UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Detección de perturbaciones en reactores nucleares utilizando redes neuronales

Trabajo Fin de Máster

Máster Universitario en Ingeniería Industrial

AUTOR/A: Lec'Hvien , Quentin

Tutor/a: Vidal Ferràndiz, Antoni

Cotutor/a: Verdú Martín, Gumersindo Jesús

CURSO ACADÉMICO: 2022/2023

AGRADECIMIENTOS

Un agradecimiento

a Toni Vidal y el señor Gumersindo Verdú por su consideración y paciencia

RESUMEN

El presente trabajo estudia las posibilidades de detección y localización de pequeñas anomalías de funcionamiento en un reactor nuclear de agua a presión. Mediante la monitorización de las medidas de los detectores de neutrones es posible la detección temprana de perturbaciones de una forma no invasiva. La detección temprana de anomalías da la posibilidad de realizar las acciones pertinentes antes que estas provoquen problemas de seguridad o repercutan en la disponibilidad de la planta.

El objetivo principal del TFM es el de diseñar una red neuronal que permita encontrar las perturbaciones en un modelo de reactor bidimensional a partir de los datos de flujo neutrónico simulados ante anomalías. En este sentido, se simulan casos de diferentes perturbaciones y se obtiene las lecturas del ruido neutrónico en los detectores, la fluctuación sobre la media. Estos datos permiten entrenar una red neuronal.

En primer lugar, se simularán los datos en el código FEMFFUSION. Luego, los datos permitirán construir una red neuronal para encontrar la localización de la perturbación. La red se construye de manera similar a la de la documentación que trata los problemas de localización mediante la librería KERAS.

Finalmente, se estudia en más detalle las diferentes redes que se pueden construir y se varían sus datos y estudiando su desempeño. Se concluye con los resultados de detección y se establecen los mejores parámetros para el funcionamiento del sistema.

Palabras Clave: Energía nuclear; ruido neutrónico; localización; redes neuronales; inteligencia artificial;

RESUM

El present treball estudia les possibilitats de detecció i localització d'anomalies de funcionament en un reactor nuclear d'aigua a pressió. Mitjançant el monitoratge de les mesures dels detectors de neutrons és possible la detecció precoç de pertorbacions d'una forma no invasiva. La detecció precoç d'anomalies dona la possibilitat de realitzar les accions pertinents abans que aquestes provoquen problemes de seguretat o repercutisquen en la disponibilitat de la planta.

L'objectiu principal del TFM és el de dissenyar una xarxa neuronal que permeta trobar les pertorbacions en un model de reactor bidimensional a partir de les dades de flux neutrónico simulats davant anomalies. En aquest sentit, se simulen casos de diferents pertorbacions i s'obté les lectures del soroll neutrònic en els detectors, la fluctuació sobre la mitjana. Aquestes dades permeten entrenar una xarxa neuronal.

En primer lloc, se simularán les dades en el codi FEMFFUSION. Després, les dades permetran construir una xarxa neuronal per a trobar la localització de la pertorbació. La xarxa es construeix de manera similar a la de la documentació que tracta els problemes de localització mitjançant la llibreria KERAS.

Finalment, s'estudia en més detalle les diferents xarxes que es poden construir i es varien les seues dades i estudiant el seu acompliment. Es conclou amb els resultats de detecció i s'estableixen els millors paràmetres pel funcionament del sistema.

Paraules Clau Clave: Energia nuclear; soroll neutrònic; localització; xarxes neuronals; intel·ligència artificial;

ABSTRACT

This work studies the possibilities of detection and localisation of small operating anomalies in a nuclear pressurised water reactor. By monitoring neutron detector measurements, early detection of disturbances is possible in a non-invasive way. Early detection of anomalies gives the possibility to take appropriate actions before they cause safety problems or impact on the availability of the plant.

The main objective of the TFM is to design a neural network to find disturbances in a two-dimensional reactor model from simulated neutron flux data in the presence of anomalies. In this sense, cases of different perturbations are simulated and the neutron noise readings in the detectors, the fluctuation above the mean, are obtained. These data are used to train a neural network.

First, the data will be simulated in the FEMFFUSION code. Then, the data will allow Building a neural network to find the location of the disturbance. The network is constructed in a similar way as in the documentation dealing with localisation problems using the KERAS library.

Finally, we study in more detail the different networks that can be built and vary their data and study their performance. We conclude with the detection results and establish the best parameters for the operation of the system.

Palabras Clave: Nuclear energy; neutron noise; localisation; neural network; Artificial inteligence;

ÍNDICE

1 Índice de la memoria

ILUSTRACIONES

TABLAS

CAPITULO 1: OBJETIVOS Y ESTRUCTURA

1. OBJETIVOS Y ESTRUCTURA DEL DOCUMENTO

1.1 Objetivo del documento

Las centrales nucleares se equipan de multitud de detectores que permiten recoger datos sobre el flujo de neutrones dentro de su núcleo para la seguridad da la central y sus distintas operaciones. Durante su funcionamiento en condiciones normales, los detectores vuelven valores estáticos además de fluctuaciones de la señal. Estas fluctuaciones pueden venir de distintas fuentes, las comunes son:

- Vibraciones mecánicas de los elementos combustibles o de la estructura del núcleo
- Perturbaciones en la transferencia de calor del fluido
- Variación de temperatura o densidad del fluido o del combustible nuclear

Uno de los parámetros importante que se tiene que observar en esta situación es el "Ruido de neutrones". Esta es la fluctuación del flujo de neutrones alrededor de un valor medio, observado en condiciones de operación estática. Estas fluctuaciones son importantes ya que transportan información relacionada con los fenómenos del núcleo que pueden ocurrir como consecuencia de las fuentes de perturbaciones descritas previamente.

El origen de las perturbaciones y sus consecuencias sobre el flujo de neutrones puede ser simuladas por ordenador. Cada perturbación se propaga en el dominio de frecuencia en todo el volumen del núcleo del reactor. Por eso, el comportamiento del flujo se puede calcular con las ecuaciones neutrónicas en el emplazamiento de un detector debido a una perturbación en cualquier lugar del reactor. En centrales reales, la situación es la inversa, se debe deducir la posición original de la perturbación y el tipo de perturbación partiendo de la información que tenemos con los detectores de flujos de neutrones.

Invertir la función de transferencia es un problema no lineal, por lo que se propone en este Trabajo Fin de Máster de utilizar redes neuronales para encontrar la posición original de las perturbaciones. Se detalle objetivos específicos:

- Diseñar una red neuronal funcional
- Maximizar la precisión de la red neuronal
- Optimizar el tiempo de cálculo de la red.

1.2 Estructura del documento

Este Trabajo de Fin de Máster se estructura de la siguiente manera:

En primer lugar, se haría introduce la importancia de la energía nuclear en el mundo y en España además de enumerar los distintos tipos de reactores en centrales nucleares y sus especificaciones.

En segundo lugar, se desarrolla los elementos necesarios sobre inteligencia artificial y las redes neuronales en general.

Estas partes permiten tener los conceptos básicos y claves para el entendimiento de las partes siguientes que son, la obtención de las ecuaciones que permiten generar datos sobre el ruido neutrónico en un reactor y, el diseño de la red neuronal para encontrar la fuente de las perturbaciones gracias a los datos generados anteriormente.

Después, se estudia más en detalle la red neuronal. Se trata de encontrar la combinación de parámetros que permite obtener una precisión máxima y un tiempo de cálculo mínimo.

Finalmente, se presentan las conclusiones principales del trabajo.

CAPÍTULO 2: INTRODUCCIÓN A LA ENERGÍA NUCLEAR

2 INTRODUCCIÓN A LA ENERGÍA NUCLEAR

2.1 Principio de la energía nuclear

Una reacción nuclear es una interacción entre un núcleo atómico y otra partícula (partícula elemental, núcleo atómico o radiación gamma) que provoca un reordenamiento nuclear.

Estas reacciones son tanto más fáciles cuanto que conducen a configuraciones más estables de los átomos. La diferencia de energía (correspondiente al defecto de masa) es la energía liberada por la reacción. Esta transformación de la masa en energía (según la famosa fórmula $E = M\,c^2$) se utiliza en las reacciones de fisión y fusión nuclear.

Ahorra, se detalle el funcionamiento de la fisión (Ilustración 1).

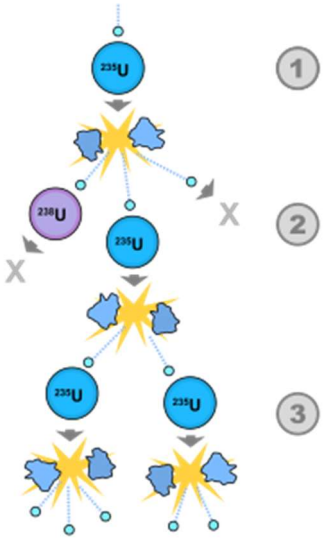

Ilustración 1- Ejemplo de fisión nuclear: reacción en cadena con n uranio 235 (Energía nuclear, s.f.).

Cuando un neutrón golpea el núcleo de ciertos isótopos pesados, existe la probabilidad de que el núcleo golpeado se divida en dos núcleos más ligeros. Esta reacción, que lleva el nombre de fisión nuclear, se traduce en un desprendimiento de energía muy importante (del orden de 200 MeV por evento, en comparación con las energías de las reacciones químicas, del orden del eV).

Esta fisión se acompaña de la emisión de varios neutrones que, en ciertas condiciones, chocan con otros núcleos y provocan así una reacción en cadena. En un reactor nuclear, esta reacción en cadena se desarrolla en condiciones estables, a velocidad lenta y controlada.

La energía liberada por esta reacción en cadena es la que permite calentar el fluido dentro del núcleo y generar vapor. Este principio es el principio de funcionamiento

de todas las centrales nucleares. Las centrales 2 tipos de reactores más utilizados en el mundo son:

- **Pressurized Water Reactors (PWR):** El agua a presion (por lo tanto, en estado líquido) es a la vez el caloportador y el moderador. El combustible utilizado es uranio enriquecido. Este tipo de reactor representa el 55% de los reactores en el mundo
- **Boiling Water Reactor (BWR):** El agua es también caloportador, pero ya no está presurizada. A presión atmosférica ambiente, se vuelve hirviendo. El combustible utilizado es uranio enriquecido. Este tipo de reactor representa 22% de los reactores en el mundo.

Este trabajo se basa en el funcionamiento de una central con un reactor tipo PWR. Se detalle su funcionamiento en la parte siguiente.

2.2 Funcionamiento de un reactor tipo PWR

El reactor de agua presurizada, PWR para Pressurized water Reactor en inglés, es el sistema de reactores nucleares más extendido en el mundo: En enero de 2021, dos tercios de los 444 reactores nucleares en funcionamiento en el mundo son de tecnología REP, así como los buques y submarinos nucleares. Se compuesta de 3 circuitos de agua independientes.

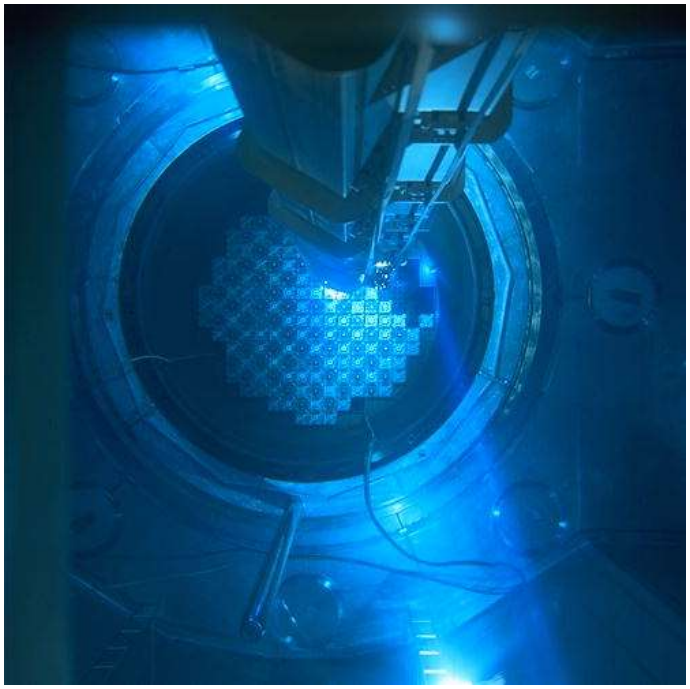

Ilustración 2- Reactor de central tipo PWR (Fonctionnement central PWR, s.f.).

La Ilustración 2 es una foto del combustible nuclear fresco cargado en el núcleo de un reactor en la central nuclear de Civaux en Francia (Poitou-Charentes). La tapa del tanque se abrió, y se puede ver en la parte inferior el mosaico de conjuntos de combustible ya en su lugar, mientras que un robot introduce un nuevo conjunto. Todas las manipulaciones se realizan bajo el agua.

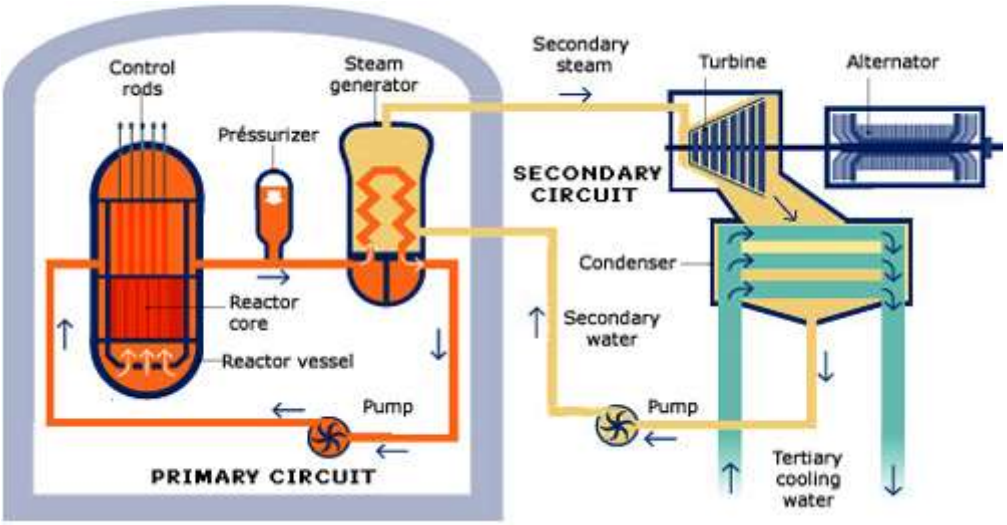

Ilustración 3 - Esquema circulación del fluido en una central PWR (Fonctionnement central PWR, s.f.).

La **ilustración 3** muestra el funcionamiento de una centra tipo PWR. Los elementos combustibles que contienen uranio forman el núcleo de un reactor de agua a presión (PWR). La reacción en cadena de la fisión permite que el agua se calienta en contacto con las barras de combustible a una alta temperatura en un sistema cerrado, el circuito primario. Esta agua vaporiza el agua de otro circuito, el (circuito secundario) a través de intercambiadores de calor llamados generadores de vapor. El vapor secundario impulsa una turbina que impulsa un alternador que produce electricidad. El vapor que sale de la turbina se enfría y se transforma en agua en un condensador antes de volver al generador de vapor. El propio condensador se enfría por el agua de un tercer circuito. Las bombas hacen circular el agua en estos diversos circuitos. Un presurizador mantiene el agua del circuito primario a alta presión y evita que hierva. Las varillas de control permiten controlar la reacción en cadena.

