir en lotes más pequeños. Estos lotes

dejará indicada la cantidad de muestras que entrena antes de realizar la evaluación de los pesos de la red. Si se escoge un tamaño de lote pequeño, se tendrán muchos lotes pero de poco tamaño, corriendo muy rápido cada lote pero teniendo que realizar el programa muchas actualizaciones de peso, llevando esto también mucho tiempo de procesamiento de entrenamiento. Sin embargo si se escoge un lote muy grande, se tendrá menos actualizaciones de pesos pero tardarán más en procesar el lote debido a la gran cantidad de muestras que tiene el lote. Por otro lado con un lote de muestras muy elevado se tendrá el problema de error de agotamiento de recursos ya que ocupará una enorme cantidad de memoria. Se debe de escoger un tamaño de lote intermedio óptimo. Es por este motivo que en este trabajo se ha corrido el programa con varios tamaños de lote para encontrar el valor óptimo.

- Epochs.
  Llamado también en español épocas o ciclos. En cada ciclo se ejecutan todos los datos de entrenamiento que se disponen y se actualizan los pesos. Los datos aunque estén separados por lotes, se ejecutaran todos los lotes que se tienen.

Todos estos parámetros explicados anteriormente se deben de tener en cuenta para realizar la buena sintonización de la red, que dependerá de la situación y los datos a tratar y de esta forma obtener el mejor resultado posible, optimizando la red al máximo. La forma de sintonizar la red es de forma empírica, que es como se ha procedido para la realización de este proyecto.

3.6. Matriz de confusión

La matriz de confusión es un útil muy utilizado para los problemas de clasificación en el mundo del aprendizaje automatizado, como es el caso del apartado experimental de este trabajo, en el cual se clasifica los diferentes modos de fallo de un proceso. Este tipo de matriz es usado para ver la precisión y sensibilidad del modelo de una forma sencilla e intuitiva. Esta matriz es usado para validar el modelo, hace uso de los datos de validación y de esta manera evalúa el nivel de aprendizaje que ha realizado la red neuronal.

La matriz de confusión dispone de dos dimensiones, una dimensión de predicción y otra de valor actual o real. Los valores que ira adoptando los datos en la tabla serán los siguientes [16];

- Verdadero positivo: Los datos reales son verdaderos y los datos de predicción también son verdaderos.
- Verdadero negativo: Los datos reales son falsos y los datos de predicción también son falsos.
- Falsos positivos: Los datos reales son falso y los datos de predicción son verdaderos. Esto quiere decir que la predicción ha sido errónea, el modelo ha predicho algo diferente al valor real.
- Falsos negativos: Los datos reales son verdaderos y los datos de predicción son falsos. Al igual que la situación anterior, en este caso la predicción ha sido incorrecta, ya que el modelo predice algo diferente al valor real.

Tabla 3.9: Valores de matriz de confusión

	Predicción		
Actual		Positivo	Negativo
	Positivo	Verdadero Positivo	Falso Negativo
	Negativo	Falso Positivo	Verdadero Negativo

Las métricas de la matriz de confusión son las siguientes.

- Exactitud:

Llamada en inglés *accuracy*, hace referencia a lo cerca que esta el resultado del valor verdadero, en resumen, la exactitud es la cantidad de predicciones positivas que fueron correctas.

$$acc = \frac{(VP + VN)}{(VP + FP + FN + VN)}$$

- Precisión:

Hace referencia a la dispersión del modelo. Se calcula realizando la relación de verdaderos positivos entre los resultados de todos los positivos como se muestra en la siguiente ecuación.

$$Precision = \frac{VP}{(VP + FP)}$$

- Sensibilidad

También llamado en inglés *Recall* o *Sensitivity*, es la tasa de Verdaderos Positivos. La tasa de positivos que han sido correctamente identificados por el algoritmo. El cálculo de este valor se muestra en la siguiente ecuación.

$$Recall = \frac{VP}{(VP + FN)}$$

- Especificidad

Conocido en inglés como *Especificity*, es la tasa de Verdaderos Negativos. Este parámetro indica de los casos negativos que el modelo ha clasificado correctamente. Se calcula de la siguiente manera.

$$Especificity = \frac{VN}{(VN + FP)}$$

- Tasa de falsos positivos

Es la probabilidad que tiene el modelo de dar una falsa alarma, de dar por positivo un dato, cuando su verdadero valor es negativo.

$$TFP = \frac{FP}{(FP + VN)}$$

- Tasa de falsos negativos

También llamada tasa de error, nos indica la probabilidad que tiene el modelo de dar por malo un verdadero positivo.

$$TFN = \frac{FN}{(FN + VP)}$$

- F1 Score

A mayores en la matriz de confusión existe otro parámetro muy empleado, ya que resume de manera simplificada la precisión y sensibilidad del modelo. Este parámetro es el llamado F1 Score. Su ecuación es la siguiente:

$$F1\ Score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

VP-. Verdadero Positivo
VN-. Verdadero Negativo
FP-. Falso Positivo
FN-. Falso Negativo

En los modelos se pueden dar varias circunstancias que vienen recogidas en el valor F1 Score.

- Alta precisión y alta sensibilidad: El modelo realizado trabaja de manera correcta.
- Alta precisión y baja sensibilidad: El modelo no detecta satisfactoriamente pero cuando lo hace, la detección es confiable.
- Baja precisión y alta sensibilidad: El modelo detecta bien la clase, pero también en esta detección incluye muestras de otra clase, la detección no es confiable.
- Baja precisión y baja sensibilidad: El modelo no consigue clasificar correctamente las clases, el modelo es poco eficaz.

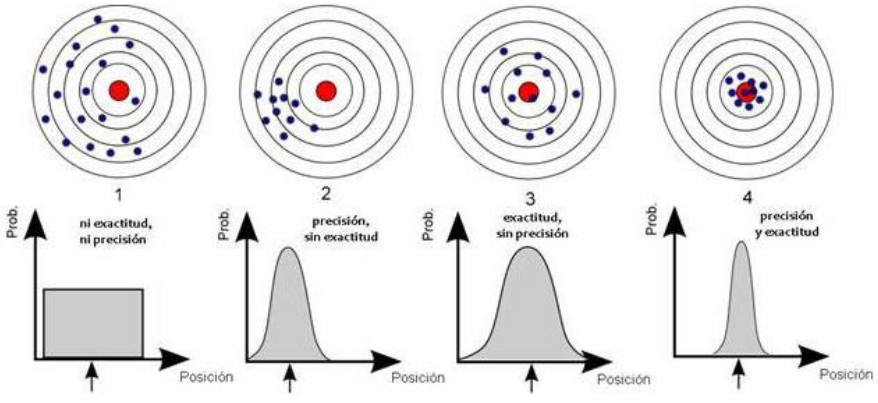


Figura 3.13: Precisión y exactitud

Capítulo 4
TRABAJO EXPERIMENTAL

4. TRABAJO EXPERIMENTAL

En este bloque se va a detallar y analizar los resultados obtenidos de la parte práctica del proyecto, teniendo en cuenta todo lo aprendido en los bloques anteriores.

En primer lugar se va a detallar la estructura de los diferentes modelos de red neuronal utilizados. En segundo lugar se entrará en la sintonización de la red, observando cómo se ha procedido y que rango de parámetros se han usado. Posteriormente se realizará una evaluación de resultados obtenidos para cada modelo a modo global, para entrar después en el detalle del análisis de los resultados de cada modo de funcionamiento en cada modelo.

4.1. Estructura red neuronal

En este trabajo se ha realizado un estudio detallado de la estructura de la red neuronal a usar. Para ello se han realizado nueve tipos de modelos con diferentes estructuras, para evaluar y comparar los resultados. La dimensión de la red neuronal será 2D.

A continuación se detalla las estructuras que se han analizado experimentalmente.

Tabla 4.1: Estructura de red neuronal de los diferentes modelos

Modelo	Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8	Capa 9	Capa 10
1	Conv2D (128filtros)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)			
2	Conv2D (128filtros)	Conv2D(128filtros)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)		
3	Conv2D (64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)				
4	Conv2D (64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)		
5	Conv2D (64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	MaxPooling2D(2,1)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)		
6	Conv2D (64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	MaxPooling2D(1,2)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)		
7	Conv2D (64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)		
8	Conv2D (64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)	
9	Conv2D (64filtros)	Conv2D(64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,1)	FullyConnected(300)	Dropout(0,2)	FullyConnected(10)

Nota; la dimensión del kernel de todas las capas convolucionales serán de (3x3)

Se puede observar como los modelos que se han generado, sus estructuras varían entre 6 a 10 capas. En dichas estructuras se han escogido el tipo de capas necesarias para la aplicación

del *Deep Convolutional Neural Network* (DCNN), donde incluye capas convolucionales, capas *pooling*, capas *fully connected*, capas *flatten* y una capa con parámetro *dropout*.

4.2. Ratio de aprendizaje, tamaño de lote y epochs

Como se ha citado anteriormente, estos tres parámetros de ratio de aprendizaje, tamaño de lote y epochs, se tratan de variables de entrada a modificar para ir sintonizando la red neuronal en busca del resultado más óptimo.

Es por ello que se han ejecutado todos los modelos con varios bucles para ir cambiando de esta forma los parámetros de entrada; ratio de aprendizaje, tamaño de lote y numero de epochs, consiguiendo de esta forma realizar un barrido de todos los parámetros, obteniendo el resultado más óptimo de cada modelo. Los valores de los parámetros a los que se han realizado los bucles son;

Tabla 4.2: Parámetros de sintonización de red

Parámetros	Valores
Epochs	50, 100, 150, 200, 250
Batch size	32, 64, 96, 128, 160, 192
Learning rate	0.001, 0.0001

Esto quiere decir que cada modelo se ha ejecutado un total de 60 veces, con cada parámetro de la tabla citada anteriormente. Se han realizado 9 modelos a 60 ejecuciones, teniendo un total de 540 ejecuciones del programa.

A continuación en la siguiente tabla se puede ver para que nº de *epochs*, *Learning rate* y *batch* se ha alcanzado la máxima exactitud de validación para cada uno de los modelos.

Tabla 4.3: Valores para la máxima exactitud de validación

Modelo	Exactitud de validación	Epochs	Learning rate	Batch size
1	0,890	100	0,0001	128
2	0,596	50	0,001	32
3	0,915	150	0,001	96
4	0,897	100	0,001	32
5	0,879	50	0,0001	64
6	0,897	100	0,0001	96
7	0,899	100	0,0001	64
8	0,899	50	0,001	96
9	0,922	150	0,0001	96

4.3. Evaluación de modelos

Como citado en los apartados anteriores, el tratamiento de este trabajo ha sido orientado a la obtención del mayor éxito para los objetivos estipulados. Haciendo un breve recordatorio de los objetivos prácticos del proyecto.

- Objetivo principal: Detección automática del fallo. Saber cuándo se está trabajando en modo normal o en modo fallo.
- Objetivo secundario: Clasificación del tipo de fallo. De esta forma el operario tendrá una pequeña noción de lo que está pasando en el proceso.

Para la evaluación de los modelos, se han ejecutado todos ellos con los parámetros indicados, realizando una comparativa entre ellos.

A continuación se muestra la satisfacción de ambos objetivos.

Tabla 4.4: Valor de satisfacción de objetivos prácticos del proyecto

Modelo	Objetivo 1: Detección del fallo	Objetivo 2; Clasificación del fallo
1	99,33%	89,01%
2	59,64%	59,64%
3	99,55%	91,48%
4	100,00%	89,69%
5	99,10%	87,89%
6	99,78%	89,69%
7	99,10%	89,91%
8	99,78%	89,91%
9	99,78%	92,15%

Se aprecia que en el modelo 4 satisface correctamente el objetivo principal de detección de fallo, teniendo una exactitud del 100%, mientras que el modelo 9 es el modelo que mejor satisface el segundo objetivo de clasificación del tipo de fallo.

Los parámetros más importantes para realizar la evaluación de la red neuronal serán:

- Exactitud de validación
  La certeza de validación nos dará el nivel de acierto que tiene la red con las predicciones realizadas. Esto se realizará con los datos de validación que se han reservado para validar el modelo. Este es uno de los parámetros más importantes del modelo.
- Perdida de validación
  La pérdida de validación nos muestra el valor de la función de pérdida, es decir, el error que se produce entre el valor real y la predicción realizada.

- Tiempo
  El valor del parámetro del tiempo dejará indicado el tiempo que tarda la red en procesar el modelo, interesando una red dinámica y ligera con el menor tiempo de procesamiento sin sacrificar el resultado de certeza de validación.

En la tabla de abajo se puede ver los parámetros de cada tipo de modelo para realizar una correcta valoración de cada modelo.