La temperatura del agua primaria que sale del núcleo del reactor es de aproximadamente 300 °C. Esta agua pasa a través de un intercambiador de calor, llamado "generador de vapor", donde se enfría vaporizando el agua de un circuito secundario. En el generador de vapor, el agua secundaria enfría los tubos donde circula el agua primaria antes de regresar al recipiente del reactor. A la salida del generador de vapor, la presión de vapor de agua secundaria es de 70 atmósferas.

El circuito secundario es un circuito cerrado. El vapor se envía a través de una turbina. La turbina acciona un alternador acoplado. El alternador produce electricidad para ser enviada a la red nacional a alta tensión. El vapor secundario se condensa en la salida de la turbina, antes de ser reciclado en los generadores de vapor.

El núcleo del reactor, el circuito primario y los generadores de vapor se encuentran en un edificio de contención sellado por una pared de hormigón simple o doble. Este edificio es de unos 50 m de diámetro y 60 m de altura. Se despresuriza para evitar fugas al exterior. Esta contención proporciona una tercera barrera protectora.

El vapor que sale de la salida de la turbina se condensa por un alto flujo de agua que circula a alta velocidad en un tercer circuito de refrigeración. El vapor condensado regresa a los generadores. El agua "terciaria", que a su vez necesita ser enfriada, se envía a grandes torres de refrigeración (una por reactor). Estas torres son la parte más visible de una planta nuclear. Emiten columnas de vapor de agua.

Para compensar el agua perdida por evaporación en las torres, se necesita agua externa. Las centrales nucleares están situadas cerca del mar o de los ríos, cuyas aguas no están en contacto con materiales radiactivos debido a las diversas barreras de contención.

En el caso de olas de calor con una disminución en el nivel del agua, se favorecen las plantas de energía nuclear ubicadas en la costa para producir electricidad. Las centrales eléctricas situadas a lo largo de los ríos están equipadas con torres de refrigeración que utilizan la atmósfera como fuente de frío.

2.3 Importancia de la energía nuclear en España

Uno de los principios fundamental de la producción de electricidad es, lo que se produce debe ser igual a lo que se consume. Es decir que no se puede sobrecargar la red en producción sin dañarla. En contrario, si no se produce suficiente electricidad, el país tendría zonas sin electricidad. Ahora se pregunta, como se hacen los países para conseguir la demanda de electricidad

En la Ilustración 4 (Treiner, 2017), tenemos un ejemplo de producción de energía diario según el tipo de generación el 18 de octubre de 2017 en Francia y España:

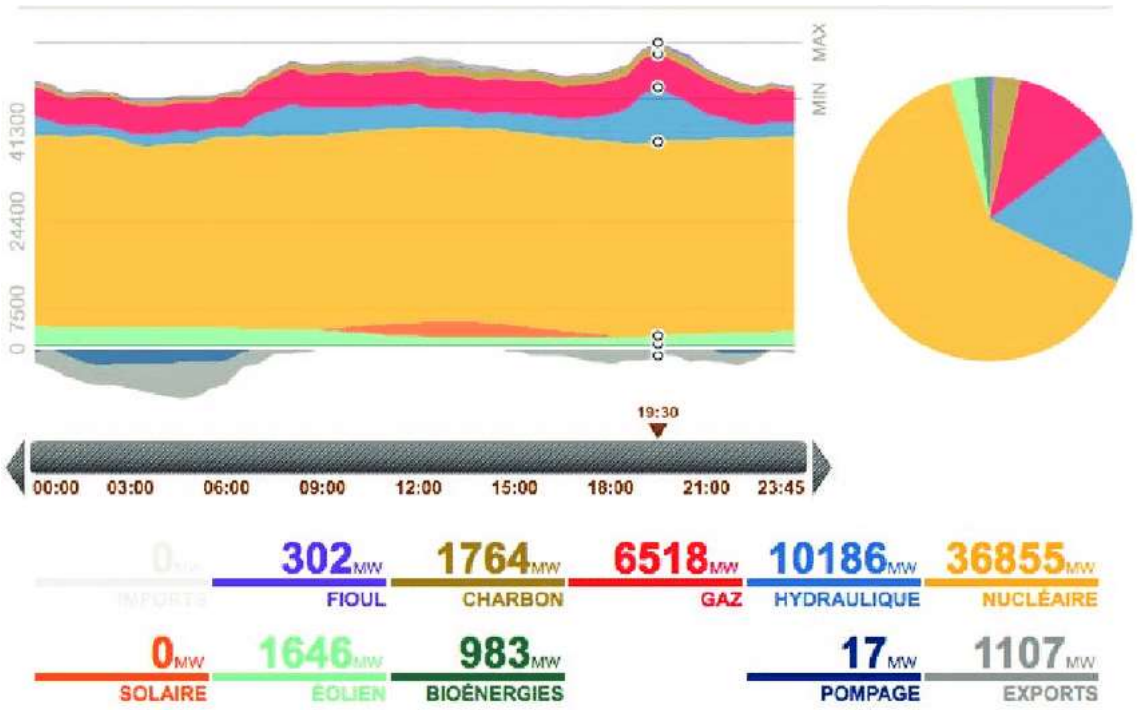

Ilustración 4- Consumo de electricidad por un día en Francia.

Traducción de la Ilustración 4 en la Tabla 1:

Frances	Esapañol
Fioul	Combustible
Charbon	Carbón
Gaz	Gas
Hydraulique	Hidráulico
Nucléaire	Nuclear
Solaire	Solar
Eolien	Eoliano
Bioénergies	Bioenergía
Pompage	Bombeo
Exports	Exportaciones

Tabla 1- traducción Frances/español.

Vemos que la parte nuclear tiene una producción bastante constante a lo largo del día. Sin embargo, la energía hidráulica varia a lo largo del día. Entonces, para conseguir la demanda, los países producen electricidad con una base bastante constante gracias por (en este ejemplo) la energía nuclear, la de gas y de carbón. Los picos de consumo se manejan con otras fuentes de energía, en este ejemplo, la energía hidráulica.

El caso de España se presenta de la manera siguiente (Ilustración 5 e Ilustración 6)para el día 08/10/2022:

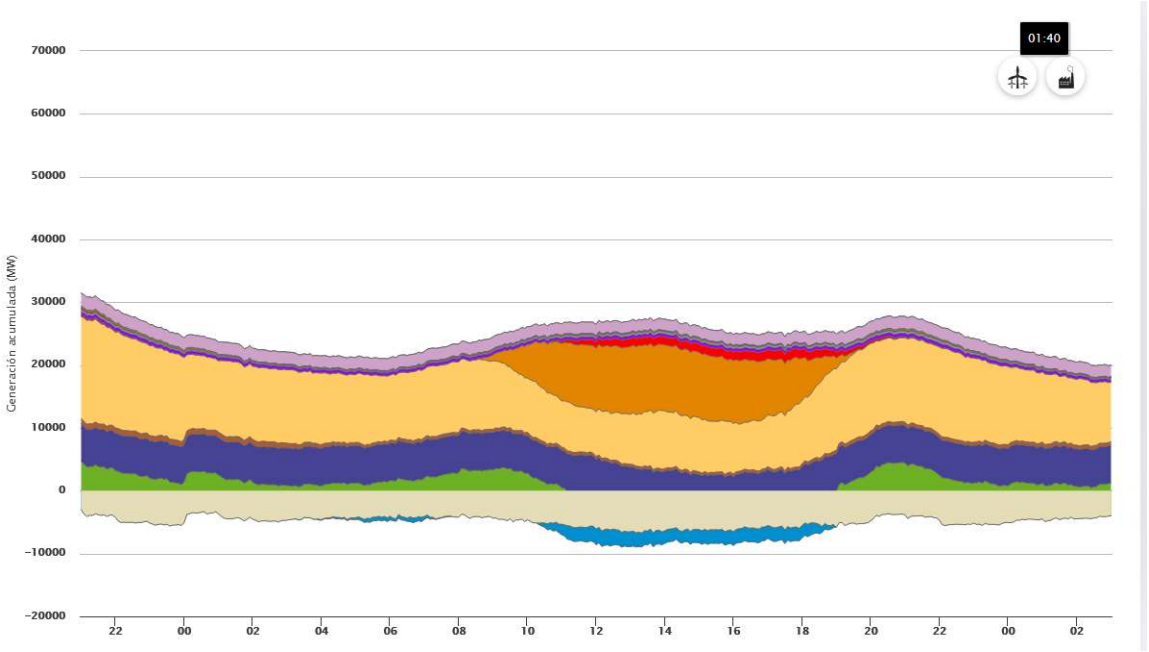

Ilustración 5- consumación de energía eléctrica en España (Nacional - Seguimiento de la demanda de energía eléctrica, 2022).

Estructura de generación acumulado progresivo (MW) a las 01:40 - 09/10/2022

Cogeneración y residuos	1910	7,51(%)
Generación auxiliar		(%)
Turbina de vapor	122	0,48(%)
Turbina de gas	36	0,14(%)
Motores diésel	194	0,76(%)
Térmica renovable	484	1,9(%)
Solar térmica	1	0(%)
Solar fotovoltaica	4	0,02(%)
Ciclo combinado	10568	41,56(%)
Carbón	588	2,31(%)
Nuclear	5972	23,49(%)
Eólica	4720	18,56(%)
Hidráulica	829	3,26(%)
Intercambios int	-4353	0(%)

Ilustración 6- producción de electricidad por cada tipo de generación (Nacional - Seguimiento de la demanda de energía eléctrica, 2022).

De la misma manera que el caso de Francia, España maneja la producción con una base constante con la energía nuclear (25%), térmica renovable (1,55%), cogeneración y residuos (7.41%) o motores Diesel (0.77%). La variación de la demanda se maneja por la mayoría con las energías solar. y de ciclo combinado.

La producción de electricidad en España se caracteriza por una preponderancia de centrales térmicas con combustibles fósiles, en mayoridad con carbón y combustible. Los combustibles totalizan 40.5% de la producción en 2019. Las energías renovables se acercan de este valor con 38.1%. En 202, la pandemia de Covid-19 permitió aumentar esta producción a 44.4% contra 33% por las energías fósiles.

En España, la proporción de producción de electricidad por energías renovables representa un 37.8% en 2019 (20,4% energía eólica, 9.8% hidráulica, 5.5% solar y 2.1% biomasa) contra 40.5% de energías fósiles y 21.4% de energía nuclear. La producción hidráulica varia fuertemente de un año al otro, en función de las precipitaciones y la cobertura de la demanda por las otras energías renovables que aumenta en función de las subvenciones. Por otra parte, las seis centrales nucleares españolas han visto disminuir progresivamente su parte en la producción eléctrica debido a la congelación del parque nuclear debido a la moratoria nuclear de 1983, mientras que la producción total continuaba su crecimiento en las últimas décadas: Las importaciones aumentaron del 35,7% en 1990 al 21,4% en 2019 y al 22,2% en 2020.

En la Tabla 2 (Électricité en Espagne, 2022) y la Ilustración 7, se pude ver la evolución de las partes de los distintos tipos de producción de energía en España:

Source	1990	%	2000	%	2010	%	2018	2019	% 2019	var. 2019/1990	2020
Charbon	60,66	*39,9*	80,86	*36,0*	26,32	*8,7*	38,72	13,98	***5,1 %***	**-77 %**	5,98
Pétrole	8,60	*5,7*	22,58	*10,1*	16,56	*5,5*	14,50	12,88	***4,7 %***	**+50 %**	11,14
Gaz naturel	1,51	*1,0*	20,18	*9,0*	94,85	*31,5*	58,00	83,70	***30,6 %***	**+5447 %**	69,39
Total fossiles	**70,78**	*46,6*	**123,61**	*55,1*	**137,74**	*45,7*	**111,22**	**110,57**	***40,5 %***	**+56 %**	**86,50**
Nucléaire	**54,27**	*35,7*	**62,21**	*27,7*	**61,99**	*20,6*	**55,77**	**58,35**	***21,4 %***	**+8 %**	**58,28**
Hydraulique	26,18	*17,2*	31,81	*14,2*	45,51	*15,1*	36,80	26,87	***9,8 %***	**+3 %**	33,89
Biomasse	0,46	*0,3*	1,16	*0,5*	3,36	*1,1*	5,16	4,80	***1,8 %***	**+939 %**	4,96
Déchets renouv.	0,08	*0,05*	0,33	*0,15*	0,66	*0,2*	0,75	0,77	***0,3 %***	**+863 %**	0,69
Éolien	0,014	*0,01*	4,73	*2,1*	44,27	*14,7*	50,90	55,65	***20,4 %***	**ns**	56,27
Solaire PV	0,006	*0,004*	0,02	*0,01*	6,42	*2,1*	7,88	9,42	***3,4 %***	**ns**	15,55
Solaire thermodyn.					0,76	*0,3*	4,87	5,68	***2,1 %***	**ns**	4,99
Total EnR	**26,75**	*17,6*	**38,04**	*16,9*	**100,98**	*33,5*	**106,35**	**103,35**	***37,8 %***	**+286 %**	**116,48**
Déchets non renouv.	0,13	*0,1*	0,61	*0,3*	0,66	*0,2*	1,02	0,98	***0,4 %***	**+658 %**	1,03
Total	**151,92**	*100*	**224,47**	*100*	**301,53**	*100*	**274,45**	**273,26**	***100 %***	**+80 %**	**262,30**
Source des données : Agence internationale de l'énergie[24]											

Tabla 2- producción de electricidad en España por fuente (TWh) (Électricité en Espagne, 2022).

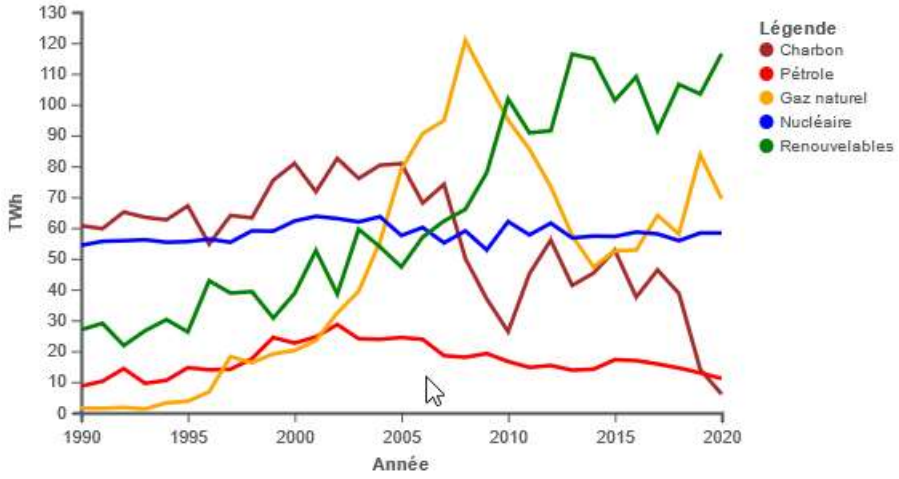

Ilustración 7- Grafico de la producción de electricidad en España por distintas fuentes (Électricité en Espagne, 2022).