Tabla 4.5: Valores de cada modelo

Parámetros	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6	Modelo 7	Modelo 8	Modelo 9	Mejor modelo
Exactitud validación	0,890	0,596	0,915	0,897	0,879	0,897	0,899	0,899	0,922	9
Perdida de validación	0,012	0,073	0,010	0,010	0,015	0,011	0,010	0,010	0,010	3,4,7,8,9
Exactitud entrenamiento	0,895	0,596	0,915	0,896	0,872	0,900	0,888	0,890	0,904	3
Perdida de entrenamiento	0,010	0,073	0,009	0,010	0,014	0,010	0,011	0,011	0,010	3
Tiempo	42410,44	30445,24	7250,00	13945,54	5832,49	11323,68	9600,81	8532,48	37639,37	1
Precisión	0,7	0,06	0,8	0,71	0,72	0,72	0,73	0,73	0,83	9
Precisión ponderada	0,86	0,36	0,91	0,87	0,87	0,87	0,87	0,87	0,92	9
Sensibilidad	0,76	0,1	0,81	0,77	0,73	0,77	0,78	0,78	0,82	9
Sensibilidad ponderada	0,89	0,6	0,91	0,9	0,88	0,9	0,9	0,9	0,92	9
F1 score	0,71	0,07	0,79	0,72	0,7	0,73	0,74	0,73	0,81	9
F1 score ponderado	0,87	0,45	0,9	0,88	0,86	0,88	0,88	0,88	0,91	9
Epochs	100	50	150	100	50	100	100	50	150	-
Learning rate	0,0001	0,001	0,001	0,001	0,0001	0,0001	0,0001	0,001	0,0001	-
Batch	128	32	96	32	64	96	64	96	96	-

Marcado en verde sobre la tabla, muestra el mejor resultado de cada parámetro. En la última columna podemos ver qué modelo satisface mejor cada parámetro. Se observa que el modelo 9 tiene mejor exactitud de validación, mayor precisión y sensibilidad. No obstante el modelo 3 tiene mejor exactitud de entrenamiento, es decir, ha realizado un aprendizaje mayor, pero la validación de este modelo 3 ha sido inferior al modelo 9. Esto quiere decir que a veces aunque se tenga una exactitud de entrenamiento muy buena, esto puede provocar un sobre aprendizaje (*Overfitting*), visto en el punto 3.3, haciendo que la exactitud de validación baje.

En la gráfica que se muestra a continuación, se puede ver la evolución de la exactitud de validación de cada modelo, mostrada en porcentaje.

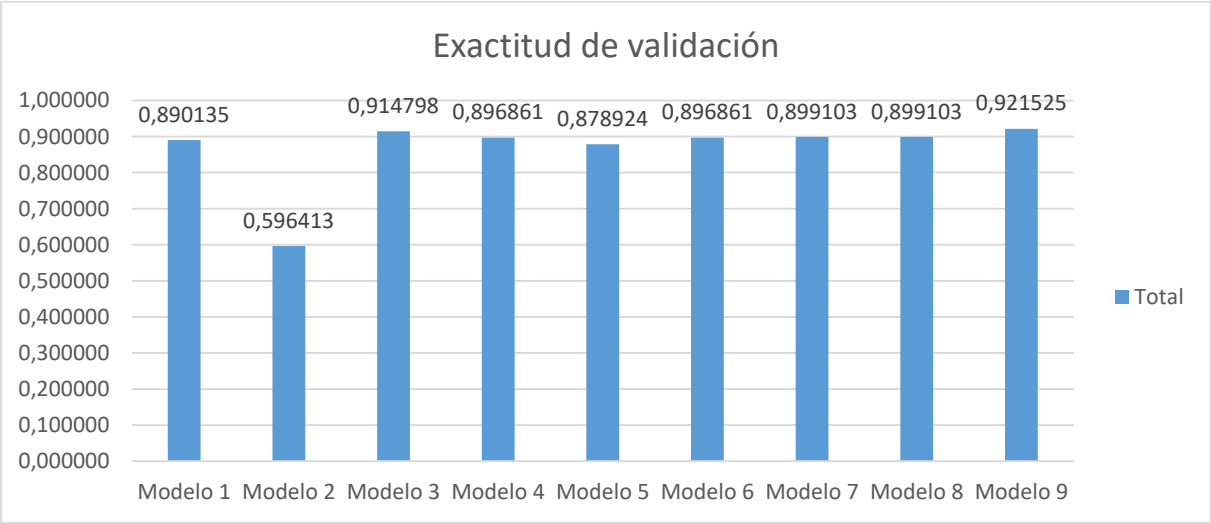


Figura 4.1: Evolución de exactitud de validación por modelos

Como se puede observar se ha obtenido una exactitud máxima de validación del 92%, en el modelo 9, precisión más que aceptable y bastante por encima del valor deseado.

No obstante cabe destacar el modelo 3 el cual tiene también una exactitud de test elevada y cercana de 91,5%, un poco menor que el modelo 9, pero en comparación al tiempo de ejecución, el modelo 3 dispone de un tiempo de ejecución mucho menor que el modelo 9.

Se observa el modelo 2 con un valor bajo del 59%. Esto pone de manifiesto que el aprendizaje de dicho modelo no ha sido realizado en condiciones óptimas, sufriendo de sobreajuste en su aprendizaje y saturando la red. Esto es debido a su estructura, en ocasiones tampoco es óptimo añadir muchas capas, ya que puede dar en una saturación de la red. En este caso en el modelo 2, se han añadido cuatro capas convulucionales con 128 filtros cada una, generando la saturación.

A continuacion tambien vamos a analizar el valor de la funcion de pérdida que ha dado la red. En la siguiente grafica podemos ver la comparacion de estas pérdidas en los diferentes modelos analizados.

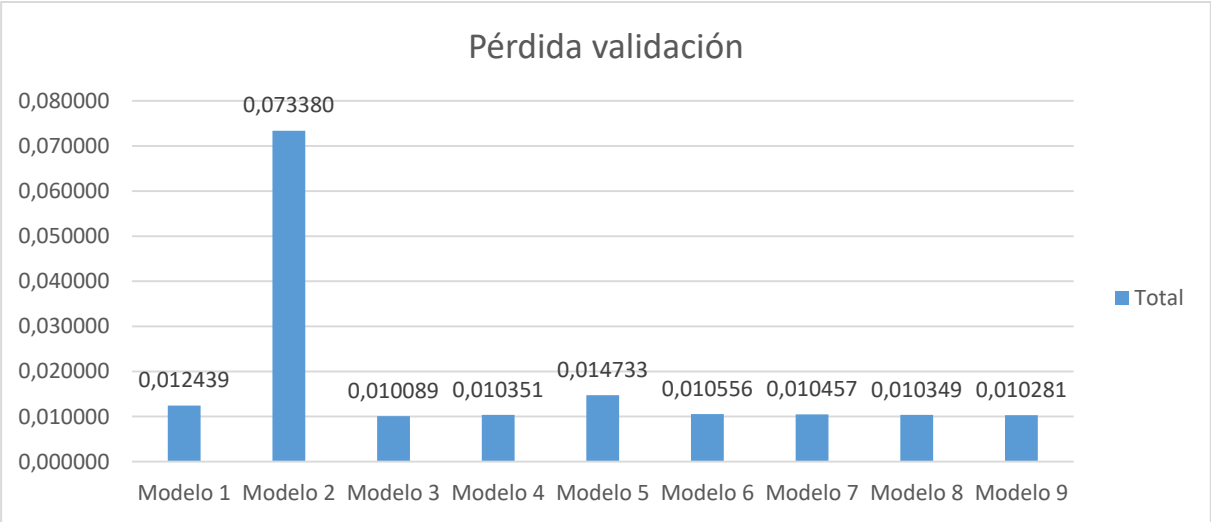


Figura 4.2: Evolución del valor de pérdida por modelos

Este valor esta inversamente relacionado con la exactitud del validación, es decir, una exactitud de validación grande, proporcionará una perdida baja y viceversa.

Al igual que comentado en el apartado de exactitud de validación, el modelo 2 se ve como tiene una perdida elevada, haciendo que la red no aprenda en condiciones óptimas. En cambio en el resto de los modelos es banstante similar, destacando el modelo 3 que es un poco mejor que el resto.

En relacion al tiempo de procesamiento. Este valor es de importancia ya que muestra la agilidad que tiene el algoritmo a la hora de realizar su ejecución. En la gráfica siguiente se puede ver el tiempo de ejecución de los modelos.

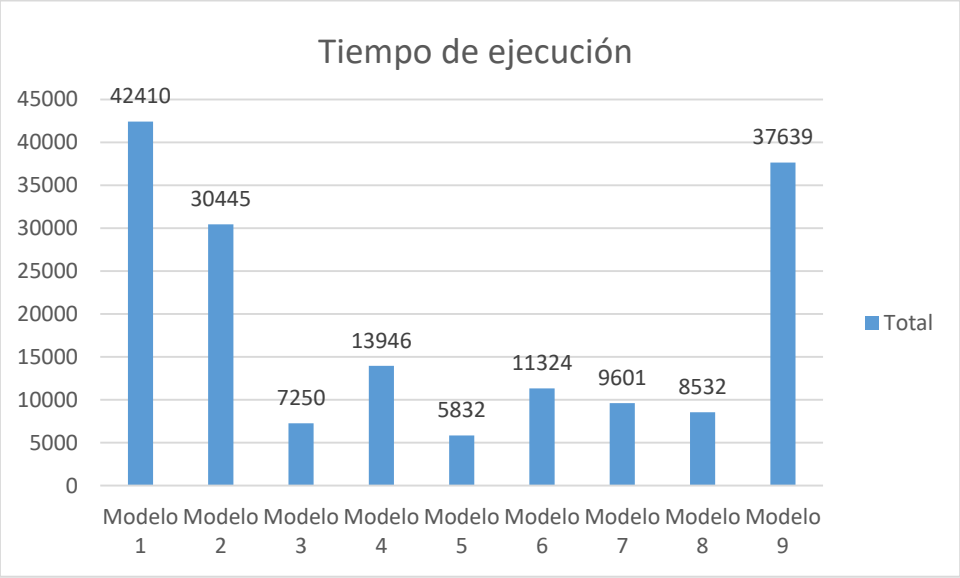


Figura 4.3: Evolución del tiempo de procesamiento por modelos

Destacado el modelo 5 que es el que representa un tiempo menor de ejecucion. No obstante el modelo 3 también dispone de un tiempo de ejecucion bastante bajo y una mayor exactitud, haciendole buen candidato.

4.3.1 Conclusión evaluación de modelos:

En este apartado se evalúan los modelos a modo global, sin entrar en la detección de cada modo de funcionamiento en particular, lo cual se realizará en el siguiente apartado.

Referente al objetivo principal del proyecto de la identificación de funcionamiento en modo fallo, el modelo óptimo es el 4, teniendo un 100% de detección.

Referente al segundo objetivo de clasificación del tipo de fallo. El modelo que mejor exactitud de clasificación tiene es el 9 con un 92%, no obstante, cabe destacar el modelo 3 que dispone de 91,5% de exactitud y un tiempo de ejecución bastante inferior.

Referente a la sensibilidad de los modelos también se aprecia como en el modelo 9 y modelo 3 tienen el valor máximo, es decir, el algoritmo detecta bien la clase a la que pertenece.

También referenciado a la precisión, el modelo 9 y 3 son elevados, es decir en la predicción que hace, habrá pocos datos que no pertenezcan al valor real.

El modelo 9 y modelo 3 también tiene un parámetro de pérdida bajo que deja indicado el error que hay entre el valor real y la predicción realizada.

En relación al modelo 2, se observa el bajo valor de exactitud de validación y de parámetros en general. Esto es debido a que este modelo ha sufrido de sobreaprendizaje, saturándose la red, debido al gran número de capas convolucionales con gran número de filtros utilizados.

En conclusión con el modelo 9 y el modelo 3 se tiene una alta precisión y sensibilidad. Esto quiere decir que los modelos trabajan de manera satisfactoria, estos modelos detectan bien la clase y cuando la detectan lo hace de manera correcta, incluyendo solo un bajo porcentaje de muestras de otros modos de fallo. Estos dos, cumplen satisfactoriamente con el objetivo de detección de modo de funcionamiento normal o de fallo, siendo un poco superior el modelo 9 en comparación al modelo 3, no obstante, el modelo 3 dispone de un tiempo de ejecución mucho menor.

4.4. Evaluación de modos de funcionamiento

Una vez analizado cada modelo de manera global en el apartado anterior, se va proceder a analizar estos modelos profundamente, viendo como detecta cada modo de funcionamiento en particular. Resumiendo, en este bloque vamos a ver qué tal detecta cada modelo los diferentes modos de funcionamiento y ver si hay algún modo de funcionamiento problemático en la detección.

Se recuerda brevemente en la siguiente tabla los modos de funcionamiento.

Tabla 4.6: Descripción de modos de funcionamiento del proceso

Modo	Descripción
0	Funcionamiento Normal
1	Fallo Salk reac - Fallo de alcalinidad en el reactor
2	Fallo Salk influent - Fallo de alcalinidad en el afluente
3	Fallo CO2 - Fallo de Oxigeno
4	Fallo Biomasa influ - Fallo biomasa del afluente
5	Fallo Biomasa reac - Fallo biomasa del reactor
6	Fallo Qr y Qw - Fallo caudales en el circuito de refrigeración
7	Fallo Caudales storage - Fallo en caudales de almacenamiento
8	Fallo Caudales primary - Fallo en caudales primarios
9	Fallo Kla - Fallo en el controlador

Para realizar una correcta evaluación de la detección de los fallos se analizarán los parámetros principales de precisión, sensibilidad y f1-scoore dados por la matriz de confusión.

Aunque se han explicado anteriormente cuando se explicó la matriz de confusión, hacemos un breve recordatorio de estos parámetros:

- Precisión: Evalúa la dispersión del modelo. Una precisión alta indica que el modelo cuando detecta el fallo la detección es confiable.
- Sensibilidad: También llamado *recall* en inglés. Una sensibilidad alta significa que el modelo detecta bien la clase, pero dentro de esa clase puede haber muestras que sean correctas o no. La tasa de que estas muestras sean correctas o no lo mide el parámetro anterior de precisión.
- F1 scoore: Este parámetro es una ponderación de los dos parámetros anteriores, creando una relacion entre ellos y de esta forma tener un resumen rápido de la efectividad del modelo.

A continuación se va a analizar profundamente el comportamiento modelo por modelo viendo cómo actúa en la detección de cada modo de funcionamiento. Se van a ver tres apartados;

A. Estructura del modelo: Breve descripción del número y tipo de capas que forman el modelo, para cada uno de los nueve modelos.
B. Resultados del modelo: Gráfica de la evolución del modelo a través de la modificación de los parámetros de épocas, tasa de aprendizaje y tamaño de lote. Se verá para que valor de esos parámetros se ha conseguido el mejor resultado y un breve comentario.
C. Matriz de confusion: Tabla de matriz de confusion con los datos de validación de cada modo de funcionamiento, donde se ve la cantidad de datos que ha acertado y la cantidad de datos fallados en la predicción. La matriz de confusión será para los parámetros de épocas, tasa de aprendizaje y tamaño de lote óptimo, indicado en el apartado B. También se va a ver en este apartado los valores obtenidos para el modelo, donde se ve de forma numérica lo observado en la tabla de la matriz de confusión.

➢ Modelo 1

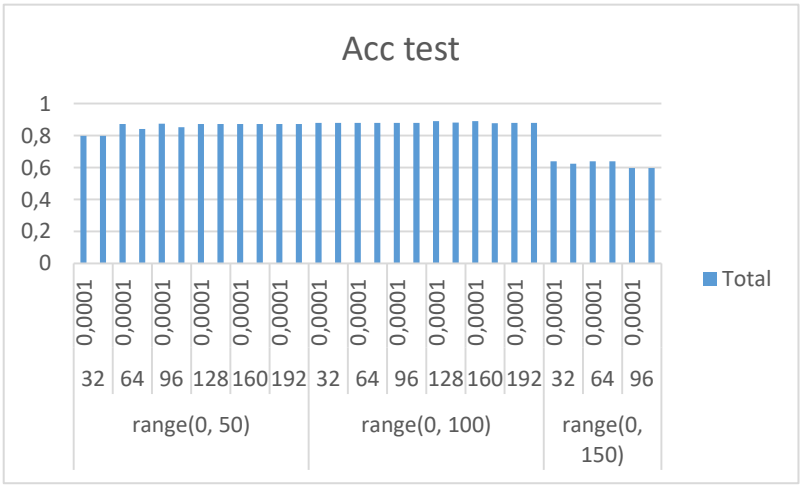
A. Estructura del modelo.

Tabla 4.7: Estructura de red neuronal modelo 1

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7
Conv2D(128filtros)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de siete capas. Las tres primeras capas son convolucionales con 128 filtros cada una, una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación dispone de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, va seguido de una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.



El máximo valor para la exactitud de validación es de 89% conseguida para los siguientes parámetros.

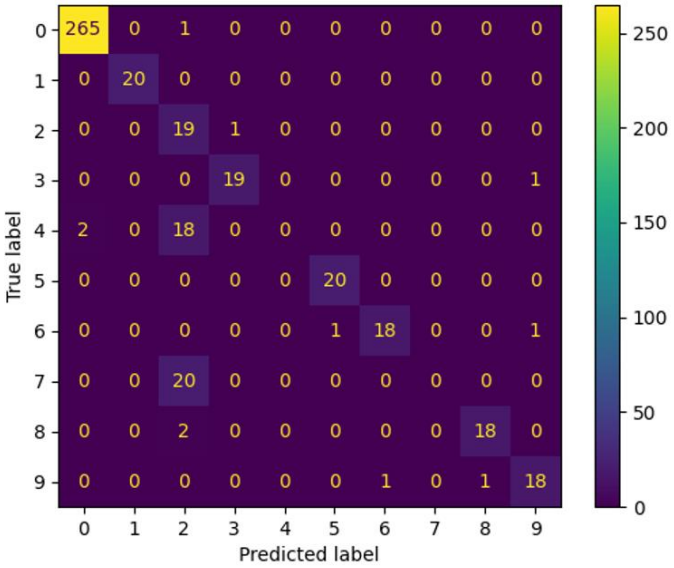
Tabla 4.8: Parámetros para máxima exactitud de validación modelo1

Epochs	Learning rate	Batch
100	0,0001	128

Figura 4.4: Evolución de exactitud de validación modelo 1

Se observa que a partir de 150 epochs, la red neuronal diseñada en este modelo sufre un defecto de aprendizaje, siendo la exactitud de validación alrededor de 60%. Este defecto es producido por el fenómeno ya comentado de sobre aprendizaje que sufre la red neuronal.

C. Matriz de confusión.



Se observa que tanto con el modo de funcionamiento 4 y 7 están totalmente confundido y considerado como modo de funcionamiento 2.

Figura 4.5: Matriz de confusión modelo 1

Tabla 4.9: Valores destacables modelo 1

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,890	0,012	0,895	0,010	42410,44	0,7	0,86	0,76	0,89	0,71	0,87

Tabla 4.10: Valores matriz de confusión modelo 1

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	0,99	1	0,32	0,95	0	0,95	0,95	0	0,95	0,9	0,70	0,86
Recall	1	1	0,95	0,95	0	1	0,9	0	0,9	0,9	0,76	0,89
F1-scoore	0,99	1	0.47	0,95	0	0,98	0,92	0	0,92	0,9	0,67	0,87

Como se ha visto en la matriz de confusion, el modo de funcionamiento 4 y 7 tiene un problema de detección, siendo la precisión y la sensibilidad nulas, mientras que en el modo de funcionamiento 2, la precisión es baja, ya que se han considerado los modos de funcionamiento 4 y 7 como modos de funcionamiento.

➢ Modelo2

A. Estructura del modelo.

Tabla 4.11: Estructura de red neuronal modelo 2

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8
Conv2D(128filtros)	Conv2D(128filtros)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de ocho capas. Las cuatro primeras capas son convolucionales con 128 filtros cada una, una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación va seguida de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, va seguido de una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.

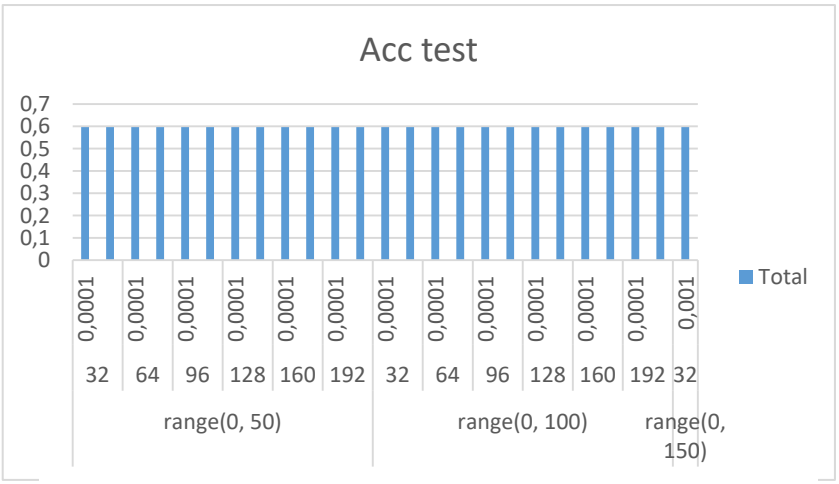
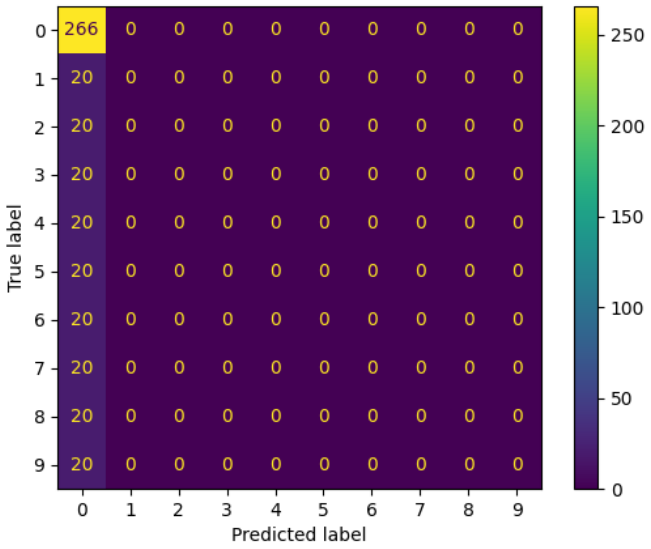


Figura 4.6: Evolución de exactitud de validación modelo 2

En este modelo se observa como en todos los bucles de parámetros usados nos dan el mismo valor de exactitud de validación, sin mejorar el aprendizaje de la red. El valor de la exactitud de validación es siempre alrededor de 59%, siendo bastante bajo. Este modelo no es óptimo para el cumplimiento de los objetivos del proyecto.

C. Matriz de confusion.



En la matriz de confusión de este modelo se observa como todos los datos de validación son considerados como el modo de funcionamiento normal, sin detectar ningún tipo de incidencia. Este modelo no nos seria valido para el proyecto.

Figura 4.7: Matriz de confusión modelo 2

Tabla 4.12: Valores destacables modelo 2

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,596	0,073	0,596	0,073	30445,237	0,060	0,360	0,100	0,600	0,070	0,450

Tabla 4.13: Valores matriz de confusión modelo 2

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	0,6	0	0	0	0	0	0	0	0	0	0,06	0,36
Recall	1	0	0	0	0	0	0	0	0	0	0,1	0,6
F1-scoore	0,75	0	0	0	0	0	0	0	0	0	0,075	0,45

Los valores de los resultados de esta matriz son muy bajos, siendo la sensibilidad del modo de funcionamiento normal del 100% ya que los 266 datos de validación les a acertado, no obstante en los demás modos de funcionamiento no ha realizado una correcta clasificación.

➢ Modelo 3

A. Estructura del modelo.

Tabla 4.14: Estructura de red neuronal modelo 3

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6
Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de seis capas, siendo el modelo que menos capas dispone entre todos los modelos realizados. La primeras capa es convolucional con 64 filtros, una dimensión de kernel de (3,3) y un paso del kernel de 1, continuando con una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, volviendo a una capa convolucional de 128 filtros, dimensión de kernel (3,3) y un paso de kernel 1, seguido de una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.

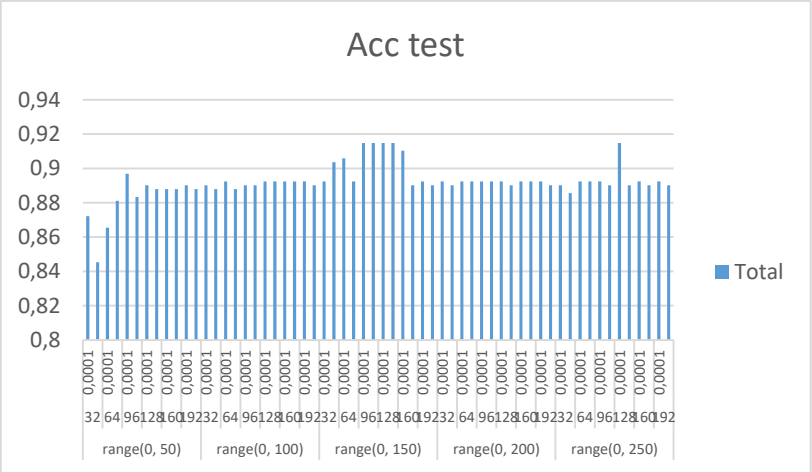


Figura 4.8: Evolución de exactitud de validación modelo 3

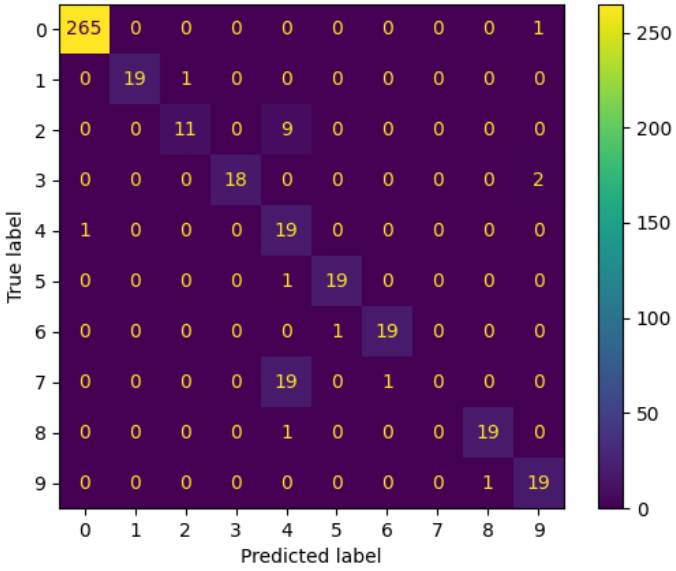
En el rango de 150 épocas se observa como el modelo trabaja con un aprendizaje más elevado, con un valor máximo de exactitud de validación de 91,5%, conseguido con los siguientes parámetros.

Tabla 4.15: Parámetros para máxima exactitud de validación modelo 3

Epochs	Learning rate	Batch
150	0,001	96

El rango de aprendizaje en este modelo se mantiene estable y alto desde el inicio. El rango de épocas óptimo es 150, donde sufre una mejora en el aprendizaje del modelo, llegando hasta el 91,5% de exactitud de validación.