El parque nuclear español se compuesta de 7 reactores en funcionamiento en 5 sitios distintos como le muestra la Tabla 1tabla siguiente:

Central	Reactor	Año de puesta en funcionamiento	Expiración de la licencia	Potencia
Almaraz	*Almaraz I* — PWR	1983	2021	977 MW
	Almaraz II — PWR	1984	2023	980 MW
Ascó	*Ascó I* — PWR	1984	2023	1032,5 MW
	Ascó II — PWR	1986	2025	1027,2 MW
Cofrentes	BWR-6	1984	2034	1092 MW
Santa María de Garoña	BWR-3	1970	2013[32]	466 MW
Vandellós II	PWR	1988	2027	1087,1 MW
Trillo I	PWR	1988	2028	1.066 MW
Total	8 reactores	—	—	**7.727,8 MW**

Table 1- Reactores nucleares de España (Energía nuclear en España, 2013)

2.4 Las centrales nucleares en el mundo

El parque nuclear mundial cuenta 444 reactores nucleares operacionales situados en 32 países según la Agencia Internacional de energía atómica (AIEA). Los reactores nucleares representan entre 11% y 16% de la producción de electricidad mundial.

Los Estados Unidos es el primer explotador nuclear con 92 reactores en 54 centrales, luego Francia cuenta 56 reactores en 18 centrales y en tercera posición; China cuenta 55 reactores en 15 centrales.

En uno de sus escenarios, la AIEA establece que la capacidad nuclear instalada en el mundo podría ser multiplicada por 2 hasta 2050 y alcanzar 792GW (contra 393 GW en fin 2020).

CAPÍTULO 3: REDES NEURONALES

3 REDES NEURONALES

3.1 ¿Qué es la inteligencia Artificial (IA)?

La Inteligencia Artificial (IA) en su forma más genérica puede definirse como la calidad de la inteligencia que se introducen en las máquinas. Un ejemplo sería una lavadora que puede decidir sobre la cantidad correcta de agua a utilizar y sobre el requerido tiempo para remojar, lavar, e hilar; es decir, toma una decisión cuando se proporcionan entradas específicas y por lo tanto funciona en una más inteligente manera. Del mismo modo, un cajero automático podría hacer una llamada en el desembolso de la cantidad que con la combinación correcta de notas disponibles en la máquina. Esto la inteligencia es técnicamente inducida en la máquina de una manera artificial, por lo tanto, el nombre IA.

Los algoritmos de machine learning aprenden de forma autónoma a realizar una tarea o realizar predicciones a partir de datos y mejoran su rendimiento con el tiempo. Una vez entrenado, el algoritmo puede encontrar patrones en nuevos datos.

Ejemplos de Machine Learning (ML) podría ser un sistema que podría predecir si un estudiante fallará o aprobará una prueba un examen. Se basaría sobre los datos históricos del estudiante para estudiar los "*pattern"* de fallos o de aprobación. El sistema no tendrá todas las reglas que pueden decir si un estudiante fallará, sino que el sistema aprende los "*patterns"* de fallos del estudiante.

Sin embargo, aunque los sistemas de Machine Learning permite resolver ciertos tipos de problemas, fallan a resolver problemas como distinguir una imagen de gato y de perro o reconocer una que viene de un hombre y una de una mujer.

Para resolver los problemas que los sistemas de ML fallan a resolver, se inspira del proceso biológico del cerebro humano, que se compone de miles de millones de neuronas conectadas y orquestadas para adaptarse y aprender: es el Deep Learning (DL), una red de neuronas con capas múltiples que aprende con datos de casos anteriores y permite resolver los problemas que un sistema de ML no puede.

Hoy en día, podemos aprovechar DL para casos como reconocer un animal en una foto o hacer una estimación del precio de una casa utilizando ML y esperar superar nuestros logros anteriores, siempre que haya un excedente de datos. Esta comprensión ha llevado a distinguir el orden de los campos basado en los datos. Una nueva regla

general se estableció: ML no sería capaz de mejorar el rendimiento con aumento de los datos de formación después de un determinado umbral, mientras que DL fue capaz de aprovechar los datos excedentes de manera más eficaz para mejorar el rendimiento. El mismo fue cierto hace unos años en el debate entre los modelos estadísticos y ML. La Ilustración 8 es una ilustración para representar la idea general del rendimiento del modelo con el tamaño de los datos para los tres campos antes mencionados.

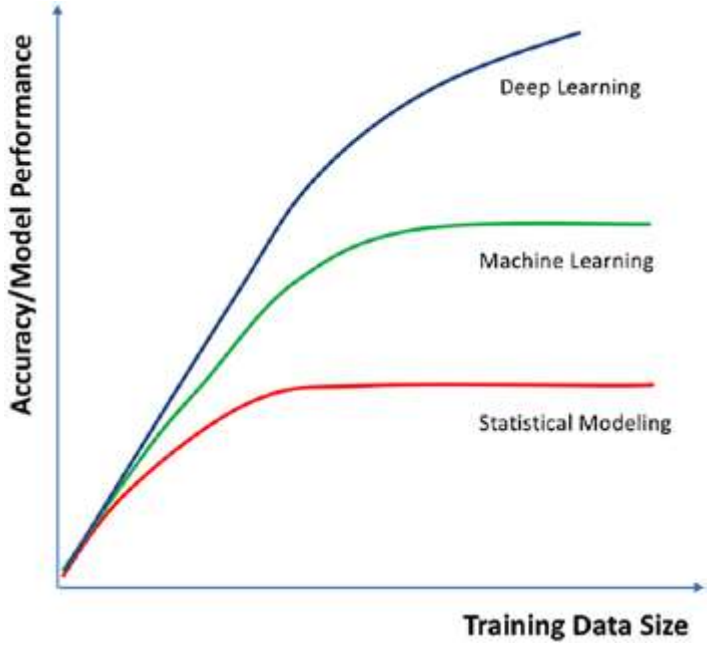

Ilustración 8- Curva de aprendizaje (Moolayil, 2019).

Ahora, si revisamos la definición formal, probablemente puedas hacer un mejor sentido de la declaración de que el subcampo de AI de ML se inspira en los aspectos biológicos de un cerebro humano. Podemos simplificar los tres campos utilizando un diagrama de Venn simple, como se muestra en la **Ilustración 9**.

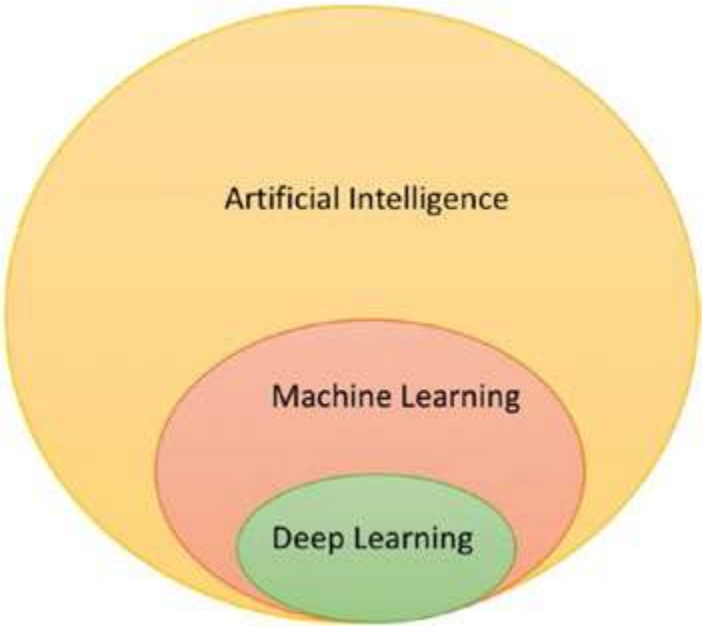

Ilustración 9 - Campos de tema de Inteligencia Artificial (Moolayil, 2019).

Poniendo todo junto, podemos decir que la IA es el campo de inducir inteligencia en una máquina o sistema artificialmente, con o sin explícita programación. ML es un subcampo en AI donde la inteligencia se induce sin programación explícita. Por último, DL es un campo dentro de ML donde la inteligencia es inducida en sistemas sin programación explícita utilizando algoritmos que han sido inspirados por el funcionamiento biológico del cerebro humano.

3.2 Principio de redes neuronales

En su forma más básica, los modelos DL se diseñan utilizando la arquitectura de red neuronal. Una red neuronal es una organización jerárquica de neuronas (similar a las neuronas en el cerebro) con conexiones a otras neuronas. Estas neuronas pasan un mensaje o señal a otras neuronas basado en la entrada recibida y forman una red compleja que aprende con algún mecanismo de retroalimentación.

La Ilustración 10 es una representación simplista de una red neuronal básica.

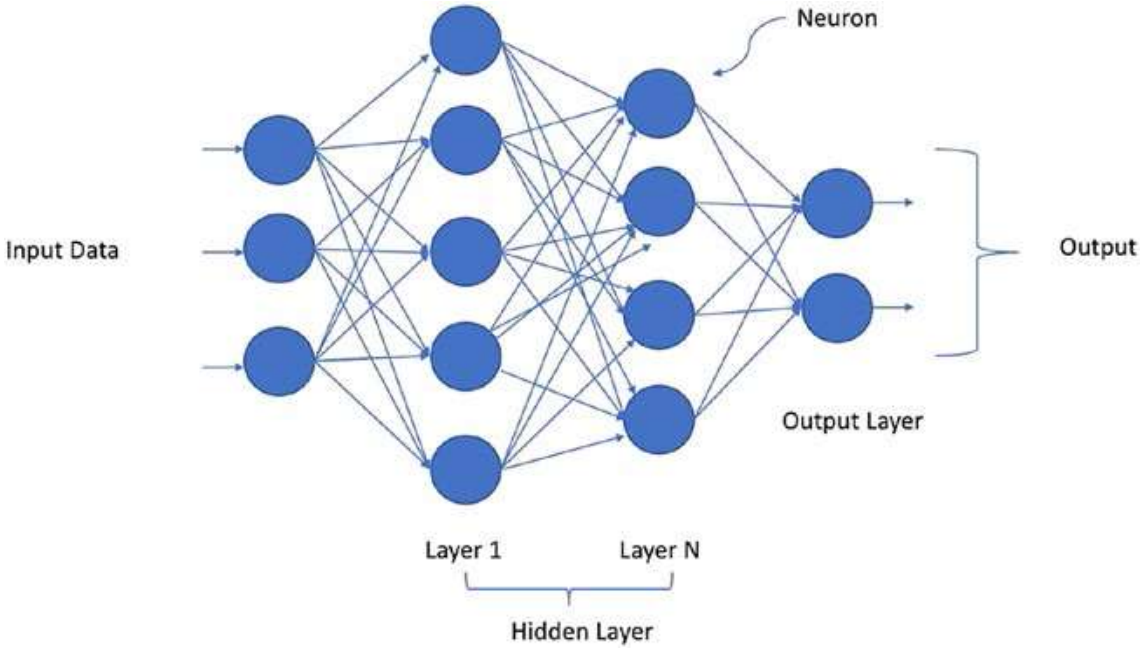

Ilustración 10 - Esquema arquitectura de una red neuronal (Moolayil, 2019).

La Ilustración 10 es una representación simplista de una red neuronal básica. Como se puede ver en la Ilustración 10 figura, los datos de entrada son consumidos por las neuronas en la primera capa oculta, que luego proporciona una salida a la siguiente capa y así sucesivamente, lo que finalmente resulta en la salida final. Cada capa puede tener una o varias neuronas, y cada una de ellas calculará una pequeña función (por ejemplo, función de activación). La conexión entre dos neuronas de capas sucesivas tendría un peso asociado. El peso define la influencia de la entrada a la salida para la neurona siguiente y eventual para la salida final total. En una red neuronal, los pesos iniciales serían aleatorios durante el entrenamiento del modelo, pero estos pesos se actualizan iterativamente para aprender a predecir una salida correcta. Descomponiendo la red, podemos definir pocos bloques de construcción lógicos como neurona, capa, peso, entrada, salida, una función de activación dentro de la neurona para calcular un proceso de aprendizaje, y así sucesivamente.

La red neuronal mostrada en la Ilustración 2Ilustración 10 trata de imitar proceso del cerebro humano utilizando un enfoque matemático. La entrada es consumida por las neuronas en la primera capa y se calcula una función de activación dentro de cada neurona. Basado en una regla simple, envía una salida a la siguiente neurona, similar a las desviaciones aprendidas por el cerebro humano. Cuanto mayor sea la producción de una neurona, mayor será la importancia de esa dimensión de entrada. Estas dimensiones se combinan en la siguiente capa para formar nuevas dimensiones adicionales, que probablemente no tengan sentido. Pero el sistema lo aprende

intuitivamente. El proceso, cuando se multiplica varias veces, desarrolla una red compleja con varias conexiones.

Ahora que se entiende la estructura de la red neuronal, entendamos cómo sucede el aprendizaje. Cuando proporcionamos los datos de entrada a la estructura definida, la salida final sería una predicción, que podría ser correcta o incorrecta. Sobre la base de la salida, si proporcionamos retroalimentación a la red para adaptarse mejor mediante el uso de algunos medios para hacer una mejor predicción, el sistema aprende actualizando el peso de las conexiones. Para lograr el proceso de proporcionar retroalimentación y definir el siguiente paso para hacer cambios de la manera correcta, utilizamos un hermoso algoritmo matemático llamado "backpropagation" Iterar el proceso varias veces paso a paso, con más y más datos, ayuda a la red a actualizar los pesos apropiadamente para crear un sistema donde pueda tomar una decisión para predecir la salida basada en las reglas que se ha creado a través de los pesos y conexiones.

El nombre "redes neuronales profundas" evolucionó a partir del uso de muchas más capas ocultas, lo que lo convierte en una red "profunda" para aprender patrones más complejos. Los casos de éxito de DL solo han surgido en los últimos años porque el proceso de formación de una red es computacionalmente pesado y necesita grandes cantidades de datos. Los experimentos finalmente vieron la luz del día solo cuando la computadora y el almacenamiento de datos se hicieron más disponibles y asequibles.

CAPITULO 4: SIMULACIÓN DE PERTURBACION NEUTRÓNICA

4 SIMULACIÓN DE PERTURBACION NEUTRÓNICA

En este apartado, vamos a describir como obtenemos la ecuación de la difusión neutrónica y la ecuación del ruido neutrónico en el dominio de la frecuencia.