C. Matriz de confusion.



El modo de funcionamiento 7 no es reconocido y es considerado como modo de funcionamiento 4, mientras que el 45% del modo de funcionamiento 2 es considerado como modo de funcionamiento 4.

Figura 4.9: Matriz de confusión modelo 3

Tabla 4.16: Valores destacables modelo 3

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,915	0,010	0,915	0,009	7250,005	0,800	0,910	0,810	0,910	0,790	0,900

Tabla 4.17: Valores matriz de confusión modelo 3

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	1	1	0,92	1	0,39	0,95	0,95	0	0,95	0,86	0,802	0,91
Recall	1	0,95	0,55	0,9	0,95	0,95	0,95	0	0,95	0,95	0,815	0,91
F1-scoore	1	0,97	0,69	0,95	0,55	0,95	0,95	0	0,95	0,9	0,791	0,9

Este modelo presenta un alto nivel de clasificación, con una exactitud de validación del 91,5%.

Tiene una debilidad de detección en los modos de funcionamiento 2 y 7 que a veces es considerado como modo de funcionamiento 4.

➢ Modelo 4

A. Estructura del modelo.

Tabla 4.18: Estructura de red neuronal modelo 4

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8
Conv2D(64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de ocho capas. Las dos primeras capas son convolucionales con 64 filtros cada una, una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación dispone de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, volviendo a una capa convulacional esta vez de 128 filtros y misma dimensión y paso que la primera, continuando con otra capa *MaxPooling* de mismas dimensiones que la capa 3, a posterior hay una capa fullyconected de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.

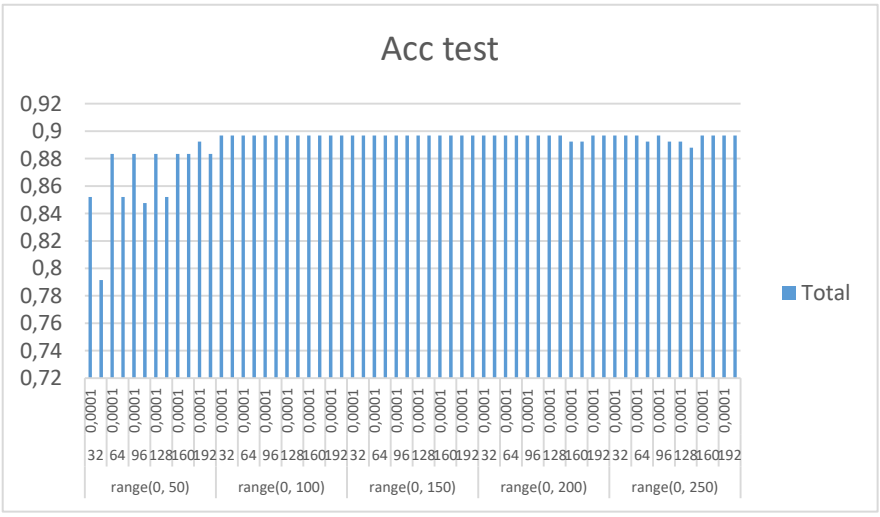


Figura 4.10: Evolución de exactitud de validación modelo 4

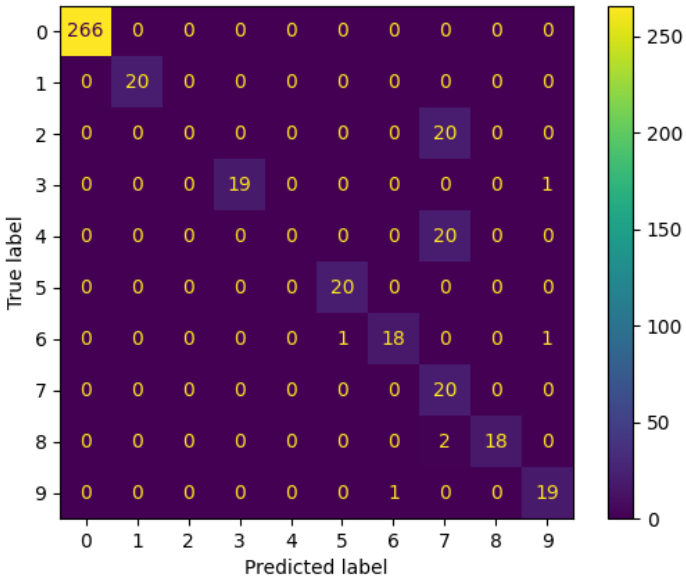
El máximo valor para la exactitud de validación es de 89,7% conseguida para los siguientes parámetros.

Tabla 4.19: Parámetros para máxima exactitud de validación modelo 4

Epochs	Learning rate	Batch
100	0,001	32

Se observa en este modelo que para el rango de 50 epochs en nivel de aprendizaje no era óptimo, en cambio a partir de 100 epochs, este nivel es óptimo, volviéndose constante para las demás épocas.

C. Matriz de confusion.



Los modos de funcionamiento 2 y 4 en su totalidad con considerados como modo de funcionamiento 7.

Figura 4.11: Matriz de confusión modelo 4

Tabla 4.20 Valores destacables modelo 4

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,897	0,010	0,896	0,010	13945,540	0,710	0,870	0,770	0,900	0,720	0,880

Tabla 4.21: Valores matriz de confusión modelo 4

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	1	1	0	1	0	0,95	0,95	0,32	1	0,9	0,712	0,87
Recall	1	1	0	0,95	0	1	0,9	1	0,9	0,95	0,77	0,9
F1-scoore	1	1	0	0,97	0	0,98	0,92	0,49	0,95	0,93	0,724	0,88

Observado en la matriz de confusion, la clasificación de este modelo es elevada a excepción de los modos de funcionamiento 2, 4 y 7. Ya que los dos primeros son considerados como el modo 7, por ese motivo aunque la sensibilidad del modo 7 sea elevada, la precisión será baja, debido a que dentro de los datos que haya considerado como modo 7 habrá datos del modo 2 y 4.

➢ Modelo 5

A. Estructura del modelo.

Tabla 4.22: Estructura de red neuronal modelo 5

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8
Conv2D(64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	MaxPooling2D(2,1)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de ocho capas. Las dos primeras capas son convolucionales con 64 filtros cada una, una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación dispone de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, volviendo a una capa convulacional esta vez de 128 filtros y misma dimensión y paso que la primera, continuando con otra capa *MaxPooling* de dimensión de filtro de (2,1) y paso de 2, a posterior hay una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.

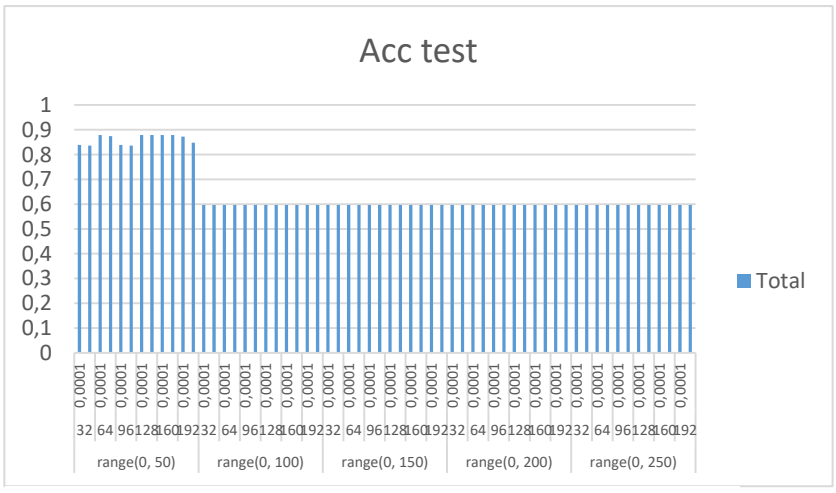


Figura 4.12: Evolución de exactitud de validación modelo 5

El máximo valor de la exactitud de validación es de 87,9% conseguida para los siguientes parámetros.

Tabla 4.23: Parámetros para máxima exactitud de validación modelo 5

Epochs	Learning rate	Batch
50	0,0001	64

Analizando la gráfica de arriba, en el rango de 50 épocas la red neuronal trabaja de forma correcta, no obstante para épocas superiores la red cae en el sobre aprendizaje, dando un valor de exactitud de validación bastante bajo

C. Matriz de confusion.

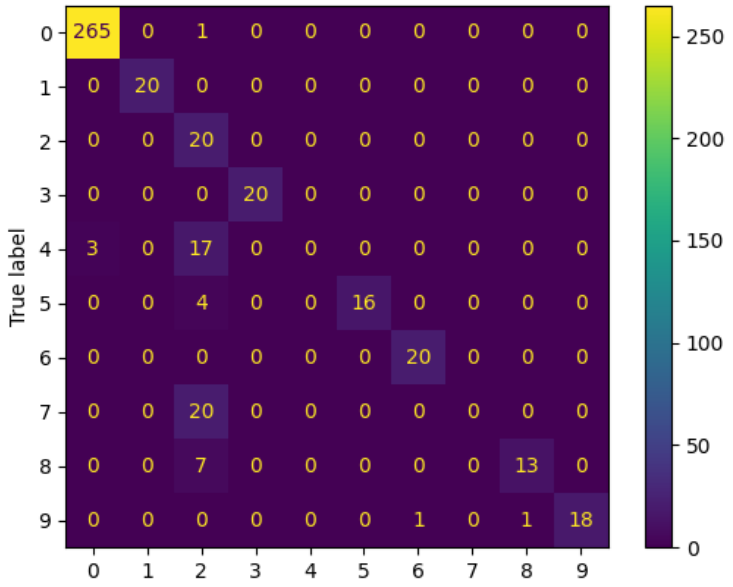


Figura 4.13: Matriz de confusión modelo 5

En este modelo el modo de funcionamiento 4 y 7 son considerados como modos de funcionamiento 2, también en este modelo, el 35% del modo de funcionamiento es considerado como modo 2.

Tabla 4.24: Valores destacables modelo 5

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,879	0,015	0,872	0,014	5832,494	0,720	0,870	0,730	0,880	0,700	0,860

Tabla 4.25: Valores matriz de confusión modelo 5

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	0,99	1	0,29	1	0	1	0,95	0	0,93	1	0,716	0,87
Recall	1	1	1	1	0	0,8	1	0	0,65	0,9	0,735	0,88
F1-scoore	0,99	1	0,45	1	0	0,89	0,98	0	0,76	0,95	0,702	0,86

Los valores de los parámetros de la red, ponen de manifiesto la detección del modo de funcionamiento 4 y 7, cuyo valor es nulo y el modo de funcionamiento 8 con una sensibilidad del 65%. Esto produce que aunque la sensibilidad del modo de funcionamiento sea del 100% ya que todos los modos de funcionamiento del tipo 2 se han clasificado correctamente, la precisión es del 29% debido a que también se han considerado más modos de fallos del tipo 2 siendo esto incorrecto.

➢ Modelo 6

A. Estructura del modelo.

Tabla 4.26: Estructura de red neuronal modelo 6

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8
Conv2D(64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	MaxPooling2D(1,2)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de ocho capas. Las dos primeras capas son convolucionales con 64 filtros cada una, una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación dispone de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, volviendo a una capa convulacional esta vez de 128 filtros y misma dimensión y paso que la primera, continuando con otra capa *MaxPooling* de dimensión de filtro de (1,2) y paso de 2, a posterior hay una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.

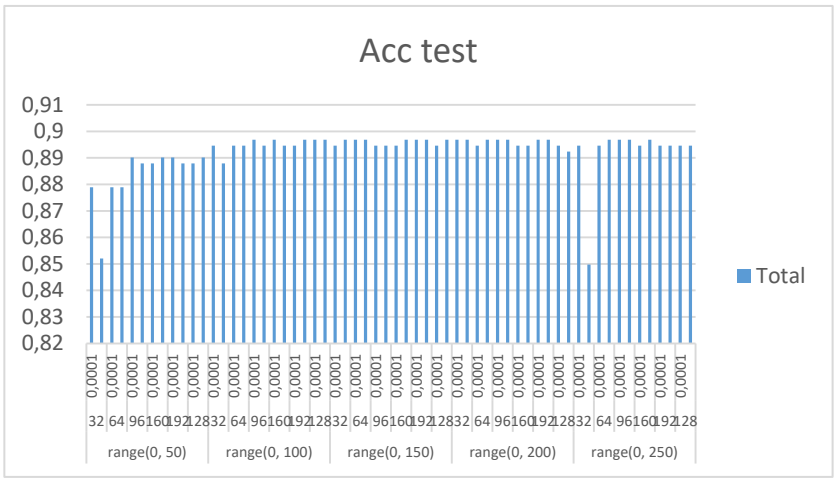


Figura 4.14: Evolución de exactitud de validación modelo 6

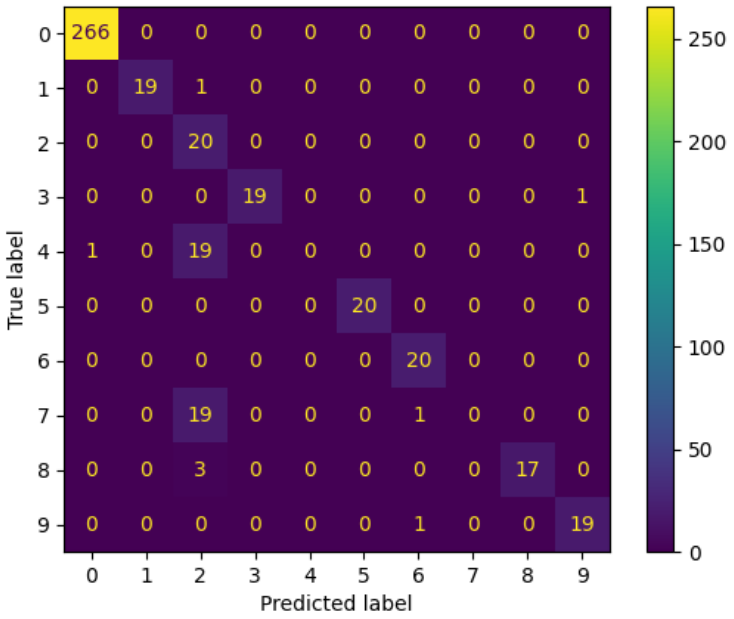
El máximo valor de la exactitud de validación es de 89,7% conseguida para los siguientes parámetros.