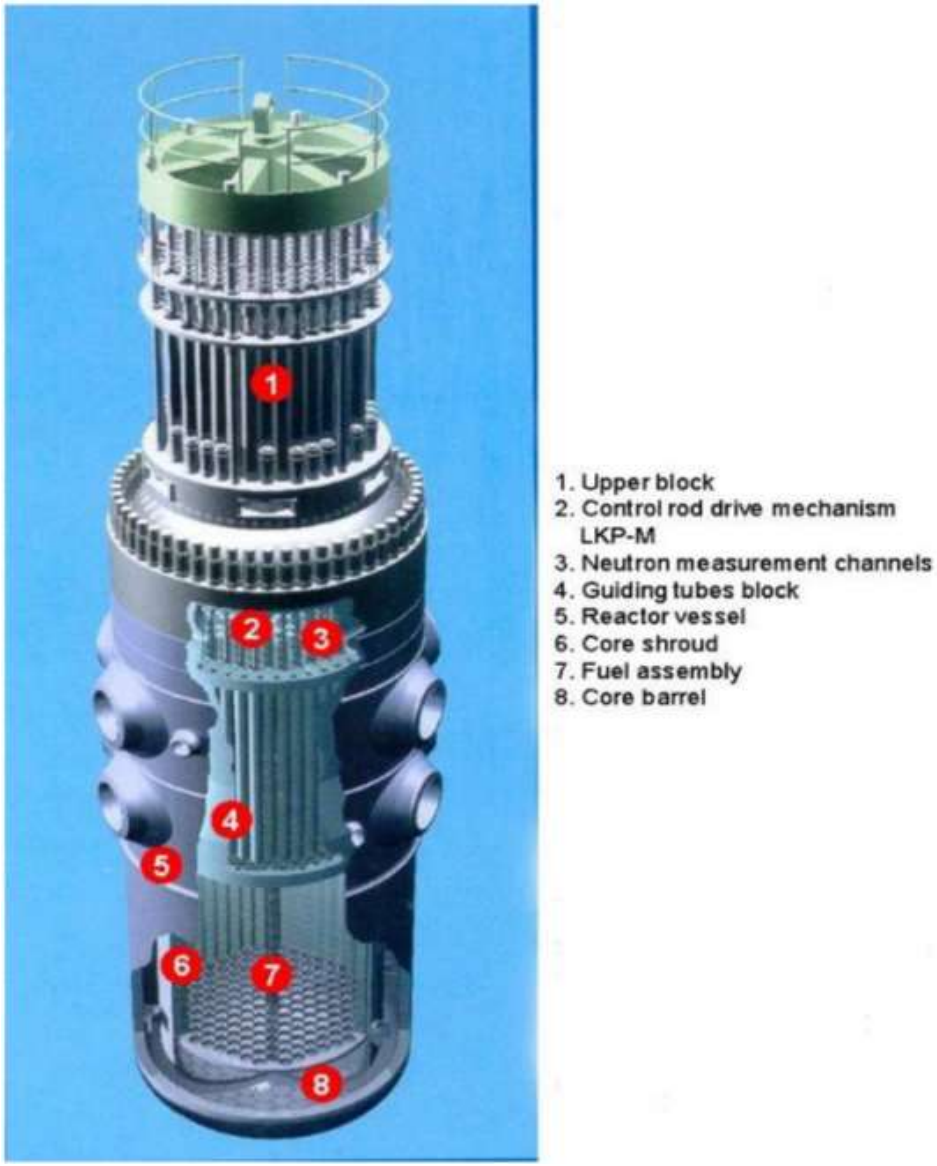

Illustration 11- General characteristics of the VVER-1000/320 reactor (Vidal-Ferràndiz, 2022)

4.1 Metodología de medida

Los datos de ruido de neutrones se miden y recopilan con el móvil, distribuido internamente sistema de prueba de medición (DMTS), a partir del sistema estándar de monitorización de vibraciones del reactor de la planta de diagnóstico (RVMS), junto con registros de datos tecnológicos. Los sensores de diagnóstico RVMS de cada unidad incluir 4 acelerómetros en la brida de la cabeza del reactor, 12 cámaras de ionización colocadas en tres planos verticales y en dos niveles horizontales, y más de 256 detectores de neutrones de potencia propia (SPNDs) a través de todo el núcleo en cuatro alturas axiales en difusión radial no uniforme.

En este estudio, se concentra sobre una perturbación de desplazamiento en una de las celdas.

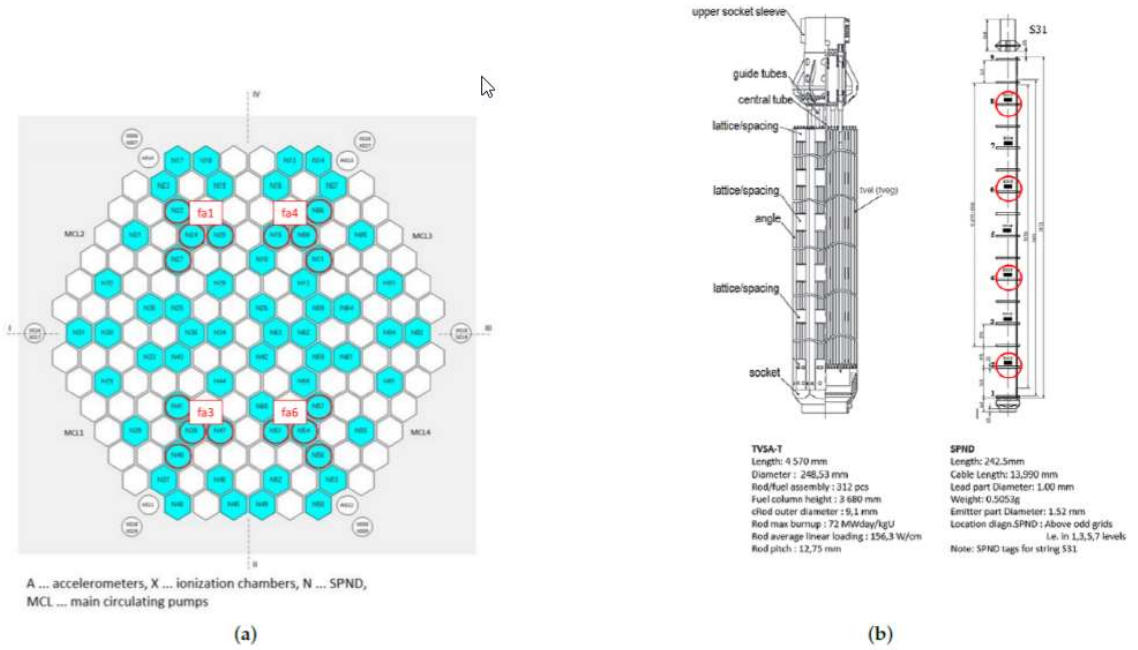

Illustration 12- SPND positions in the core of the VVER-1000/320 reactor with data from the U1C09 cycle. (a) Radial detector positions with marked fa1, fa3, fa4 and fa6 sets. (b) Vertical detector positions in the TVSA-T fuel assembly (Vidal-Ferràndiz, 2022)

Todos los datos de ruido de neutrones se acortan a la longitud uniforme de 720.000 muestras en orden para evitar transitorios indeseables al inicio de las mediciones de la planta. Por lo tanto, hay existen intervalos uniformes de 12 minutos para los pasos de procesamiento descritos a continuación. Corriente continua (CC) se eliminan los componentes de todos los datos de ruido de neutrones de estos conjuntos. Todas las cámaras de ionización y los datos SPND se normalizan a la parte DC de la señal respectiva. Figura 3 exhibe los espectrogramas conjuntos de tiempo-frecuencia (JTFSs) de un SPND, seleccionados del conjunto de configuración fa1 como una vista típica en el dominio de tiempo de frecuencia. Los datos del inicio del ciclo (BOC) U1C09 se

adquirieron en 19 configuraciones establece en octubre de 2010, durante las pruebas físicas de la instrumentación de neutrones y bajo estrictas condiciones de explotación.

4.2 Datos de simulación

Antes de desplegar, los algoritmos de aprendizaje automático necesitan ser alimentados con datos de entrenamiento, es decir, datos en los que se supone una perturbación conocida y el correspondiente neutrón inducido ruido en la ubicación de los detectores se estima. Esta sección se ocupa de la generación de estos juegos de entrenamiento para reactores hexagonales La generación de los conjuntos de datos de formación es realizada con el código FEMFFUSION en su modo de dominio de frecuencia.

4.2.1 El código de FEMFFUSION

FEMFFUSION (A. Vidal-Ferràndiz, 2021) es un código neutrónico C++ de código abierto que resuelve la ecuación de transporte de neutrones multigrupo utilizando la aproximación de difusión o la aproximación SPN. El código utiliza el método de elementos finitos Galerkin continuo para poder tratar con cualquier tipo de geometría y dimensión del problema (problemas 1D, 2D y 3D). Funciona sobre la biblioteca deal.II, que proporciona soporte y avances en el método de elementos finitos.

Las principales características de FEMFFUSION son:

- Software de código abierto (publicado bajo los términos de la GNU GPL versión 3)
- Uso del FEM para resolver las ecuaciones de difusión de neutrones multigrupo (o SPn).
- Uso de la técnica libre de matriz para mantener demandas razonables de memoria.
- Posibilidad de utilizar e implementar una variedad de solucionadores de valores propios y precondicionadores.
- Válido para todo tipo de geometrías, rectangulares, hexagonales, pin-nivel y no estructuradas.
- Posibilidad de importar redes no estructuradas desde GMsh.
- Capacidad para resolver problemas en 1D, 2D y 3D.
- Resolver el flujo directo o adjunto y varios pares propios, si se solicita.
- Problemas dependientes del tiempo como: movimientos de barras de control, problemas de ruido, secciones transversales personalizadas dependientes del tiempo...
- Análisis de frecuencia de problemas de ruido.
- La salida proporciona el keff, el mapa de la potencia media de neutrones por conjunto y cada uno de los flujos. También estándar. Se proporcionan archivos

vtk entre otros formatos de salida. Para los problemas dependientes del tiempo, la potencia de neutrones y los flujos de neutrones se proporcionan cada paso de tiempo.

- Interactúa con bibliotecas de código abierto de alta calidad: deal.II, PETSc, SLEPc, Sundials...
- Interfaz sencilla con herramientas de trazado y post-procesamiento (Matlab, ParaView, Matplotlib...).
- Bien documentado y fácil de extender a los problemas relacionados.

En realidad, FEMFFUSION construye las matrices y que proyecta la ecuación de transporte/difusión de neutrones multigrupo como un problema de valor propio basado en matriz:

$$L\,\phi = \frac{1}{k_{eff}} M\phi$$

Se espera que estas matrices sean escasas, ya que son el resultado de la discretización del operador de transporte diferencial utilizando el método de los elementos finitos, sobre una determinada red espacial.

4.2.2 Ecuación de difusión del ruido de neutrones

FEMFFUSION resuelve la ecuación de difusión de ruido de neutrones en el grupo de multi energía aproximación. La ecuación de difusión de neutrones dependiente del tiempo se puede escribir como en Ecuaciones (1) y (2) a continuación [23]

$$[\,\mathrm{v}^{-1}\,]\frac{\partial \phi}{\partial \mathrm{t}} + \mathcal{L}\phi = (1-\beta)\mathrm{M}\phi + \sum_{\mathrm{p}=1}^{\mathrm{Np}} \lambda_{\mathrm{p}} \chi \mathrm{C}_{\mathrm{p}} \tag{1}$$

$$\frac{\partial \mathrm{C}_{\mathrm{p}}}{\partial \mathrm{t}} = \beta_{\mathrm{p}} \left[\mathrm{v}\sum\nolimits_{f}\right]^{\mathrm{T}} \phi - \lambda_{\mathrm{p}} \mathrm{C}_{\mathrm{p}} \tag{2}$$

En la teoría de dos grupos sin upscattering, las matrices v 1, L,M, nSf , c y F son definido como en las ecuaciones (3) y (4)

$$[v^{-1}] = \begin{bmatrix} \frac{1}{v_1} & 0 \\ 0 & \frac{1}{v_2} \end{bmatrix}, \qquad \boldsymbol{\mathcal{L}} = \begin{bmatrix} -\boldsymbol{\nabla}.(\boldsymbol{D_1}\boldsymbol{\nabla}) + \Sigma_{a1} + \Sigma_{a2} & \mathbf{0} \\ -\Sigma_{12} & \Sigma_{a2} \end{bmatrix} \tag{3}$$

$$M = \begin{bmatrix} v\,\Sigma_{f1} & v\,\Sigma_{\mathrm{f2}} \\ 0 & 0 \end{bmatrix}, \quad v\sum\nolimits_{\boldsymbol{f}} = \begin{bmatrix} v\Sigma_{\mathrm{f1}} \\ v\Sigma_{\mathrm{f2}} \end{bmatrix} \tag{4}$$

La principal incógnita de la ecuación de transporte de neutrones es la que depende del espacio y el tiempo flujo de neutrones en su separación habitual en los grupos de energía rápida y térmica $\phi = [\phi_1(\vec{r},t),\ \phi_2(\vec{r},t)]^T$, y la concentración precursora de neutrones $C_p(\vec{r},t)$ para cada precursor de neutrones grupo p. La fracción total de neutrones retardados es $\beta = \sum_{p=1}^{N_p}\beta_p$ $\sum_{a1}$ y $\sum_{a2}$ son las secciones transversales de absorción rápida y térmica que cuantifican la probabilidad de que un neutrón rápido o térmico sea absorbido por un núcleo por centímetro de viaje de neutrones, respectivamente. De la misma manera, $\sum_{f1}$ y $\sum_{f2}$ son las probabilidades de una reacción de fisión producida por un neutrón rápido o térmico por centímetro de viaje de neutrones. v es el número medio de neutrones liberados por fisión, y $\sum_{12}$ es la probabilidad de una interacción de dispersión por centímetro de viaje rápido de neutrones. Estas secciones están determinadas por los materiales del reactor. Todas las demás cantidades tienen su significado habitual en el ámbito de la ingeniería nuclear [23]. La teoría del ruido de neutrones de primer orden [24] divide todos los términos dependientes del tiempo de las ecuaciones (1) y (2), $X(\vec{r},t)$, en sus valores medios (o estáticos) $X_0(\vec{r})$ y su fluctuación alrededor de la media $\delta X(\vec{r},t)$ (Ecuación 5)

$$X(\vec{r},t) = X_0(\vec{r}) + \delta X(\vec{r},t) \tag{5}$$

Esta separación significa (i) Los operadores de neutrones, $\boldsymbol{\mathcal{L}}$ y M; (ii) Los materiales se cruzan

secciones $\sum_a$, $\sum_f$, $\sum_{12}$; (iii) La concentración de precursores de neutrones retardados; (iv) C_p; (v) El flujo de neutrones, ϕ

Se hace las hipótesis siguientes:

- Las fluctuaciones son pequeñas

$$|\delta X(\vec{r},t)| \ll X_0(\vec{r}),\ \ \forall(\vec{r},t)$$

- Se supone que el transitorio es estacionario

$$\langle \delta X(\vec{r},t)\rangle = 0, \qquad \forall(\vec{r},t)$$

- Los términos de segundo orden se descuidan

$$\delta X(\vec{r},t).\,\delta Y(\vec{r},t) \approx 0$$

Aplicando la separación del ruido de neutrones de la Ecuación (5) en las Ecuaciones (1) y (2), eliminando los términos de segundo orden y realizando una transformación de Fourier, la ecuación de ruido de neutrones del dominio de frecuencia se convierte (Ecuación (6))

$$A\delta\phi = B\phi_0 \tag{6}$$

En la aproximación habitual de dos grupos.

$$A = \begin{bmatrix} \frac{i\omega}{\sum_{a1} + \sum_{a2}} - \gamma v\Sigma_{f1} & -\gamma v\Sigma_{f2} \\ -\Sigma_{12} & \frac{i\omega}{\sum_{a2}} \end{bmatrix}, \quad B = \begin{bmatrix} -\delta\Sigma_{a1} & \gamma\delta\Sigma_{f2} \\ \delta\Sigma_{12} & -\delta\Sigma_{a2} \end{bmatrix} \tag{7}$$

Donde

$$\gamma = (1 - \beta) + \sum_{p=1}^{N_p} \frac{\lambda_p \beta_p}{i\omega + \lambda_p} \tag{8}$$

Se puede ver que la ecuación de ruido de neutrones (6) es una ecuación no homogénea con cantidades de valor complejo. La aplicación de la discretización continua de elementos finitos de Galerkin [25] conduce a un sistema algebraico de ecuaciones con valores complejos y la estructura de bloques de la ecuación (7) a continuación

$$\begin{pmatrix} \widetilde{A_{11}} & \widetilde{A_{12}} \\ \widetilde{A_{21}} & \widetilde{A_{22}} \end{pmatrix} \begin{pmatrix} \delta\widetilde{\phi_1} \\ \delta\widetilde{\phi_2} \end{pmatrix} = \begin{pmatrix} \widetilde{B_{11}} & \widetilde{B_{12}} \\ \widetilde{B_{21}} & \widetilde{B_{22}} \end{pmatrix} \begin{pmatrix} \widetilde{\phi_{0,1}} \\ \widetilde{\phi_{0,2}} \end{pmatrix}, \tag{9}$$

donde$\delta\widetilde{\phi_1}\ y\ \delta\widetilde{\phi_2}$ son los vectores algebraicos de pesos asociados con el ruido de neutrones rápidos y térmicos en el dominio de frecuencia. Este complejo sistema de valores tiene que ser resuelto después del problema de estado estacionario que calcula los vectores algebraicos de pesos, asociados con el flujo de neutrones estáticos rápidos y térmicos, $\widetilde{\phi_{0,1}}\ y\ \widetilde{\phi_{0,2}}$, respectivamente. El problema de valor propio estático relacionado debe resolverse con la misma discretización espacial que el ruido de neutrones del dominio de frecuencia para obtener resultados coherentes. Este sistema se transforma en un sistema equivalente de ecuaciones con valores reales, y el sistema escaso de la ecuación (7) se resuelve mediante un método de gradiente biconjugado estabilizado [26], con una descomposición de LU incompleta [27] empleada como preacondicionador.