Tabla 4.27: Parámetros para máxima exactitud de validación modelo 6

Epochs	Learning rate	Batch
100	0,0001	96

Se observa como en la gráfica superior como el aprendizaje de la red se mantiene elevado y constante a lo largo del bucle de los parámetros.

C. Matriz de confusion.



Se observa que tanto con el modo de funcionamiento 4 y 7 están totalmente confundido y considerado como modo de funcionamiento 2.

Figura 4.15: Matriz de confusión modelo 6

Tabla 4.28: Valores destacables modelo 6

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,897	0,011	0,900	0,010	11323,682	0,720	0,870	0,770	0,900	0,730	0,880

Tabla 4.29: Valores matriz de confusión modelo 6

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	1	1	0,32	1	0	1	0,91	0	1	0,95	0,718	0,87
Recall	1	0,95	1	0,95	0	1	1	0	0,85	0,95	0,77	0,9
F1-scoore	1	0,97	0,49	0,97	0	1	0,95	0	0,92	0,95	0,725	0,88

La tabla de valores que se muestra arriba pone de manifiesto lo que se ha reflejado en la matriz de confusion. Para los modos de fallo 4 y 7 los valores son nulos, ya que el modelo no los detecta, en cambio para el modo de funcionamiento 2, la sensibilidad es del 100% en cambio la precisión es del 32% ya que en esta identificación tiene modos de funcionamiento 2, 4 y 7.

➢ Modelo 7

A. Estructura del modelo.

Tabla 4.30: Estructura de red neuronal modelo 7

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8
Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de ocho capas. La primera capa es una convolucional con 64 filtros con una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación dispone de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, volviendo a dos capas convulacionales esta vez de 128 filtros y misma dimensión y paso que la primera, continuando con otra capa *MaxPooling* de dimensión de filtro de (2,2) y paso de 2, a posterior hay una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.

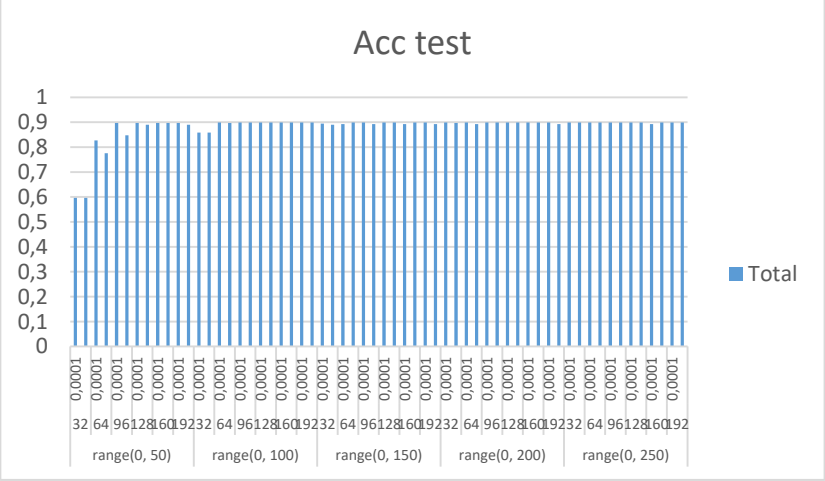


Figura 4.16: Evolución de exactitud de validación modelo 7

El máximo valor de la exactitud de validación es de 89,9% conseguida para los siguientes parámetros.

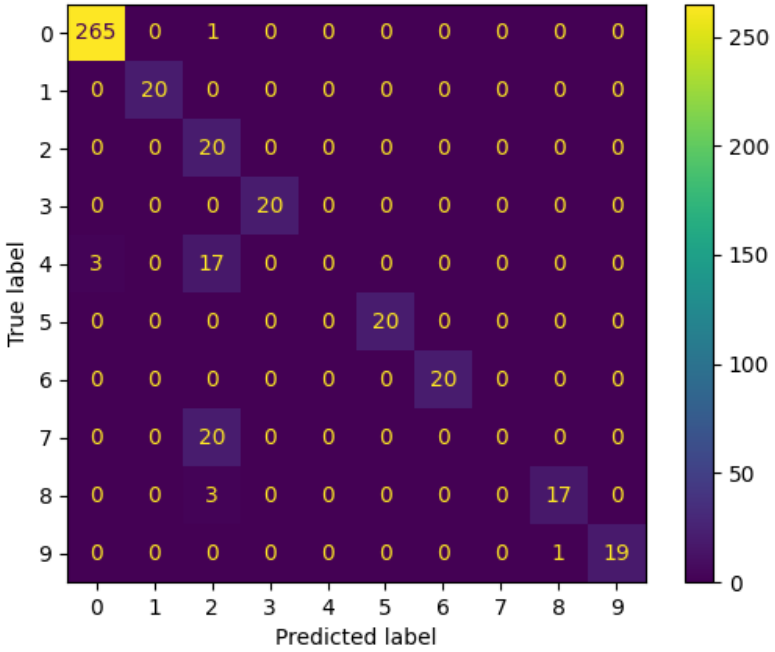
Tabla 4.31: Parámetros para máxima exactitud de validación modelo 7

Epochs	Learning rate	Batch
100	0,0001	64

Este modelo da resultados parecidos al modelo 6, la optimización de aprendizaje al igual que en el modelo anterior se produce a partir de las 50 epocas. Al igual que el modelo 6 la precisión de validación se mantiene constante alrededor de un 89%.

Se puede decir que de la estructura del modelo 6 no se ve gran diferencia en con la del modelo7.

C. Matriz de confusion



La matriz de decisión es muy semejante a la del modelo anterior, considerando los modos de funcionamiento 4 y 7 en su totalidad por modos de funcionamiento 2.

Figura 4.17: Matriz de confusión modelo 7

Tabla 4.32: Valores destacables modelo 7

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,899	0,010	0,888	0,011	9600,808	0,730	0,870	0,780	0,900	0,740	0,880

Tabla 4.33: Valores matriz de confusión modelo 7

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	0,99	1	0,33	1	0	1	1	0	0,94	1	0,726	0,87
Recall	1	1	1	1	0	1	1	0	0,85	0,95	0,78	0,9
F1-scoore	0,99	1	0,49	1	0	1	1	0	0,89	0,97	0,734	0,88

Los valores del modelo muestran unos resultados parejos al modelo 6, como ha dejado reflejado la matriz de confusión.

➢ Modelo 8

A. Estructura del modelo.

Tabla 4.34: Estructura de red neuronal modelo 8

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8	Capa 9
Conv2D(64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,2)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de 9 capas. Las dos primeras capas son convolucionales con 64 filtros cada una, una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación dispone de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, volviendo a dos capas convulacionales esta vez de 128 filtros y misma dimensión y paso que la primera, continuando con otra capa *MaxPooling* de dimensión de filtro de (2,2) y paso de 2, a posterior hay una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

B. Resultados del modelo.

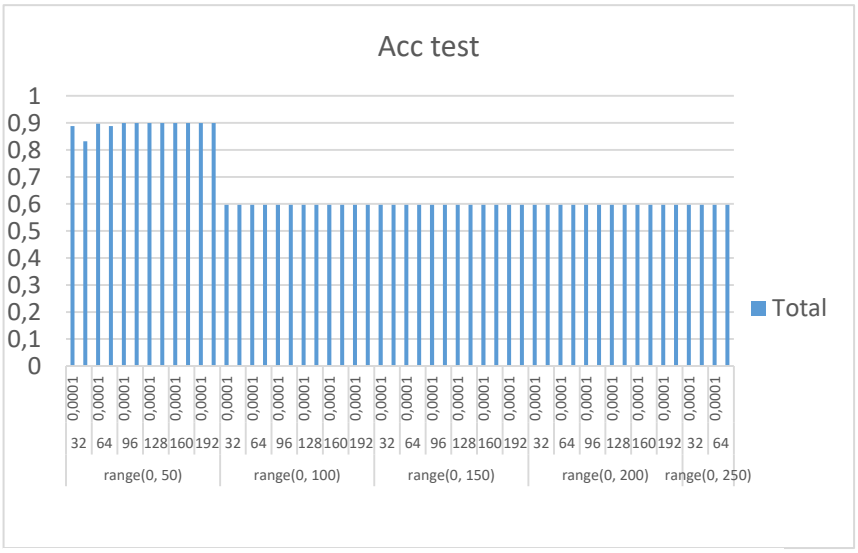


Figura 4.18: Evolución de exactitud de validación modelo 8

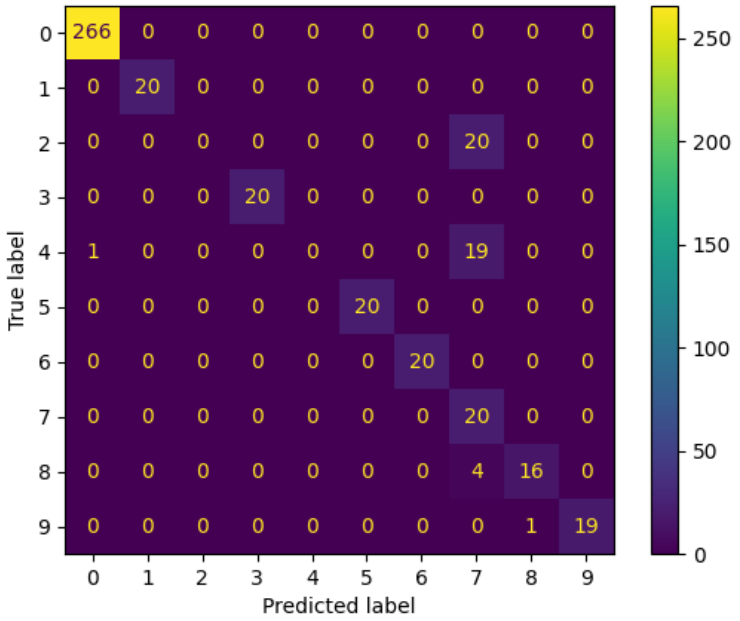
El máximo valor de la exactitud de validación es de 89,9% conseguida para los siguientes parámetros.

Tabla 4.35: Parámetros para máxima exactitud de validación modelo 8

Epochs	Learning rate	Batch
50	0,001	96

Se observa como en la gráfica superior como a partir de 50 épocas la red sufre de sobre aprendizaje, causando esto que la tasa de aprendizaje sea baja.

C. Matriz de confusion.



En este caso, la matriz de confusión refleja como los modos de funcionamiento 2 y 4 son considerados como el modo de funcionamiento 7.

Figura 4.19: Matriz de confusión modelo 8

Tabla 4.36: Valores destacables modelo 8

Acc test	Loss test	Acc train	Loss train	Tiempo	Precision	Precision ponderada	Recall	Recall ponderada	F1 score	F1 score ponderado
0,899	0,010	0,890	0,011	8532,477	0,730	0,870	0,780	0,900	0,730	0,880

Tabla 4.37: Valores matriz de confusión modelo 8

Modo	0	1	2	3	4	5	6	7	8	9	Media	Media ponderada
Precisión	1	1	0	1	0	1	1	0,32	0,94	1	0,726	0,87
Recall	1	1	0	1	0	1	1	1	0,8	0,95	0,775	0,9
F1-scoore	1	1	0	1	0	1	1	0,48	0,86	0,97	0,731	0,88

Los valores que indica la tabla, se ve la buena clasificación de los modos de funcionamiento a excepción de los modos de fallo 2 y 4, siendo considerados de modo de funcionamiento 7, lo que genera que la precisión del modo de fallo 7 disminuya, con un valor de 32%.

➢ Modelo 9

A. Estructura del modelo.

Tabla 4.38: Estructura de red neuronal modelo 9

Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8	Capa 9	Capa 10
Conv2D(64filtros)	Conv2D(64filtros)	Conv2D(64filtros)	MaxPooling2D(2,2)	Conv2D(128filtros)	Conv2D(128filtros)	MaxPooling2D(2,1)	FullyConnected(300)	Dropout(0,7)	FullyConnected(10)

Consta de una estructura de 10 capas. Las tres primeras capas son convolucionales con 64 filtros cada una, una dimensión de kernel de (3,3) y un paso del kernel de 1. A continuación dispone de una capa *Maxpooling* con una dimensión de filtro de (2,2) y un paso de 2, volviendo a dos capas convulacionales esta vez de 128 filtros y misma dimensión y paso que la primera, continuando con otra capa *MaxPooling* de dimensión de filtro de (2,2) y paso de 2, a posterior hay una capa *fullyconected* de dimensión 300. A continuación le sigue una capa *Dropout* de parámetro 0,7 y finalmente la capa de salida será una capa fully connected con valor de parámetro 10, ya que tenemos 10 tipos de clasificación.