Cada simulación calcula el ruido térmico relativo de neutrones $\frac{d\phi_2}{\phi_{0,2}}$ en cada posición del detector.

El escenario que se está considerando es una perturbación similar a la de Dirac en la célula hexagonal, expresada directamente como una perturbación de las secciones

transversales de absorción macroscópica. Este escenario es particularmente importante ya que se puede utilizar para localizar un tipo genérico de perturbación que no se ajusta a ninguna categoría especial. La perturbación insertada en cada simulación se establece en $\delta \sum_{a1}(c, \omega) = 0.1$ y $\delta \sum_{a2}(c, \omega) = 0.1$, donde c es la celda hexagonal y w es la frecuencia angular de la perturbación. Se consideran tres frecuencias de perturbación: (i) 0,1 Hz; (ii) 1 Hz; (iii) 10 Hz.

CAPITULO 5: DISEÑO DE LA RED NEURONAL

5 DISEÑO DE LA RED NEURONAL

5.1 Localización con redes neuronales

Sobre la base de los datos disponibles (simulados y reales), el objetivo del modelo de aprendizaje automático es realizar una tarea de identificación: es decir, identificar si se producen perturbaciones con respecto a cada conjunto de combustible. Para este propósito, el modelo elegido se entrena en primer lugar en todos los datos simulados disponibles, donde se sabe de antemano en qué conjunto de combustible se produce la perturbación, formando así un problema de clasificación supervisado. Una vez finalizada la fase de entrenamiento, el rendimiento del modelo se valida en las mediciones de la planta en un esfuerzo por identificar posibles lugares de perturbación.

5.2 Creación del dataset

El dataset es el conjunto de data de entrada y de salida que permite entrenar la red neuronal. Es la primera etapa que se necesita hacer antes de construir la red neuronal. Esta etapa es crucial en la construcción de la red porque fija el formato de datos de entrada y de salida.

5.2.1 Datos Brutos

En este estudio, se estudia en 2D las perturbaciones. Es decir que el reactor se compuesta de capas apiladas. Se estudiará solamente 1 de estas capas compuesta por 257 celdas. Las celdas numeran empezado por la esquina superior izquierda (0) y acabando por la esquina inferior derecha.

En este estudio, los datos de entrada se entregan de la manera siguiente:

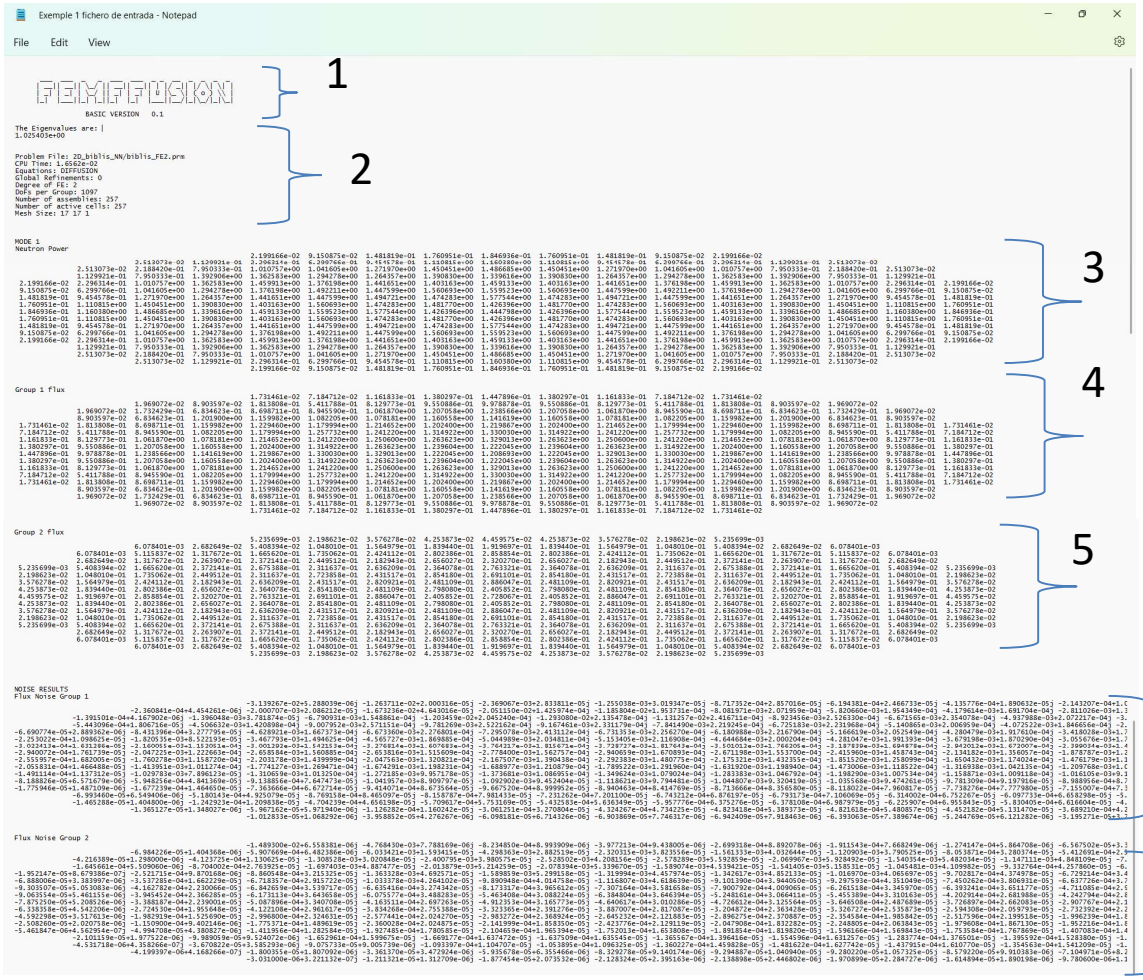

Ilustración 13- Ejemplo de datos de entrada.

Abajo se describe la imagen:

1- Título : FEMFUSSION : programa de cálculo del flux de neutrones y de perturbaciones
2- Parámetros de la simulación
3- Modelo 1, potencia neutrónica: 1 es la medida de la cantidad de reacciones de fisión hay en cada punto del reactor. Es proporcional a la potencia térmica del reactor en cada punto del reactor
4- Groupo 1 Flujo neutrónico rápido, es decir, neutrones de más de 1MeV
5- Groupo 2 Flujo neutrónico térmico, es decir, neutrones de menos de 1MeV
6- Resultados de ruido, Ruido de flujo del groupo 1 : Valor de la perturbación del flux 1 por cada celda del reactor en forma de números complejos (ai+b)
7- Resultados de ruido, Group 2 : Valor de la perturbación del flux 2 por cada celda del reactor en forma de números complejos (ai+b)

En este fichero, la perturbación viene de la celda 1. Entonces, hay 257 ficheros. Uno por cada ubicación de las perturbaciones posibles del reactor; es decir, 257.

5.2.2 Datos de entrada de la red

El principio del estudio es el de encontrar fuentes de perturbaciones en reactores nucleares mediante los datos de los detectores dentro del reactor. Entonces, se tiene que coger la información de flux (en nuestro caso el 2) y de su perturbación correspondiente en cada una de las celdas de los detectores. Por ejemplo, como primer caso, se coge las informaciones de los detectores en las celdas 25,35,223 y 233 como se ve en la imagen siguiente:

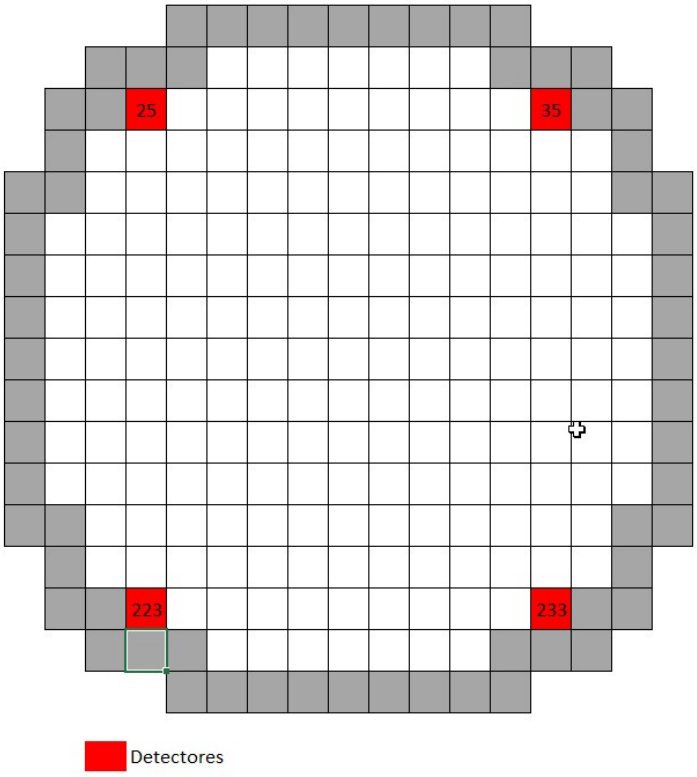

Ilustración 14- Posición de detectores en el reactor.

En este caso, en cada emplazamiento de detectores, se obtiene un valor del flux (que es un valor constante) ϕ_i y un nombre imaginario que corresponde a la señal de la perturbación

$$\delta\phi = a + i\,b \tag{10}$$

Una especificación de las redes neuronal es que soportan solamente datos entre 0 y 1. Este proceso por lo que se producen estos datos se llama normalización. Existe distintas maneras de normalizar los datos de entrada, pero se compara las estrategias de normalización en otra parte.

En cada detector, el valor de la perturbación es un número complejo (ai+b) que se transforma en un módulo y en una fase.

$$M_i = \sqrt{a^2 + b^2} \tag{11}$$

$$\theta = \text{arctg}\left(\frac{b}{a}\right) \tag{12}$$

Sabemos que el valor del módulo de la perturbación siempre esta menor que el valor del flux y la fase menor que pi. Entonces, se divide Mi por ϕi y θ por pi.

Obtenemos:

$$M_i = \frac{\sqrt{a^2 + b^2}}{\phi_i} \tag{13}$$

$$\theta_i = \frac{\text{arctg}\left(\frac{b}{a}\right)}{\pi} \tag{14}$$

Al final, por cada detector, tenemos un valor de Modulo y de Fase.

La forma de datos de entrada seria:

$$[M_{25}, M_{35}, M_{223}, M_{233}, \theta_{25}, \theta_{35}, \theta_{223}, \theta_{233}]$$

No se olvidará que se tiene 1 conjunto de datos por cada perturbación. Entonces, al final, se tiene solamente 257 datos de entrada por la red.

Se sabe que utilizamos detectores para medir el flux y su perturbación, pero ellos no son perfectos y no miden el mismo valor por una perturbación idéntica. Se modeliza esta imperfección por una ley normal de media $\mu = M_i$ y de varianza $\sigma^2 = 5\%\ \mu$.. Con esta modelización, se puede crear cuantos datos que queramos. En este primer ejemplo, multiplicaremos por 50 el número de situaciones en cada celda.

Al final obtenemos 50x257 =12 850 datos de entrada.

En una última etapa, se crea el dataset de validación de la red de manera análoga a la del dataset de aprendizaje. Su tamaño es de 257 datos lo que corresponde a un 2% del dataset de aprendizaje.

Este datase será utilizado para el entrenamiento de la red neuronal que se diseña en una parte ulterior.

5.3 Arquitectura de la red

A continuación, se describirá la arquitectura de una red neuronal posible para resolver el problema del estudio diseñada con la biblioteca de Python Keras (Keras API reference, s.f.). Es decir que puede haber diferentes arquitecturas para encontrar una

fuente de perturbación en el reactor. Cada solución tendría su propio tiempo de computación que se optimizara en otra parte. Abajo, se puede ver un ejemplo de arquitectura de redes neuronales. Se compuesta de 3 neuronas por la entrada, 2 capas ocultas (compuesta de neuronas también) y la capa de salida con 2 neuronas.

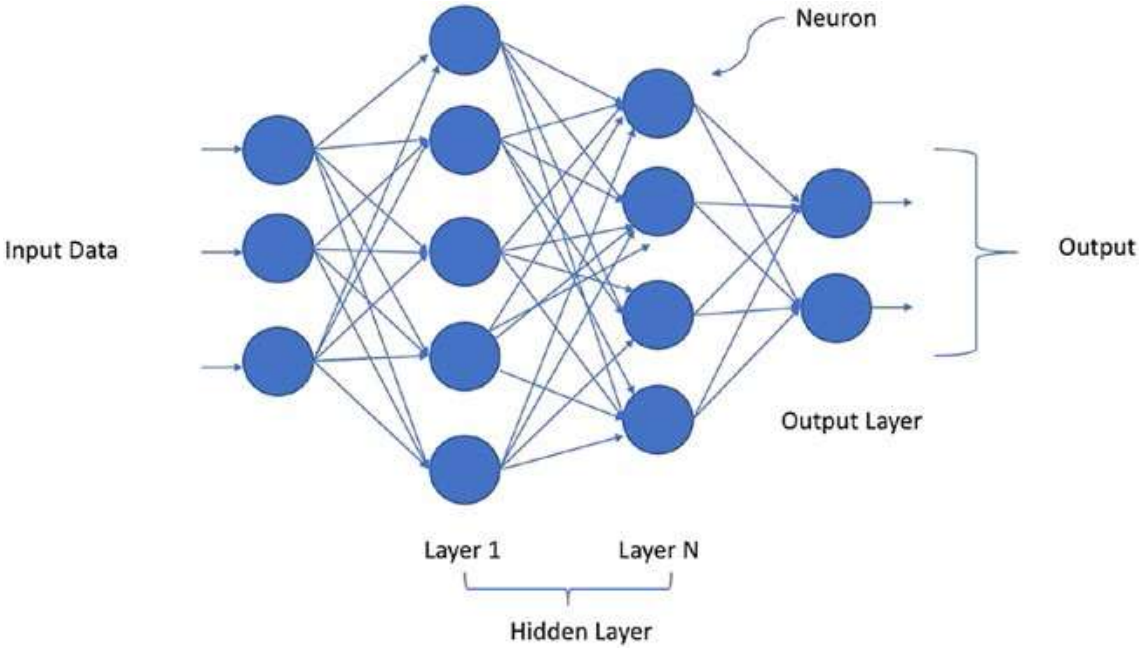

Ilustración 15- Esquema de una red neuronal (Moolayil, 2019).