A diferencia con el modelo anterior, en este modelo se ha añadido una capa convolucional adicional.

B. Resultados del modelo.

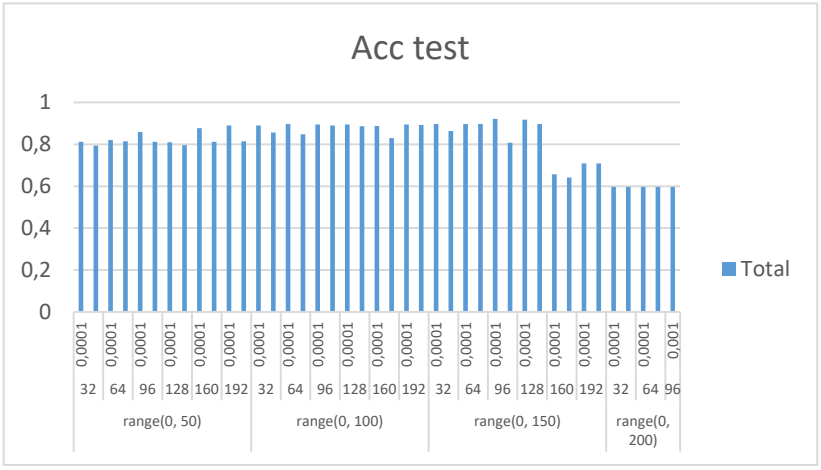


Figura 4.20: Evolución de exactitud de validación modelo 9

El máximo valor de la exactitud de validación es de 92,2% conseguida para los siguientes parámetros.

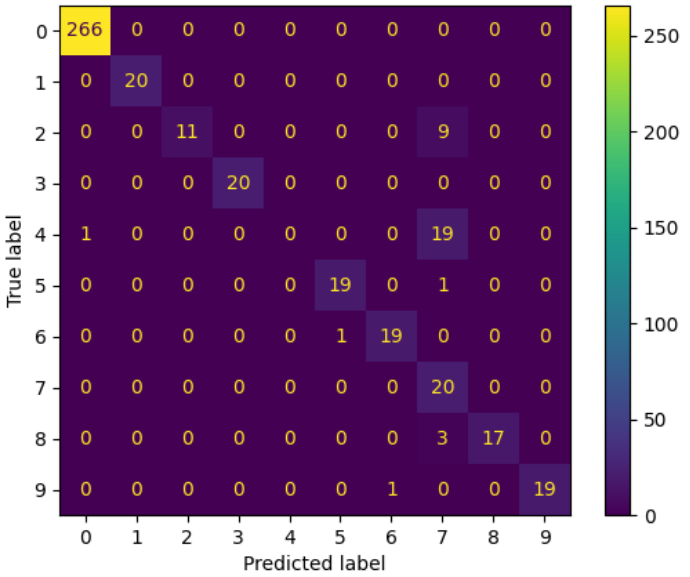
Tabla 4.39: Parámetros para máxima exactitud de validación modelo 9

Epochs	Learning rate	Batch
150	0,0001	96

Añadiendo una capa a mayores convulacional se ha solucionado el problema del sobreaprendizaje hasta las 150 épocas, volviendo otra vez a partir de las 150 épocas con un tamaño de lote de 160.

Con este modelo se ha conseguido el máximo valor de exactitud de validación. Un valor bastante satisfactorio.

C. Matriz de confusion.



En este modelo se ha conseguido clasificar parte de los modos de funcionamiento 2, 4 y 7.

Figura 4.21 Matriz de confusión modelo 9

Tabla 4.40: Valores destacables modelo 9

Acc test	Loss test	Acc train	Loss train	Tiempo	Precisio n	Precision ponderad a	Recal l	Recall ponderad a	F1 score	F1 score ponderad o
0,922	0,010	0,904	0,010	37639,37 1	0,830	0,920	0,820	0,920	0,810	0,910

Tabla 4.41: Valores matriz de confusión modelo 9

Modo	0	1	2	3	4	5	6	7	8	9	Medi a	Media ponderad a
Precisió n	1	1	1	1	0	0,95	0,95	0,38	1	1	0,828	0,92
Recall	1	1	0,55	1	0	0,95	0,95	1	0,85	0,95	0,825	0,92
F1-scoore	1	1	0,71	1	0	0,95	0,95	0,56	0,92	0,97	0,806	0,91

En este modelo se ha conseguido clasificar parte de los modos de fallos 2, 4 y 7. El 55% del modo de funcionamiento 2 se ha clasificado correctamente, mientras que el resto ha sido considerado como modo de funcionamiento 7. El modo de funcionamiento 4 ha sido considerado como modo de funcionamiento 7.

Una vez detallado cada modelo, a continuación se pueden ver todos los datos de los diferentes modelos con sus diferentes modos de funcionamiento en conjunto, permitiendo realizar una comparación entre ellos de manera más sencilla.

La tabla 4.42 muestra todos los modelos con todos los modos de funcionamiento de los valores de precisión, sensibilidad y f1 score. Se indica la media y la media ponderada de los resultados ya que cada modo no tiene el mismo peso debido a que no tiene el mismo número de entradas. En la última columna se detalla cual sería el mejor modelo para cada modo de fallo. Igualmente, el mejor resultado para cada modo de fallo está indicado en verde.

Tabla 4.42: Valores de precisión, exactitud y f1 score para dos los modelos y modos de funcionamiento

Modo	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Modelo 6	Modelo 7	Modelo 8	Modelo 9	Media	Mejor modelo
Precisión											
0	0,99	0,6	1	1	0,99	1	0,99	1	1	0,95	3,4,6,8.9
1	1	0	1	1	1	1	1	1	1	0,89	1,3,4,5,6,7,8,9
2	0,32	0	0,92	0	0,29	0,32	0,33	0	1	0,35	9
3	0,95	0	1	1	1	1	1	1	1	0,88	3,4,5,6,7,8,9
4	0	0	0,39	0	0	0	0	0	0	0,04	3
5	0,95	0	0,95	0,95	1	1	1	1	0,95	0,87	5,6,7,8
6	0,95	0	0,95	0,95	0,95	0,91	1	1	0,95	0,85	7,8
7	0	0	0	0,32	0	0	0	0,32	0,38	0,11	9
8	0,95	0	0,95	1	0,93	1	0,94	0,94	1	0,86	4,6,9
9	0,9	0	0,86	0,9	1	0,95	1	1	1	0,85	5,7,8,9
Media	0,70	0,06	0,80	0,71	0,72	0,72	0,73	0,73	0,83	0,67	9
Media ponderada	0,86	0,36	0,91	0,87	0,87	0,87	0,87	0,87	0,92	0,82	9
Recall											
0	1	1	1	1	1	1	1	1	1	1,00	1,2,3,4,5,6,7,8,9
1	1	0	0,95	1	1	0,95	1	1	1	0,88	1,4,5,7,8,9
2	0,95	0	0,55	0	1	1	1	0	0,55	0,56	5,6,7
3	0,95	0	0,9	0,95	1	0,95	1	1	1	0,86	5,7,8,9
4	0	0	0,95	0	0	0	0	0	0	0,11	3
5	1	0	0,95	1	0,8	1	1	1	0,95	0,86	1,4,6,7,8
6	0,9	0	0,95	0,9	1	1	1	1	0,95	0,86	5,6,7,8
7	0	0	0	1	0	0	0	1	1	0,33	4,8,9
8	0,9	0	0,95	0,9	0,65	0,85	0,85	0,8	0,85	0,75	3
9	0,9	0	0,95	0,95	0,9	0,95	0,95	0,95	0,95	0,83	3,4,6,7,8,9
Media	0,76	0,10	0,82	0,77	0,74	0,77	0,78	0,78	0,83	0,70	9

Media ponderada	0,89	0,60	0,91	0,90	0,88	0,90	0,90	0,90	0,92	0,87	9
					F1-scoore						
0	0,99	0,75	1	1	0,99	1	0,99	1	1	0,97	3,4,6,8,9
1	1	0	0,97	1	1	0,97	1	1	1	0,88	1,4,5,7,8,9
2	0.47	0	0,69	0	0,45	0,49	0,49	0	0,71	0,31	9
3	0,95	0	0,95	0,97	1	0,97	1	1	1	0,87	5,7,8,9
4	0	0	0,55	0	0	0	0	0	0	0,06	3
5	0,98	0	0,95	0,98	0,89	1	1	1	0,95	0,86	6,7,8
6	0,92	0	0,95	0,92	0,98	0,95	1	1	0,95	0,85	7,8
7	0	0	0	0,49	0	0	0	0,48	0,56	0,17	9
8	0,92	0	0,95	0,95	0,76	0,92	0,89	0,86	0,92	0,80	3
9	0,9	0	0,9	0,93	0,95	0,95	0,97		0,97	0,97	0,84	7,8,9
Media	0,67	0,08	0,79	0,72	0,70	0,73	0,73		0,73	0,81	0,66	9
Media ponderada	0,87	0,45	0,90	0,88	0,86	0,88	0,88	0,88	0,91	0,83	9	

4.4.1 Conclusión evaluación modos de fallo;

Entrando en este apartado más en el detalle de detección de los modos de funcionamiento. Vemos como en los modelos creados, la identificación del modo de funcionamiento es bastante buena a excepción de los modos de funcionamiento 2,4 y 7.

Con el modelo 9 se ha mejorado la identificación entre estos tres modos de funcionamiento, no obstante sigue siendo confusa entre ellos. Es por ello que se ha entrado más en el detallo del análisis de estos modos de funcionamiento.

Para ver la naturaleza de los datos de entrada se ha hecho uso del metodo *Stochastic Neighbor Embedding* (t-SNE). Este metodo lo que realiza es incrustar las características de altas dimensiones en un espacio de dos o tres dimensiones. Con esto podemos ver gráficamente la naturaleza de los datos. Se puede ver más sobre este metodo en la ref [4].

En la figura 4.22 que se muestra a continuación podemos ver la base de datos representada en grafica 2D aplicando el método t-SNE.

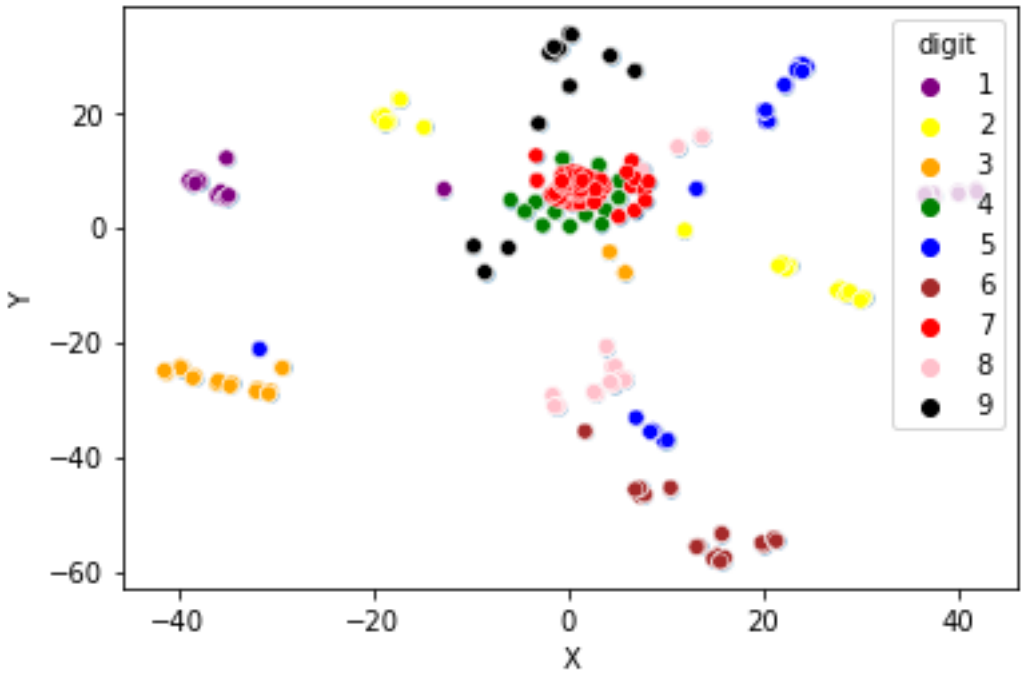


Figura 4.22: Representación t-SNE datos de entrada

Se observa como la naturaleza de los modos de fallo 4 (en verde) y 7(en rojo) son muy parecidos, haciendo difícil su clasificación. También se observa como el modo de fallo 2(en amarillo) tiene una gran dispersión en su naturaleza de datos, haciéndolo a su vez también de difícil clasificación.

4.5. Conclusión de resultados

Este proyecto tiene como principal objetivo la detección del funcionamiento del proceso en modo normal o en modo fallo y como segundo objetivo la identificación y clasificación del modo de fallo.

El primer objetivo ha sido cumplido con un valor del 100% con el modelo 4. Esto quiere decir que el algoritmo de detección de fallos que se ha realizado en este proyecto es bastante fiable para la detección del fallo del proceso.

Como segundo objetico de identificación y clasificación del fallo, se ha obtenido un 92,15% de correcta clasificación para el modelo 9. Este valor es bastante fiable para el proceso.

El valor de clasificación del modo de funcionamiento es bastante elevado, con las técnicas de clasificación de datos que existen actualmente, quedando de manifiesto los buenos resultados que se obtienen con la técnica del *Deep Convolutional Neural Network* (DCNN) usada en este proyecto.

En comparación con otros artículos científicos, los cuales han aplicado también el método de DCNN, el resultado de clasificación del 92,15% es bastante elevado. Se puede comparar con el artículo científico de la ref [4] donde aplica esta técnica de DCNN a la detección de fallos en un proceso químico denominado *Tenneseee Eastman process*. Este artículo consigue un valor máximo de 88,4% de media de validación, valor también elevado.

Para la obtención del valor de detección del presente proyecto, a parte del uso de la técnica del DCNN, ha sido conseguido por un buen preprocesamiento y preparación de la base de datos, punto crucial para la obtención de buenos resultados. También se debe tener en cuenta la metodología empleada y las estructuras de las redes neuronales utilizadas.

El análisis de resultados muestra como los datos del modo de funcionamiento 2,4 y 7 debido a su naturaleza y semejanza, son de difícil detección. Con el modelo 9 se ha conseguido mejorar la clasificación entre estos tres modos de funcionamiento.

Capítulo 5
ESTUDIO ECONÓMICO

5. ESTUDIO ECONÓMICO

En este apartado se realiza una evaluación del impacto económico de los recursos empleados en la realización del proyecto. Recursos materiales asi como de tiempo que se han empleado en las diferentes etapas del mismo.

A mayores del impacto económico, también se indica la planificación y organización de tareas a llevar a cabo para la realización del proyecto.

5.1. Planificación

Se han establecido la organización y orden de tareas a realizar durante el proyecto con un total de 6 bloques de tareas con sus subtareas correspondientes.

En la tabla que se muestra a continuación se puede ver el desarrollo de estos bloques y una breve descripción de las tareas que se deben de realizar.

Tabla 5.1: Tareas a realizar en el proyecto

Nº	Tarea	Subtarea	Descripción
1	Asignación de proyecto a realizar	Pre análisis y búsqueda de informacion	Búsqueda de información de trabajo a realizar, campos de aplicaciones, relación con competencias del master impartido en la escuela, experiencia adquirida para aplicación al mundo laboral.
		Definición de objetivos a alcanzar	Se fija los objetivos que se quieren adquirir con la realización del proyecto. Conocimientos a adquirir en el uso de redes neuronales con orientación a experiencia profesional.
2	Estudio y análisis de información	Búsqueda de información	Investigación de herramientas y métodos a aplicar para la IA y ML. Investigación de fuentes oficiales y científicamente validadas para la búsqueda de información correctamente contrastada.
		Estudio de apuntes cursados	Estudio de material dado en la escuela durante el curso del master en la asignatura de Tecnología de control.
		Autoformación	Adquisición de conocimientos. -Lenguaje de programación Python. -Mundo de la inteligencia artificial. -Detección y diagnóstico de fallos. -Técnicas de machine Learning: Deep Learning
		Análisis de recursos de hardware	Verificación de recursos a tener en cuenta para la realización del proyecto. -Hardware: Características de ordenador, ordenador de sustitución, soportes de material de la universidad. -Software: Lista de programas a usar. -Acceso a material de universidad: Bibliotecas librería.
3	Planteamiento de metodología	Definición de objetivos a alcanzar	Establecer objetivos que se quieren alcanzar con la parte práctica. Estudio de posibles objetivos a alcanzar con las técnicas usadas, investigando cual es la mejor técnica para alcanzar el mejor objetivo.
		Análisis de la base de datos a trabajar	Análisis de la base de datos proporcionada por la universidad: formato de los datos, variables, cantidad de datos a tratas, dimensional dad, tipo de procesamiento a realizar, forma de tratamiento.
		Definición de metodología	Establecer la metodología de la programación: -Reprocesamiento de los datos. Normalización de los datos, reducción de tamaño de datos, establecer modo de lectura de los datos, automatización de lectura, etc. -Realización de algoritmos de ML: -Creación algoritmo de red neuronal. -Registro de datos: Análisis de los datos del programa que queremos registrar y explorar, establecer el metodo, manera y formato de la informacion a registrar.
		Desarrollo de código	Realizar código con la metodología definida en el punto anterior
4	Implementación de metodología	Implementación de código	Ajuste de código, solución de errores y optimización de tratamiento de la información
		Correr código	Correr código en servidor de la Universidad.
		Análisis de resultados	Estudio de resultados obtenidos y planteamiento de posibles errores y mejoras para la optimización del resultado.
		Corrección de código.	Modificación del código para obtener resultados acorde a los objetivos establecidos. Realizar punto de análisis de resultado y corrección tantas veces como sea necesario
		Análisis de resultados finales	Estudio y comparación entre modelos. Comparación con los objetivos establecidos
		Conclusión	Síntesis de estudio realizado y cumplimiento de objetivos. Calidad y validez de resultados
5	Memoria	Elaboración de memoria	Realización de presente documento, redacción corrección de documento por parte de tutor. Garantizando un correcto formato con bibliografía y cumpliendo normas de elaboración.
6	Presentación	Elaboración de presentacion	Realización de presentación, plasmando los aspectos importantes del proyecto.
		Preparación de presentacion	Estudio y practica de presentación
		Realización de presentacion	Exposición de presentacion ante tribunal

En la tabla que se muestra a continuación se ha desplegado las tareas a realizar durante el desarrollo del proyecto. En este despliegue se incluye informacion de la duración de cada tarea y la fecha de inicio y fin, con la finalidad de tener una buena organización en el proyecto. Posteriormente se adjunta el diagrama de Gantt para de esta manera, ver en formato grafico la secuenciación de tareas a realizar y analizar aquellas tareas que se pueden hacer paralelamente y optimizar tiempos.

Tabla 5.2: Duración de tareas a realizar

Nº	Nombre tarea	Duración	Comienzo	Fin
1	**Asignación de proyecto a realizar**	**12 Dias**	**16/01/2021**	**02/02/2021**
	Pre análisis y búsqueda de información	8 Dias	16/01/2021	26/01/2021
	Definición de objetivos a alcanzar	5 Dias	27/01/2021	02/02/2021
2	**Estudio y análisis de información**	**43 Dias**	**03/02/2021**	**02/04/2021**
	Búsqueda de información	28 Dias	03/02/2021	12/03/2021
	Estudio de apuntes cursados	15 Dias	22/02/2021	12/03/2021
	Autoformación	43 Dias	03/02/2021	02/04/2021
	Análisis de recursos de hardware	5 Dias	12/03/2021	18/03/2021
3	**Planteamiento de metodología**	**61 Dias**	**15/03/2021**	**07/06/2021**
	Definición de objetivos a alcanzar	5 Dias	15/03/2021	19/03/2021
	Análisis de la base de datos a trabajar	15 Dias	22/03/2021	09/04/2021
	Definición de metodología	13 Dias	12/04/2021	28/04/2021
	Desarrollo de código	28 Dias	29/04/2021	07/06/2021
4	**Implementación de metodología**	**98 Dias**	**08/06/2021**	**21/10/2021**
	Implementación de código	24 Dias	08/06/2021	09/07/2021
	Correr código	40 Dias	12/07/2021	03/09/2021
	Análisis de resultados	4 Dias	06/09/2021	09/09/2021
	Corrección de código	15 Dias	10/09/2021	30/09/2021
	Análisis de resultados finales	7 Dias	01/10/2021	11/10/2021
	Conclusión	9 Dias	11/10/2021	21/10/2021
5	**Memoria**	**171 Dias**	**08/06/2021**	**01/02/2022**
	Elaboración de memoria	171 Dias	08/06/2021	01/02/2022
6	**Presentacion**	**17 Dias**	**02/02/2022**	**24/02/2022**
	Elaboración de presentación	8 Dias	02/02/2022	11/02/2022
	Preparación de presentación	4 Dias	14/02/2022	17/02/2022
	Realización de presentación	1 Día	24/02/2022	24/02/2022

En la siguiente figura se muestra el diagrama de Gantt con las tareas a realizar situadas temporalmente.

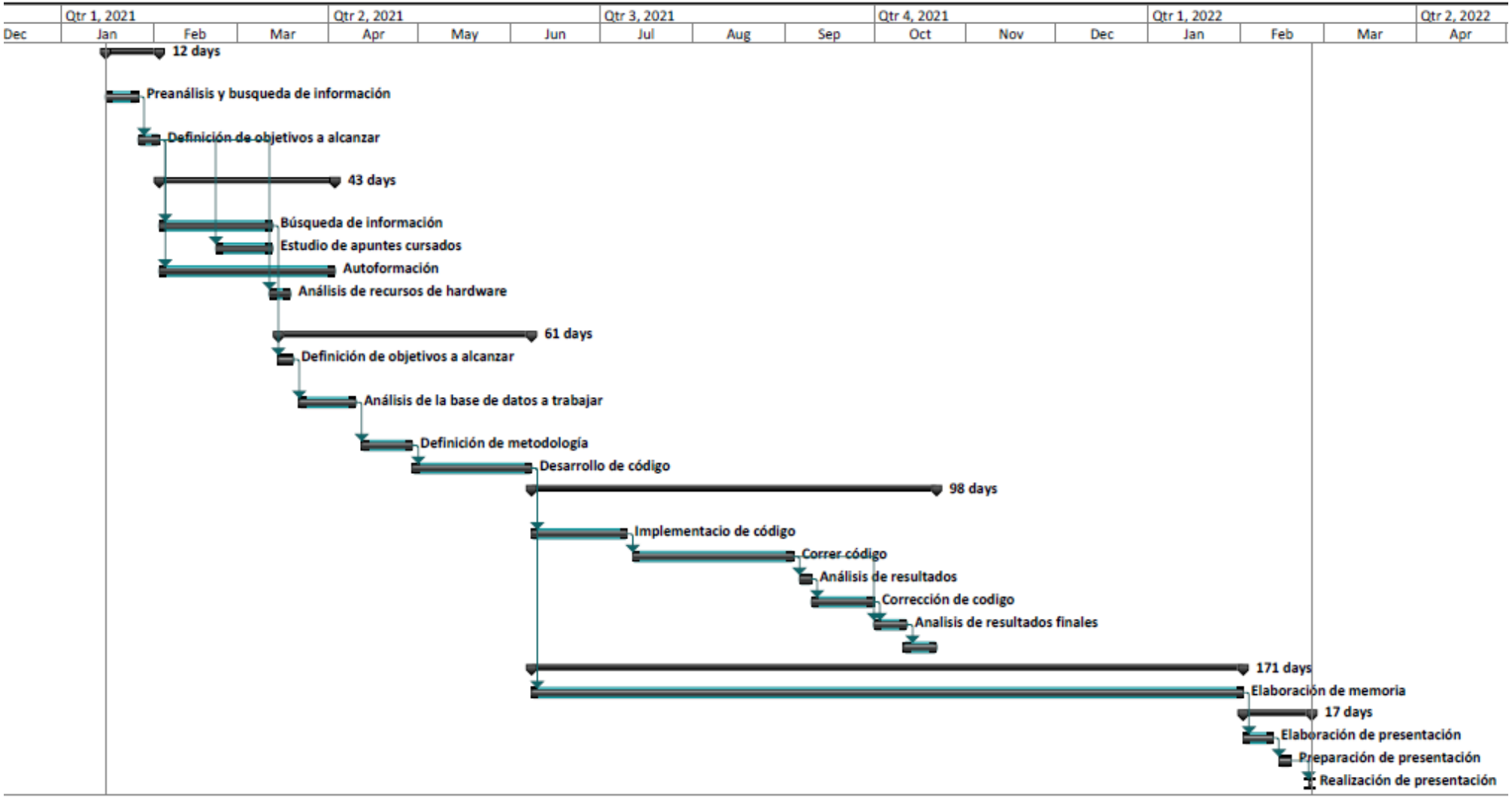


Figura 5.1: Diagrama de Gantt del proyecto

El presente proyecto dispone de una duración de 404 dias, con una fecha inicial del 16/01/21 y una finalización a fecha 24/02/22. Esta duración comprende desde la etapa inicial de pre análisis y búsqueda de informacion para asentar las bases del proyecto hasta la etapa final de presentación del mismo.

Tabla 5.3: Duración total de proyecto

Fecha de inicio	Fecha fin	Duración(días)
16/01/2021	24/02/2022	404

5.2. Recursos utilizados

Una vez establecida la planificación del proyecto con las tareas a realiza, en este apartado se va a analizar los recursos necesarios para llevar a cabo el proyecto. En la tabla que se muestra a continuación se van a ver los recursos y el coste económico asociado a ellos.

Tabla 5.4: Recurso utilizados para la realización del proyecto

Recurso	Descripción	Coste (€)
Ordenador	Ordenador portátil i5 - 7200 U, 8GB RAM. Realización de tareas básicas de análisis, programación básica, reuniones, realización de memoria, realización de presentación.	1840
Ordenador programación	Ordenador destinado a proceso de programación que requiere gran capacidad de memoria y procesamiento. Destinado al procesamiento del programa.	900 €
Webex	Software de ordenador dedicado al uso de tutorías online	Versión gratuita
Softwares	Softwares de programación, anaconda, spyder. Softwares dedicados a la creación de documentación de memoria, planning.	Versión gratuita
Suscripción a contenido científico	Licencia de socio para acceso a documentación científica	15€/año

5.3. Coste de horas de trabajo

En este apartado se va a detallar el desglose de horas empleadas en la realización del proyecto asi como una estimación de coste por unidad horaria empleada de trabajo.

En la primera tabla se muestra el desglose de horas realizadas por el alumno. En la segunda tabla se puede ver el desglose del coste asociado al trabajo del tutor.