La arquitectura utilizada en este estudio será diferente de la de la Ilustración 15 por tener un número de parámetros de entrada y de salida diferente.

En la entrada, la red tiene un número de neuronas igual al número de parámetros de entrada. En este caso, hay el doble de neuronas que de detectores porque cada detector ofrece 2 parámetros; el module y la fase del número complejo que mide.

En las capas ocultas, la red tiene una capa para el primer ejemplo compuesta por 64 neurona. Este número podría variar cuando se haga la fase de optimización de tiempo del entrenamiento de la red. A notar que es más conveniente para la red que el número de neuronas dentro de las capas oscuras sea una potencia de 2 (8,16,32,64,…) porque permite un tiempo de computación más corto según los expertos de Inteligencia Artificial.

La salida de la red será la probabilidad con la de la celda en que se ubica la perturbación. Por eso, esta capa (output layer) tiene 257 neuronas numerados de 0 a 256.

5.3.1 Optimizador

Los optimizadores son algoritmos utilizados para cambiar los atributos de la red neuronal, como los pesos y la tasa de aprendizaje para reducir el error de la red. Los

optimizadores se utilizan para resolver problemas de optimización minimizando una función de coste.

Cómo debe cambiar sus pesos o tasas de aprendizaje de su red neuronal para reducir las pérdidas se define por los optimizadores que se utilizan.

El peso se inicializa utilizando algunas estrategias de inicialización y se actualiza con cada época de acuerdo con la ecuación de actualización.

$$w_{new} = w_{old} - lr + (\nabla_w L)_{w_{old}} \tag{15}$$

La ecuación anterior es la ecuación de actualización utilizando qué pesos se actualizan para alcanzar el resultado más preciso. El mejor resultado se puede lograr utilizando algunas estrategias de optimización o algoritmos llamados optimizadores. En este parte, se utilizará el optimizador Adam de Keras (Keras API reference, s.f.).

El optimizador Adam

Adam, o Adaptive Moment Estimator, es una técnica de optimización que tiene una tasa de aprendizaje adaptativa para cada parámetro o peso. Adam no solo mantiene una media de gradientes cuadrados, sino que también mantiene una media de gradientes pasados. (Diederik P. Kingma, 2015)

Que la tasa de decaimiento de la media de gradientes $m_{ij}^{(t)}$ y la media del cuadrado de gradientes $v_{ij}^{(t)}$ para cada peso wij ser β1 y β2 respectivamente. Además, que ɳ sea el factor de tasa de aprendizaje constante. Entonces, las reglas de actualización para Adán son las siguientes:

$$m_{ij}^{(t)} = \beta_1 m_{ij}^{(t)} + \frac{(1-\beta_1)\partial C^{(t)}}{\partial w_{ij}} \tag{16}$$

$$v_{ij}^{(t)} = \beta_2 v_{ij}^{(t-1)} + (1-\beta_2)\left(\frac{\partial C^{(t)}}{\partial w_{ij}}\right)^2 \tag{17}$$

La media normalizada de los gradientes $\widehat{m_{ij}^{(t)}}$ y la media de los gradientes cuadrados $\widehat{v_{ij}^{(t)}}$ se calculan como sigue:

$$\widehat{m_{ij}^{(t)}} = \frac{m_{ij}^{(t)}}{(1-\beta_1^t)} \tag{18}$$

$$\widehat{v_{ij}^{(t)}} = \frac{v_{ij}^{(t)}}{(1-\beta_2^t)} \tag{19}$$

La regla de actualización final para cada peso w_{ij} es la siguiente:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \frac{\eta}{\sqrt{\widehat{v_{ij}^{(t)}} + \epsilon}} \widehat{m_{ij}^{(t)}} \tag{20}$$

5.3.2 Función de activación

Las unidades neuronales artificiales se inspiran en las neuronas biológicas, con algunas modificaciones para mayor comodidad. Al igual que las dendritas, las conexiones de entrada a la neurona llevan las señales de entrada atenuadas o amplificadas de otras neuronas vecinas. Las señales se transmiten a la neurona, donde las señales de entrada son resumidas y luego se toma una decisión en cuanto a qué salida basada en la entrada total recibida. Por ejemplo, para una neurona de umbral binario, se proporciona un valor de salida de 1 cuando la entrada total excede un predefinido umbral; de lo contrario, la salida se mantiene en 0. Varios otros tipos de neuronas se utilizan en los nervios artificiales redes, y su implementación solo difiere con respecto a la función de activación en la entrada total para producir la salida de la neurona. En la Ilustración 2, los diversos equivalentes biológicos se etiquetan en la neurona artificial para facilitar la analogía y la interpretación.

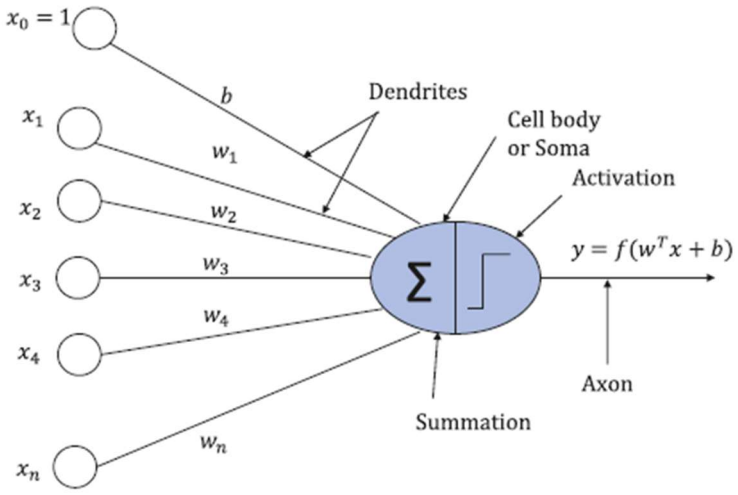

Ilustración 16- Esquema funcionamiento de una función de activación (Pattanayak, 2017).

Hay varias funciones de activación para las unidades neuronales, y su uso varía con respecto al problema a la mano y la topología de la red neuronal. En esta parte del estudio, se utilizará la función de activación RELU en la capa oculta y la sigmoid en la capa de salida porque esta configuración es común en el diseño de redes neuronales.

Rectified Linear Unit(ReLU) Activation Function :

En una unidad lineal rectificada, como se muestra en la Ilustración 17 la salida es igual a la entrada neta a la neurona si la entrada total es mayor que 0; sin embargo, si la entrada total es menor o igual a 0 las salidas neuronales son 0. La salida para una unidad ReLU (Keras API reference, s.f.) puede representarse de la siguiente manera:

$$f(x) = \max(0, x) \tag{21}$$

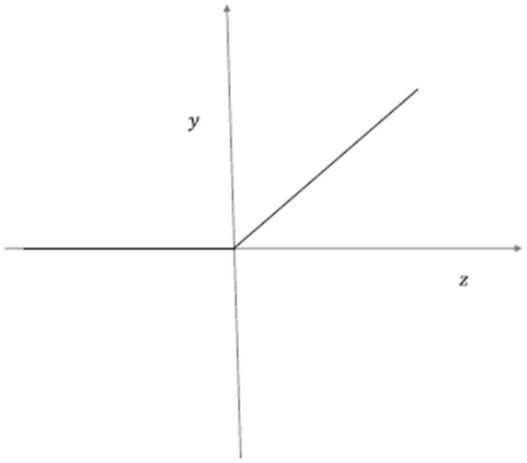

Ilustración 17- Función ReLU (Layer activation function, s.f.).

El ReLU es uno de los elementos clave que ha revolucionado el aprendizaje profundo. Son más fáciles de calcular. Las ReLUs combinan lo mejor de ambos mundos: tienen un gradiente constante mientras que la entrada neta es positiva y un gradiente cero en otros lugares. El gradiente constante para la entrada neta positiva asegura que el algoritmo de gradiente-descenso no pare el aprendizaje debido a un gradiente de fuga. Al mismo tiempo, la salida cero para una entrada neta no positiva produce no linealidad.

Hay varias versiones de funciones de activación de unidades lineales rectificadas, como la unidad lineal rectificada paramétrica (PReLU) y la unidad lineal rectificada con fugas.

Para una función normal de activación de ReLU, tanto la salida como los gradientes son cero para los valores de entrada no positivos, por lo que el entrenamiento puede detenerse debido al gradiente cero. Para que el modelo tenga un gradiente distinto de cero incluso cuando la entrada es negativa, PReLU puede ser útil. La relación entrada-salida para una función de activación PReLU viene dada por lo siguiente:

$$y = \max(0, z) + \beta \min(0, z) \tag{22}$$

Donde $z = w^T x + b$ es la entrada neta a la función de activación PReLI y β es el parámetro de aprendizaje del entrenamiento. Cuando β es igual a -1, $y = |z|$ y la función de activación se llama ReLU. Cuando β es igual a un pequeño valor como 0.01, la función de activación se llama fugas ReLU.

Sigmoid:

La ecuación de la sigmoid (Keras API reference, s.f.) es la siguiente:

$$y = \frac{1}{1 + e^{-z}} \tag{23}$$

donde $z = w^T + b$ es la entrada neta a la función sigmoid

- Si la entrada neta es un número positivo grande, $\lim_{z \to \infty} y = 1$
- Si la entrada neta es un numero negativo grande, $\lim_{z \to -\infty} y = 0$
- Si z=0, $y = \frac{1}{2}$

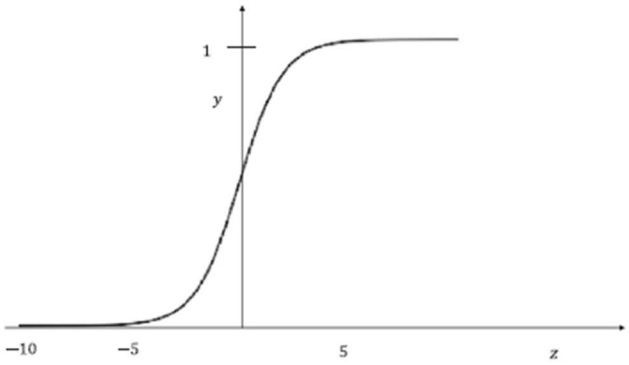

Ilustración 18- función Sigmoid (Layer activation function, s.f.).

La Ilustración 18 ilustra la relación entrada-salida de una función de activación sigmoide. La salida de una neurona que tiene una función de activación sigmoid es muy suave y da agradables derivados continuos, que funciona bien cuando se entrena una red neuronal. La salida de una función de activación sigmoide oscila entre 0 y 1. Debido a su capacidad para proporcionar valores continuos en el rango de 0 a 1, la función sigmoide se utiliza generalmente para generar probabilidad con respecto a una clase dada para una clasificación binaria. Las funciones de activación sigmoide en las capas ocultas introducen características no-lineales para que el modelo pueda aprender características más complejas.

5.3.3 Función de pérdida

El propósito de las funciones de pérdida es calcular la cantidad que un modelo debe tratar de minimizar durante el entrenamiento.

En el caso de este estudio, necesitamos una función que sale falso si la celda de la predicción no es correcta y verdadero si la celda predicada es correcta. Keras utiliza la

función de perdida “Sparse Categórical Crossentropy” para este tipo de problema. El sitio web de la biblioteca Keras recomienda utilizar esta función cuando hay más de dos valores finales posibles. En este estudio, se tiene 257 valores posibles.

5.3.4 Métricas

Una métrica es una función que se utiliza para juzgar el rendimiento de un modelo o red.

Las funciones métricas son como las funciones de pérdida, excepto que los resultados de la evaluación de una métrica no se utilizan cuando se entrena el modelo. Se debe tener en cuenta que se puede usar cualquier función de pérdida como métrica.

En este caso se puede utilizar dos métricas que son la “precisión “y la “Sparse Categorical Crossentropy” que son las mismas en nuestro caso

- Precisión: Calcula con qué frecuencia las predicciones son iguales a las etiquetas.
- Sparse categorical Crossentropy: Calcula con qué frecuencia las predicciones de la celda de origen de perturbación de la red coinciden con las celdas determinadas por el cálculo.

5.3.5 Numero de “Epochs” o iteraciones

Para entrenarse, una red ajuste los coeficientes de las neuronas en cada iteración. El número de Epochs es el número total de iteraciones.

5.4 El overfitting o sobreajuste

Un desafío significativo cuando se entrena un modelo de aprendizaje automático es decidir cuántas epochs ejecutar. Muy pocas epochs podrían no conducir a la convergencia del modelo, mientras que demasiadas epochs podrían conducir al sobreajuste.

El sobreajuste o overfitting ocurre cuando un modelo estadístico se ajusta exactamente a los datos de entrenamiento. Cuando esto sucede, el algoritmo desafortunadamente no puede funcionar con precisión contra otros, frustrando su propósito. La generalización de un modelo a nuevos datos es en última instancia lo que nos permite utilizar algoritmos de aprendizaje automático todos los días para hacer predicciones y clasificar datos.

Cuando se construyen algoritmos de aprendizaje automático,estos aprovechan un conjunto de datos de muestra para entrenar el modelo. Sin embargo, cuando el modelo se entrena durante demasiado tiempo con datos de muestra o cuando el modelo es demasiado complejo, puede comenzar a aprender el "ruido", o información irrelevante, dentro del conjunto de datos. Cuando el modelo memoriza el ruido y se ajusta demasiado al conjunto de entrenamiento, el modelo se vuelve "sobrecargado" y no puede generalizar bien a nuevos datos. Si un modelo no puede generalizar bien a nuevos datos, entonces no será capaz de realizar las tareas de clasificación o predicción para las que fue diseñado.

Las bajas tasas de error y una alta varianza son buenos indicadores de sobreajuste. Con el fin de evitar este tipo de comportamiento, parte del conjunto de datos de entrenamiento se reserva típicamente como el "conjunto de pruebas" para verificar si hay sobreajuste. Si los datos de entrenamiento tienen una baja tasa de error y los datos de prueba tienen una alta tasa de error, indica sobreajuste.

Si el entrenamiento excesivo o la complejidad del modelo resultan en sobrecalentamiento, entonces una respuesta lógica de prevención sería pausar el proceso de entrenamiento antes, también conocido como "parada temprana" o reducir la complejidad en el modelo eliminando insumos menos relevantes. Sin embargo, si hace una pausa demasiado temprano o excluye demasiadas características importantes, puede encontrar el problema opuesto, y en su lugar, puede que no se ajuste a su modelo. El subajuste se produce cuando el modelo no se ha entrenado durante suficiente tiempo o las variables de entrada no son lo suficientemente significativas como para determinar una relación significativa entre las variables de entrada y salida.

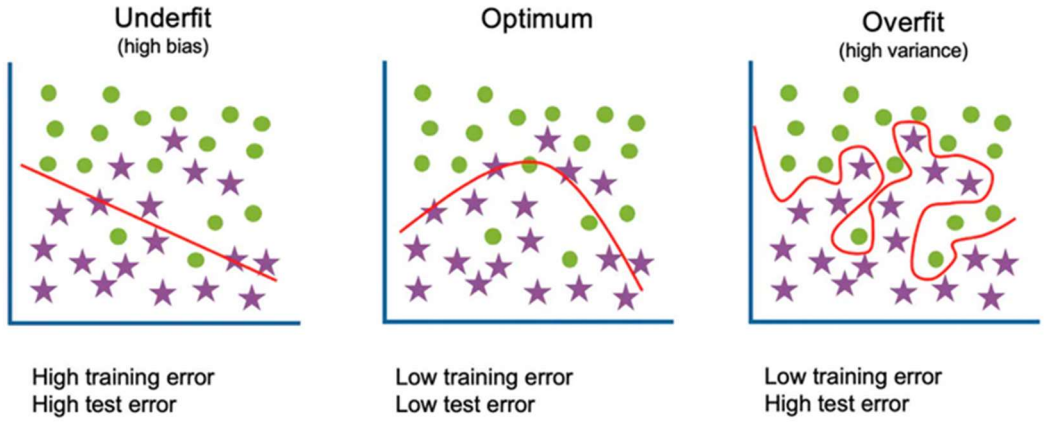

Ilustración 19- esquema de los casos de aprendizaje (IBM, s.f.).

En ambos casos, el modelo no puede establecer la tendencia dominante dentro del conjunto de datos de formación. Como resultado, la subadaptación también generaliza los datos poco visibles. Sin embargo, a diferencia del sobreajuste, los modelos subajustados experimentan un alto sesgo y menos varianza dentro de sus predicciones. Esto ilustra la compensación de varianza de sesgo, que ocurre cuando un modelo poco equipado se desplaza a un estado de sobreabastecimiento. A medida que el modelo aprende, su sesgo se reduce, pero puede aumentar en la varianza a medida que se vuelve sobrecargado. Al ajustar un modelo, el objetivo es encontrar el "punto dulce" entre el ajuste inferior y el ajuste excesivo, de modo que pueda establecer una tendencia dominante y aplicarla ampliamente a nuevos conjuntos de datos.

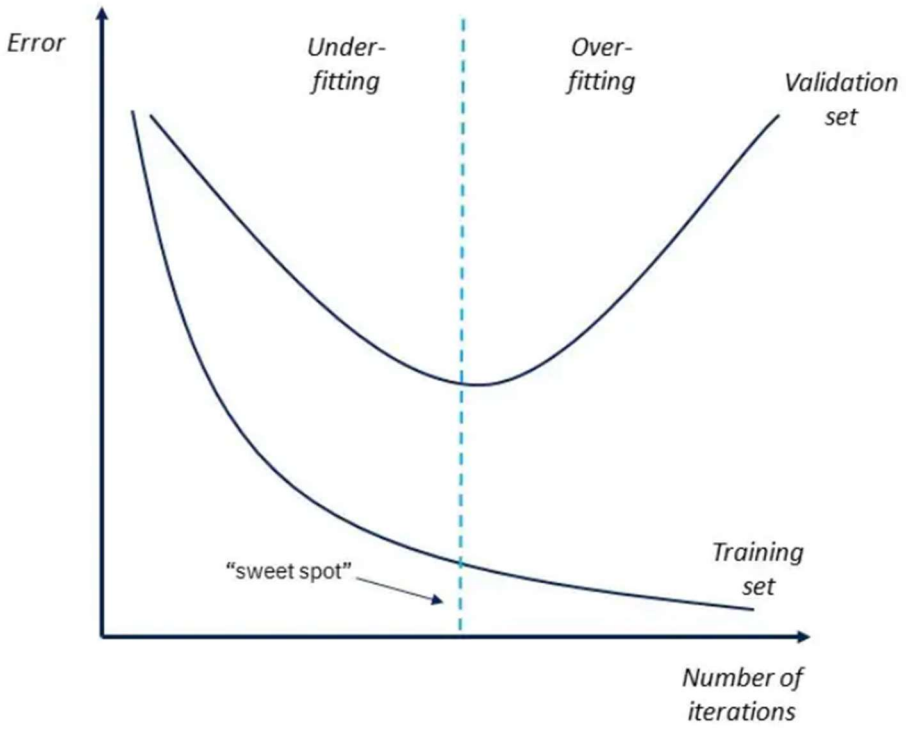

Ilustración 20- Explicación del Overfitting (IBM, s.f.).

Si bien el uso de un modelo lineal nos ayuda a evitar el sobreajuste, muchos problemas del mundo real son no lineales. Además de entender cómo detectar el sobreajuste, es importante entender cómo evitar el sobreajuste por completo. A continuación, se presentan las técnicas más comunes para evitar el sobreajuste:

- Parada temprana o early stopping: Este método busca pausar el entrenamiento antes de que el modelo comience a aprender el ruido dentro del modelo. Este enfoque corre el riesgo de detener el proceso de formación demasiado pronto,

lo que lleva al problema opuesto de la instalación insuficiente. Encontrar el "punto dulce" entre el ajuste y el sobreajuste es el objetivo final aquí.

- Entrenar con más datos: Ampliar el conjunto de entrenamiento para incluir más datos puede aumentar la precisión del modelo al proporcionar más oportunidades para analizar la relación dominante entre las variables de entrada y salida. Dicho esto, este es un método más eficaz cuando se inyectan datos limpios y relevantes en el modelo. De lo contrario, podría seguir agregando más complejidad al modelo, haciendo que se sobrecargue.

- Aumento de datos: Si bien es mejor inyectar datos limpios y relevantes en los datos de entrenamiento, a veces se agregan datos ruidosos para hacer un modelo más estable. Sin embargo, este método debe hacerse con moderación.

Early stopping con Keras:

Keras proporciona un módulo de parada temprana. En el sitio web de keras, se describe los argumentos del módulo:

- **monitor:** Cantidad a controlar.

- **min_delta**: El cambio mínimo en la cantidad supervisada para calificar como una mejora, es decir, un cambio absoluto inferior a min_delta, contará como ninguna mejora.

- **paciencia**: Número de épocas sin mejoría después de las cuales se detendrá el entrenamiento.

- **verbose**: Modo de verbosidad, 0 o 1. El modo 0 es silencioso, y el modo 1 muestra los mensajes cuando el callback realiza una acción.

- **modo**: Uno de {"auto", "min", "máx"}. En el modo min, el entrenamiento se detendrá cuando la cantidad monitoreada ha dejado de disminuir; en el modo "máx" se detendrá cuando la cantidad monitoreada ha dejado de aumentar; en el modo "auto", la dirección se infiere automáticamente del nombre de la cantidad monitoreada.

- **línea de base**: Valor de línea de base para la cantidad monitoreada. La capacitación se detendrá si el modelo no muestra una mejora sobre la línea de base.

- **restore_best_weights**: Si restaurar los pesos del modelo de la época con el mejor valor de la cantidad monitoreada. Si es False, se utilizan los pesos del modelo obtenidos en el último paso del entrenamiento. Una época será restaurada independientemente del desempeño relativo a la línea de base. Si ninguna época mejora en la línea de base, el entrenamiento se ejecutará para épocas de paciencia y restaurar pesos de la mejor época en ese conjunto.

Se ha utilizado los siguientes valores para obtener la máxima precisión(precisión) en un número mínimo de iteraciones ("Epochs") en el conjunto de validación:

- monitor=val_precisión (precisión del dataset de validación)
- baseline=0
- min_delta=0,001
- paciencia=50
- verbose=1
- restore_best_wights=True
- los otros parámetros son por defecto

5.5 Primer resultado

Con la arquitectura descrita en el apartado dedicado y los parámetros de la construcción de la red, se muestra el aprendizaje de la red y sus valores de precisión y pérdida en función de numero de "Epochs" es decir el número de turno de aprendizaje

Hay dos valores de precisión y de pérdida en el aprendizaje de una red neuronal:

- "Training Precisión": El valor de precisión del dataset de aprendizaje de la red
- "Training Loss": El valor de error del dataset de aprendizaje de la red
- "Validation Precisión": El valor de precisión del dataset de validación que no sirve para el aprendizaje
- "Validation Loss": El valor de error del dataset de validación que no sirve para el aprendizaje

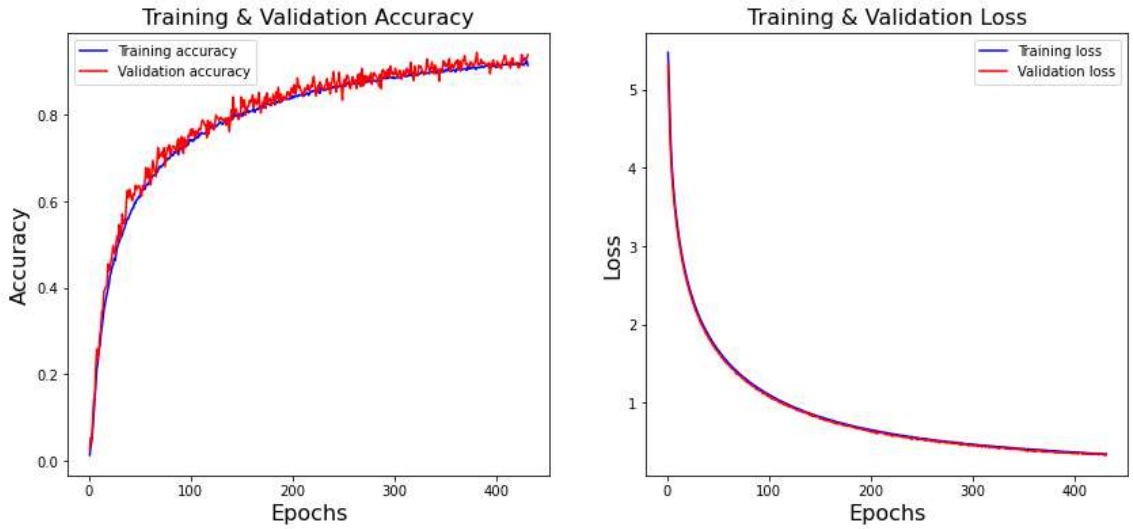

Ilustración 21 - Curva de aprendizaje de la red neuronal.

Valor de criterio	con modulo y fase
test Precisión	94.4%
Test Loss	0.3541
N de epochs	431

Tabla 3 - Valores de precisión, pérdida y numero de iteraciones.

En primer lugar, se puede ver que no hay “Overfitting” (sobreajuste) porque la curva de “Validation Precisión” sigue la curva de “Training Precisión”.

En segundo lugar, se observa que las curvas de precisión se asimilas a dos pendientes casi vertical al principio y casi horizontal al final. Eso quiere decir que la red puede llegar a un valor de precisión de 80% en 25 “Epochs”, pero necesita 375 “Epochs” para pasar de 80% a 98% de precisión

De manera análoga, el valor de pérdida puede llegar a un valor inferior a 1 en 25 “Epochs”. Sin embargo, necesita 375 “Epochs” para pasar de 1 a 0.2 valor de pérdida.

Entonces, se puede decir que hay dos fases de aprendizaje con esta red neuronal:

- 1 fase: fase de aprendizaje rápido
- 2 fase: fase de aprendizaje lento

5.6 Conclusiones del capítulo

En este capítulo, se ha visto una primera construcción de la red neuronal para resolver un problema. Para hacerlo, se ha identificado el problema de manera clara para que se puede elegir los parámetros de la red adecuados. Se ha visto la construcción del dataset que permite alimentar la red de casos de estudio y el problema más común de

las redes neuronales que es el sobreajuste. Al final, se ha visto las curvas de aprendizaje de la red y su valor final de precisión que no puede llegar al 100%.

En el siguiente capitulo, se estudiará con más detalle la red neuronal, su distintos parámetros y sus posibles optimizaciones.

CAPÍTULO 6: ESTUDIO DE LA RED Y OPTIMIZACIÓN

6 ESTUDIO DE LA RED Y OPTIMIZACÍON

Este capítulo se dedica al estudio de la red neuronal y sus distintos parámetros para saber cómo se puede optimizar el tiempo de aprendizaje de la red.

6.1 Número de neuronas en la capa oscura

Como se ha dicho anteriormente, se utiliza un número de neuronas en la capa oscura que sea una potencia de 2. En este apartado, se variará este número de neuronas entre 8 y 256.

Se construye una red neuronal por cada número de neuronas en la capa oscura; es decir, 6 redes al total (8,16,32,64,128 y 256 neuronas). Además, el carácter aleatorio de las condiciones iniciales (especialmente la inicialización de los pesos) obliga a entrenar 10 veces la misma red y coger la media de los resultados de precisión, pérdida y tiempo para tener una imagen global de estas variables. Al final se han entrenado 60 redes neuronales.

Se puede observar con los gráficos de la Ilustración 22 , las curvas valor de precisión y de pérdida final de cada red neuronal en función del número de neuronas en la capa oscura. Además, se puestan las curvas de tiempo de computación y el número de “Epochs” en función del número de neuronas en la Ilustración 23.

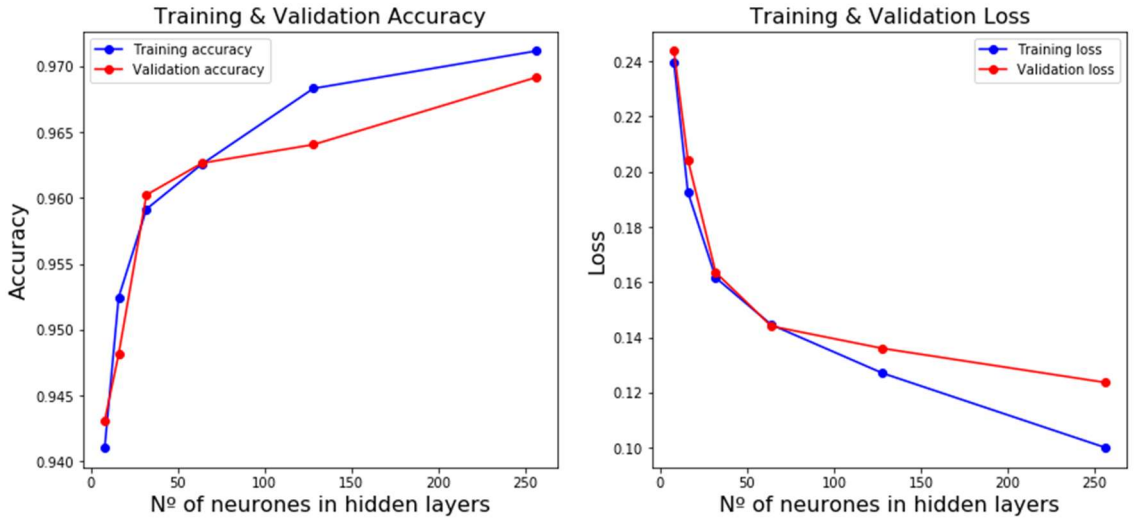

Ilustración 22 - Valores finales medias de precisión y de pérdida de las distintas redes neuronales.