Tabla 5.5: Coste de hora realizado por el alumno

Nº	Nombre tarea	Unidad	Cantidad	Coste/unidad	Total (€)
1	**Asignación de proyecto a realizar**	**h**	**48**	**15 €**	**720**
2	**Estudio y análisis de información**	**h**	**172**	**15 €**	**2580**
3	**Planteamiento de metodología**	**h**	**244**	**15 €**	**3660**
4	**Implementación de metodología**	**h**	**290**	**15 €**	**4350**
5	**Memoria**	**h**	**438**	**15 €**	**6570**
6	**Presentación**	**h**	**34**	**15 €**	**510**
				TOTAL	**18390**

Tabla 5.6: Coste de hora realizado por el tutor

Nº	Nombre tarea	Unidad	Cantidad	Coste/unidad	Total (€)
1	**Explicación de trabajo inicial a tratar**	**h**	**8**	**30 €**	**240**
2	**Tutorías seguimiento trabajo**	**h**	**22**	**30 €**	**660**
3	**Resolución de dudas**	**h**	**6**	**30 €**	**180**
4	**Corrección de código**	**h**	**9**	**30 €**	**270**
5	**Validación de metodología**	**h**	**10**	**30 €**	**300**
6	**Corrección de memoria**	**h**	**4**	**30 €**	**120**
				TOTAL	**1770**

Coste total.

Teniendo en cuenta el coste de los recursos necesarios para la realización de este proyecto asi como las hora invertidas en el mismo. El coste de realización de este proyecto asciendo a **22 915€.**

En la tabla que se muestra a continuación se ve el presupuesto desglosado.

Tabla 5.7: Coste de total de proyecto

Concepto	Coste(€)
Coste recursos	2755
Coste horas de trabajo	20160
TOTAL	22915

Capítulo 6
CONCLUSIÓN DE PROYECTO

6. CONCLUSIÓN DE PROYECTO.

Este capítulo recoge la conclusión del proyecto en visión general, haciendo uso de la técnica del *Machine Learning* expuesta. Este capítulo es complementación también de las conclusiones de la parte práctica que se han visto en el capítulo 4 de trabajo experimental.

En el presente trabajo Fin de Master realizado en el Master de Ingenieria Industrial de la escuela de Ingenieros Industriales de la Universidad de Valladolid se ha adquirido y ampliado las competencias en el ámbito de la inteligencia artificial, haciéndose uso del *Machine Learning* (ML) impartida en la asignatura de Tecnología de Control del departamento de Ingenieria de Sistemas y Automática, obteniendo resultados satisfactorios.

Tras el análisis de los resultados obtenidos en la realización de este trabajo fin de master, se han analizado gran cantidad de métodos y modelos, para la obtención de un resultado correcto.

Con la realización de este trabajo se ha alcanzado la comprensión de las técnicas de detección y diagnóstico de fallos en un proceso industrial, se han adquirido los conocimientos necesarios para la aplicación del *Deep Learning* y se ha logrado aplicar las técnicas de procesamiento de datos en entornos FDI&DI.

Es por todo esto que con la realización del presente trabajo queda de manifiesto el dominio que se ha adquirido con el uso de la técnica del *Deep Learning* (DL) para la detección y clasificación de fallos (FDI) en una planta de estación de aguas residuales (EDAR).

El uso de las redes neuronales artificiales (ANN) para la detección de fallos (FDI) es un útil bastante eficaz, en particular en este proyecto se han constatado los buenos resultados que da la técnica del *Deep Convolutional Neural Network* (DCNN) para este cometido.

Capítulo 7
LÍNEAS FUTURAS

7. LÍNEAS FUTURAS DE PROYECTO

Tras el análisis y los resultados obtenidos en la realización del proyecto podemos establecer una vía de trabajos futuros.

Las líneas futuras a realizar en este trabajo son; la optimización y explotación en la mejora de la detección de los fallos números 2,4 y 7. Pudiéndose explorar técnicas del *Machine Learning* alternativas a DCNN como se ha visto en este proyecto. Pudiéndose aplicar técnicas de redes artificiales u otras técnicas del *Machine Learning*. Por poner un ejemplo se podria aplicar la técnica de *clustering* del *Machine Learning* haciendo uso de un aprendizaje no supervisado. En este ref [20] se puede ver también los buenos resultados de esta técnica en la clasificación de anomalías.

La identificación del modo de funcionamiento en fallo o modo normal se ha conseguido al 100% con el modelo nº4. Una vez obtenido el resultado de modo normal o modo fallo, en la identificación del fallo podríamos aplicar un nuevo algoritmo para una detección profunda en el caso de que el programa nos diese los modos de fallo 2, 4 y 7.

En la tabla 7.1 se muestra los modelos 3,4 y 9 con los valores de detección para los modos de funcionamiento 2,4 y 7.

Tabla 7.1: Valor de precisión, exactitud y F1 score de modelo 3,4 y 9

Modo	Modelo 3	Modelo 4	Modelo 9
	Precisión		
0	1	1	1
1	1	1	1
2	0,92	0	1
3	1	1	1
4	0,39	0	0
5	0,95	0,95	0,95
6	0,95	0,95	0,95
7	0	0,32	0,38
8	0,95	1	1
9	0,86	0,9	1
Media	0,80	0,71	0,83
Media ponderada	0,91	0,87	0,92

Modo	Modelo 3	Modelo 4	Modelo 9
	Recall		
0	1	1	1
1	0,95	1	1
2	0,55	0	0,55
3	0,9	0,95	1
4	0,95	0	0
5	0,95	1	0,95
6	0,95	0,9	0,95
7	0	1	1
8	0,95	0,9	0,85
9	0,95	0,95	0,95
Media	0,82	0,77	0,83
Media ponderada	0,91	0,90	0,92

Modo	Modelo 3	Modelo 4	Modelo 9
	F1-scoore		
0	1	1	1
1	0,97	1	1
2	0,69	0	0,71
3	0,95	0,97	1
4	0,55	0	0
5	0,95	0,98	0,95
6	0,95	0,92	0,95
7	0	0,49	0,56
8	0,95	0,95	0,92
9	0,9	0,93	0,97
Media	0,79	0,72	0,81
Media ponderada	0,90	0,88	0,91

8. BIBLIOGRAFÍA

[1] G. I. Sainz Palmero, J. M. Benítez, M. J. De la Fuente Aparicio y A. Sánchez Fernández, «Linguistic OWA and two time-windows based fault identification in wide plants,» ELSEVIER, 2018.

[2] Á. S. Fernandez, «Métodos de detección y diagnóstico de fallos mediante aproximaciones distribuidas: Modelos, métodos y computación,» *TESIS DOCTORAL: Escuela de Doctorado de la Universidad de Valladolid,* 2020.

[3] L. O. P. P. C. D. a. J. Puigjaner, «Estrategias de modelado, simulación y optimización de procesos,» 2006.

[4] J. H. Wu, «Computers and Chemical Engineering,» *elsevier,* p. www.elsevier.com/locate/compchemeng, 2018.

[5] R. Tibshirani, «Regression Shrinkage and Selection via the lasso,» *Jounal of the Royal Statistical Society,* 1996.

[6] A. Y. Lu y M. Johnstone, «On consistency and Sparsity for Principal Components Analyses in High Dimensions,» *Jounal of the American Statistical Association,* 2009.

[7] M. R.Segal, «Machine Learning Benchmarks and Random Forest Regression,» *Center for Bioinformatics & Molecular Biostatistics,* 2004.

[8] V. Kotu y B. Deshpande, Data Science, ELSEVIER, 2018.

[9] F. Chollet, Deep Learning with Python, New York: Manning Publications, 2018.

[10] L. Rouhiainen, Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro, Alienta, 2018.

[11] S. Russell y P. Norving, Inteligencia artificial: un enfoque moderno, Madrid: Pearson Prentice Hall, 2004.

[12] J. Gomez Ines, «Implementación del modelo de red neuronal RBM,» Universidad de Valladolid, Valladolid, 2021.

[13] S. Raschka y V. Mirjalili, «Python Machine Learning: Machine Learning and Deep Learning with Python, sckit-learn, and TesonrFlow,» Packt Publishing, Birmingham, 2007.

[14] M. Minsky y S. Papert, Perceptrpms: An introduction to Computational Geometry, Estados Unidos: MIT Press, 1969.

[15] D. Rummerlhart, G. Hinton y R. Williams, «Leaning representations by back-propagating errors,» Institute for Cognitive Science, University of California, Philadelphia, 1986.

[16] W. Gil González, J. J. Mora Florez y S. M. Pérez Londoño, «Análisis del procesamiento de los datos,» Tecnura, 2014.

[17] D. Berrar, «Cross-validation. Encyclopedia of Bioinformatics and Computational,» Elsevier, Tokyo, 2019.

[18] P. Refaeilzadeh, T. Thang y H. Lui, «Cross-Validation,» Arizona State University, Arizona, 2008.

[19] F. Berzal, Redes neuronales & Deep Learning, Granada: EUG, Editorial Universidad de Granada, 2008.

[20] L. Borreguero Cortón, «Detección de anomalías en líneas ferroviarias mediante técnicas de Machine Learning,» Universidad de Valladolid, Valladolid, 2021.

[21] S. Chazallet, Pythin 3. Los fundamento del lenguaje - 2ª Edición., ENI Edifiones, 2016.

[22] R. D. R.-A. B.-G. Ivet Pérez, «El lenguaje de programación Python,» *Ciencias Holguín*, 2014.

[23] A. L. E. V.Alonso, 8th Manufacturing Enf¡gineering Soxiety International Conference, Procedia manufacturing, 2019.

[24] K. S. R. Taymanov, Metrology challenges of Industry 4.0, Journal of Physics: Conference Series, 2018;1065:072044, 2018.

[25] Deloitte Corporation, *Forces of change industry 4.0*, 2019.

9. ÍNDICE DE FIGURAS

10. ÍNDICE DE TABLAS

11. APÉNDICE A: PYTHON

El código de este proyecto ha sido realizado en lenguaje Python. A continuación se detalla una breve introducción y la historia de este lenguaje.

12.1. Introducción

Para la realización del código se ha hecho uso del lenguaje universal de programación PYTHON, ya que es el lenguaje más usado en el mundo, más flexible y polivalente contando con una gran comunidad de usuarios.

Una de las fortalezas que posee Python son las librerías standard con las que cuentan, ya que es capaz de cubrir todas las necesidades básicas de programación.

Otra ventaja del lenguaje Python es su capacidad de reutilización de código escrito en lenguaje C y C++.

En conclusión Python es un lenguaje de programación que permite realizar de forma cómoda y elegante el arte de la programación, estructurándolo.

Unas de las muchas razones por las que se ha escogido Python para la realización de este trabajo han sido [17].

- Multiplataforma: Se trata de lenguaje de programación el cual puede funcionar en cualquier tipo de sistema que integre su interpretador.
- Frameworks: Debido a su universalidad, hay uso de estructuras bases para usar como punto de partida.
- Libre y de código abierto.
- Calidad de sintaxis.
- Gran uso en el mundo laboral: La mayoría de empresas usas como lenguaje de programación el lenguaje Python.

12.2. Historia

Este lenguaje fue creado por un programador holandés a finales de los años 80 y cuyo nombre era Guido Van Rossum, cuando estaba trabajando en el sistema operativo Amoeba. En un primer momento fue diseñado como interfaz para Amoeba y siendo definitivamente sucesor del lenguaje ABC.

En 1991 es publicada la primera versión 0.9.0 por dicho programador, Guido Van Rossum.

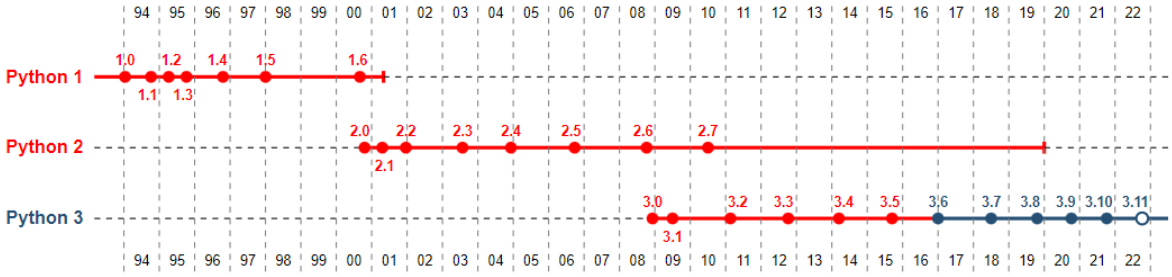
El 16 de octubre del año 2000 se lanza Python 2.0 con nuevas características, lo más importante de este nuevo lanzamiento es que este software comenzó a ser desarrollado por la comunidad, bajo la supervisión del programador Guido.

Posteriormente el 3 de diciembre del 2008 se actualiza la versión 3. Se trata de una versión mayor e incompatible en muchos aspectos con las versiones anteriores. Esta nueva versión trae grandes modificaciones respecto a la anterior, solucionando problemas que presentaba la versión 2.0.

Guindo Van Rossumn fue premiado con el Free Sorfteare Award en 2001 por sus trabajos con Python.

En la siguiente tabla se puede ver las diferentes versiones que ha sufrido Python a lo largo de la historia.

Tabla A.P: Evolución de la historia del Python.



En rojo podemos ver las versiones que se consideran obsoletas y en círculo blanco las futuras versiones.

En la actualidad estamos en la versión 3.10 [18].