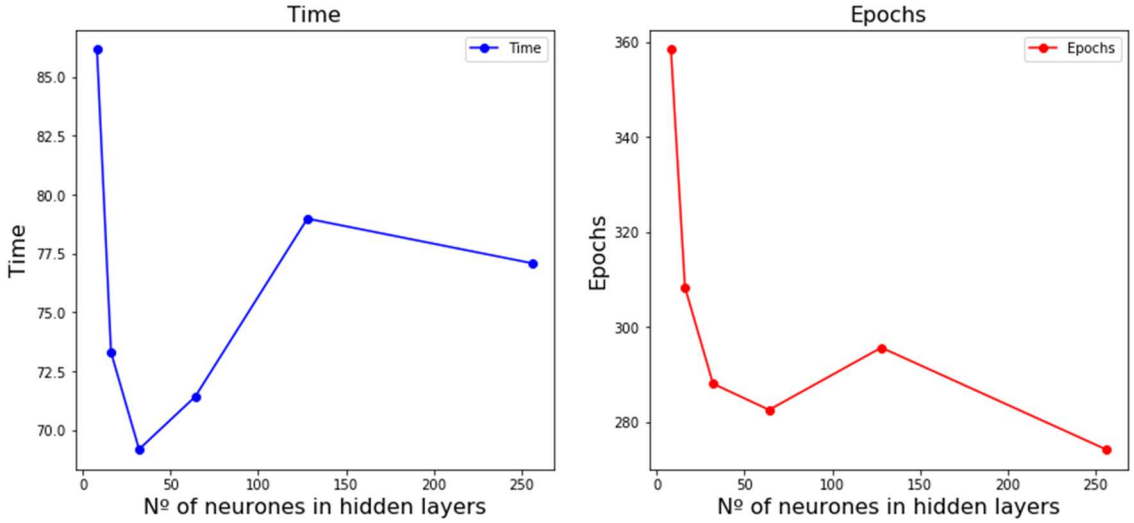

Ilustración 23 - Valores finales medias de tiempo y de número de Epochs de las distintas redes neuronales.

Podemos ver que, con los parámetros de early-stopping definidos anteriormente, se aumenta el nivel de precisión con el número de neuronas y el valor de pérdida disminuye.

Observamos que el tiempo de cálculo varia de la manera siguiente:

- Al principio, empieza con 86 s de tiempo de cálculo para 8 neuronas y disminuye hasta de 70 s para 32 neuronas.
- Después se aumenta el tiempo de computación hasta 78.5s para 126 neuronas y 256 neuronas

- Al final, el tiempo para 256 neuronas queda en el mismo orden de valor de tiempo

Se debe notar que no se calcula el tiempo de computación ni los valores de precisión y de pérdida después de 256 neuronas ya que se ha intentado hacerlo y se sabe que el tiempo de cálculo aumenta siempre con el número de neuronas.

El criterio de precisión está definido a 96%, entonces, se puede seleccionar las redes con 64,128 y 256 neuronas. El tiempo de computación de la red con 64 es el más bajo de los 3. Además, el tiempo de computación es de 2 segundos más que el mínimo, es decir una desviación relativa del 3% lo que es despreciable. Por todo ello, se elige la red con 64 neuronas en la capa oscura.

En este apartado, se ha entrenado 10 veces cada red neuronal para suprimir la componente aleatoria de las condiciones iniciales. Se ha elegido este número de 10 arbitrariamente. Por eso, se tiene que verificar si el mismo cálculo de esta red sea el mismo otra vez.

Abajo, se obtiene las curvas del mismo cálculo en las Ilustración 24 e Ilustración 25:

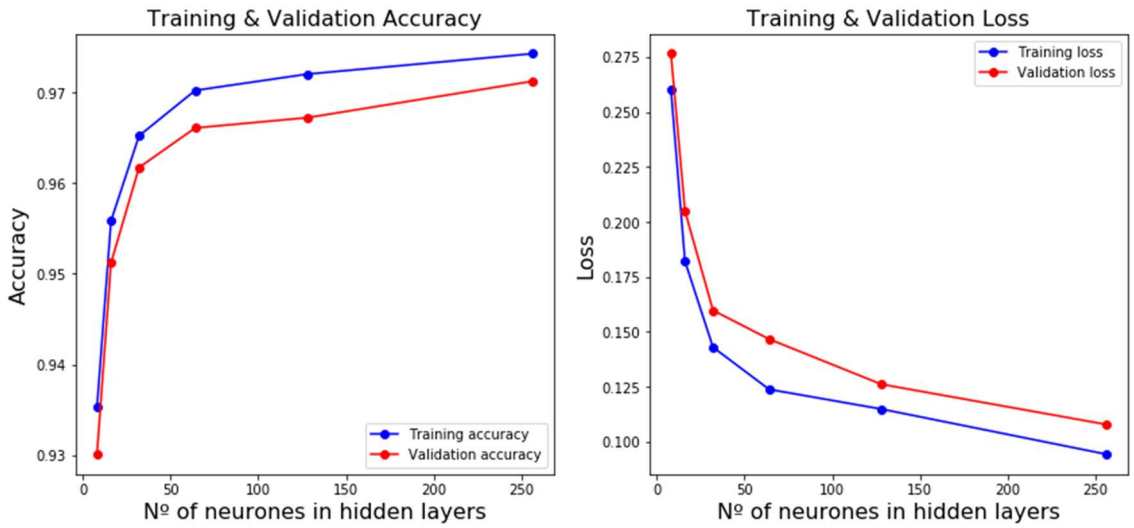

Ilustración 24 - Valores finales medias de precisión y de pérdida de las distintas redes neuronales

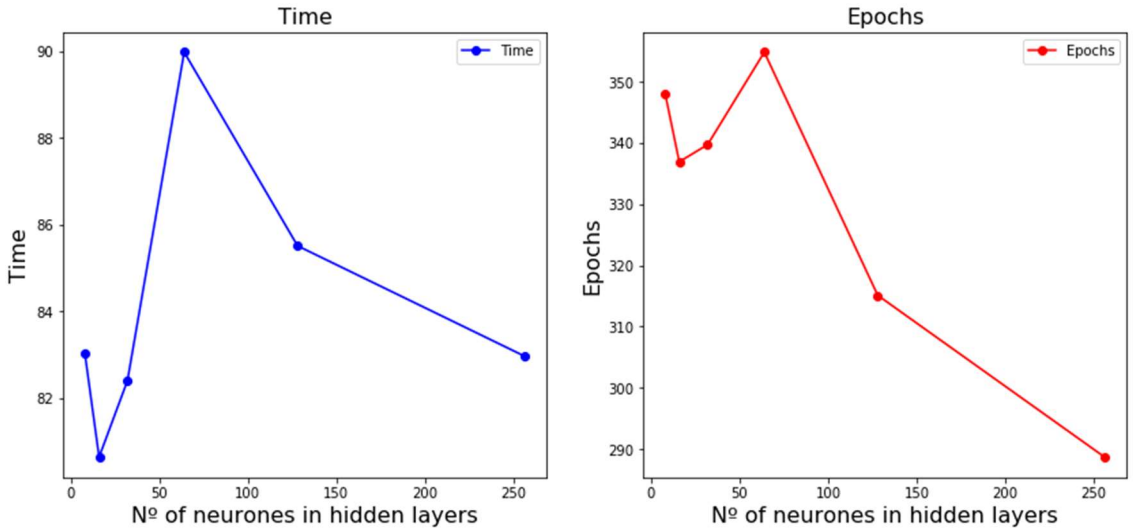

Ilustración 25 - Valores finales medias de Tiempo y de numero de Epochs de las distintas redes neuronales

Se observan pocas diferencias entre las curvas de precisión y de pérdida de los dos cálculos. Al revés, los tiempos de cálculo pueden variar considerablemente. Por ejemplo, se nota una desviación relativa del 10% para 128 neuronas o una de 27% para 64 neuronas.

Se concluye que las condiciones iniciales pueden tener una influencia no despreciable sobre los tiempos de computación.

Al final, vemos que el estudio de la optimización de tiempo de computación para el cálculo permite entender que el cálculo de entrenamiento de una red neuronal tiene un carácter aleatorio. Por eso, no se podría asegurarse que un estudio estadístico permitiría obtener un resultado optimo ya que, por un entrenamiento con 64 neuronas en la capa oculta, el tiempo de cálculo puede tener una desviación relativa de aproximadamente 25%.

Entonces, se elige un número de neuronas en la capa oculta de 128 porque es uno de los números de neurona que tiene los mejores criterios de precisión y de pérdida y de tiempo de computación, aunque no sea el número de neuronas optimo.

6.2 Variación de los datos de entrada

En este apartado, se estudia la influencia del número o del tipo de los datos de entrada sobre el entrenamiento de la red.

6.2.1 Supresión parcial de los datos de entrada

La idea de una red neuronal es de entrar un número máximo de datos para entrenar la red el más rápidamente posible. Se va a verificar esta idea en este apartado. Se debe recordar que los detectores dan con entrada a la red un módulo y una fase que corresponden al número complejo como resultado de la transformada de Fourier para la frecuencia dada. Se variará los datos de entrada de la siguiente manera:

- Se hace una simulación sin los datos de fase,
- Se hace una simulación sin los datos de modulo.

6.2.1.1 Datos de entrada sin información de fase

En este apartado, se modifica la cuantidad de información en cada caso. Es decir que, en los apartados anteriores, tenía la información sobre el módulo del flujo y la fase en cada detector. En este apartado, se pretende diseñar una red que funcionaria sin los datos de fase

Se haría una red neuronal casi igual a la de los casos anteriores:

- Numero de detectores :4
- Número de neuronas en la capa oculta: 126
- Número de neuronas a la salida: 257
- Loss función: “Sparse Categórical Crossentropy”
- Metrics: precisión
- Función de activación: ReLU
- “Optimizer”: Adam

Se cambiará el número de neuronas a la entrada a 4 porque en este caso solo, tenemos 4 informaciones de módulo.

Con esta red neuronal, obtenemos los resultados de la Ilustración 26:

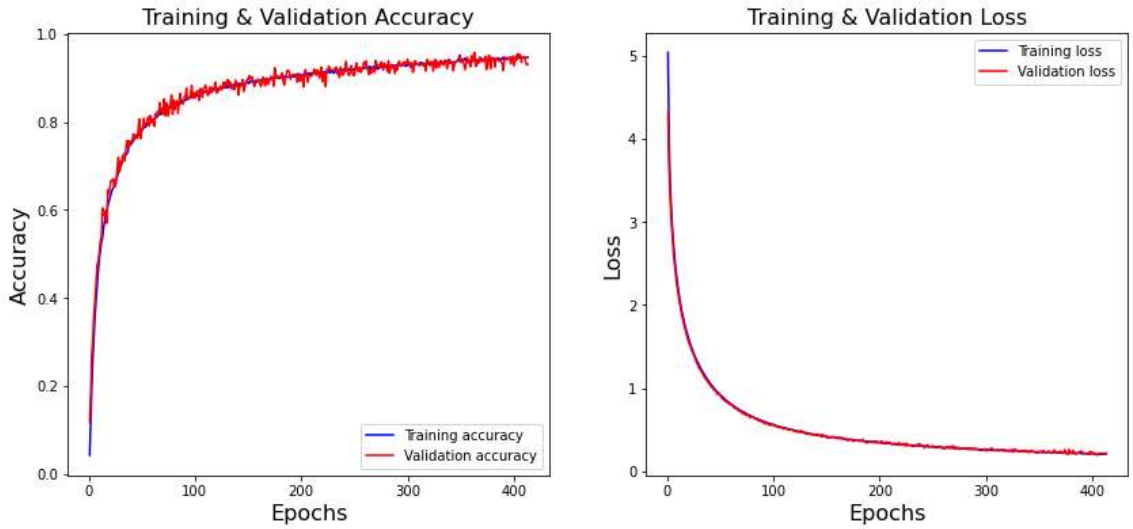

Ilustración 26- Curvas de aprendizaje para el caso de los datos de entrada sin información de fase.

De manera sorprendente, obtenemos resultados análogos a los resultados con información de fase.

Valor de criterio	con modulo y fase	sin fase
test Precisión	0.9444	0.958
Test Loss	0.3541	0.2237
N de epochs	431	413

Tabla 4- Valores de Precisión, de Loss y de Epochs para el caso de los datos de entrada sin información de fase.

Nota Bene: En este caso, las valores de Precisión, los y el número de epochs son mejores que el caso con modulo y fase ya que los valores de desviación relativa son:

- Desviación relativa Precisión=1,4%
- Desviación relativa de Loss= 5,8%
- Desviación relativa Nº Epochs=4,3%

Eso no quiere decir que la red sin información de fase es mejor sino similar. En efecto, el carácter aleatorio de las condiciones iniciales no permite hacer este tipo de conclusión.

Sin embargo, se puede cuestionar la utilidad de la información de fase. En caso siguiente, se estudia el caso sin modulo y con información de fase solo.

6.2.1.2 Datos de entrada sin módulo

En este apartado, se modifica la cuantidad de información en cada caso. Es decir que, en los apartados anteriores, tenía la información sobre el módulo del flujo y la fase en cada detector. En este apartado, se pretende diseñar una red que funcionaria sin los datos de modulo

Se hace una red neuronal casi igual a la de los casos anteriores:

- Número de detectores :4
- Número de neuronas en la capa oculta: 126
- Número de neuronas a la salida: 257
- Loss función: “Sparse Categórical Crossentropy”
- Metrics: precisión
- Función de activación: ReLU
- “Optimizer”: Adam

Se cambiará el número de neuronas a la entrada por 4 porque en este caso tenemos 4 informaciones de fase (se quita la información de modulo).

Se debe notar que se supone que los datos de fase tienen un error que sigue una ley normal

Con esta red neuronal, obtenemos los resultados de la Ilustración 27:

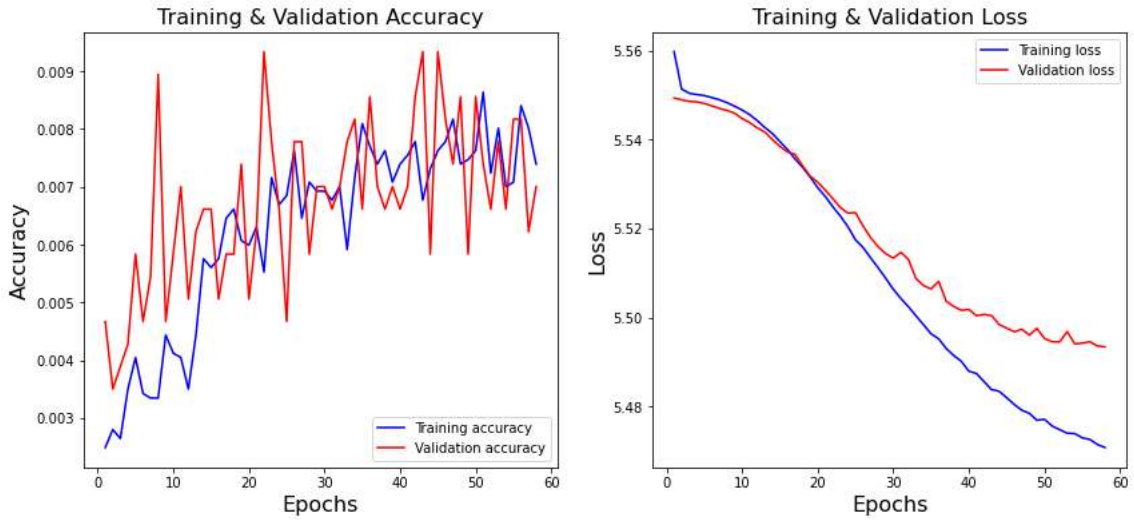

Ilustración 27- Curvas de aprendizaje para el caso de los datos de entrada sin información de modulo.

Con los criterios de apagamiento de early stopping, los datos de fase no permiten entrenar la red. Se intenta otra vez sin el early stopping. Se obtiene los resultados de la Ilustración 28:

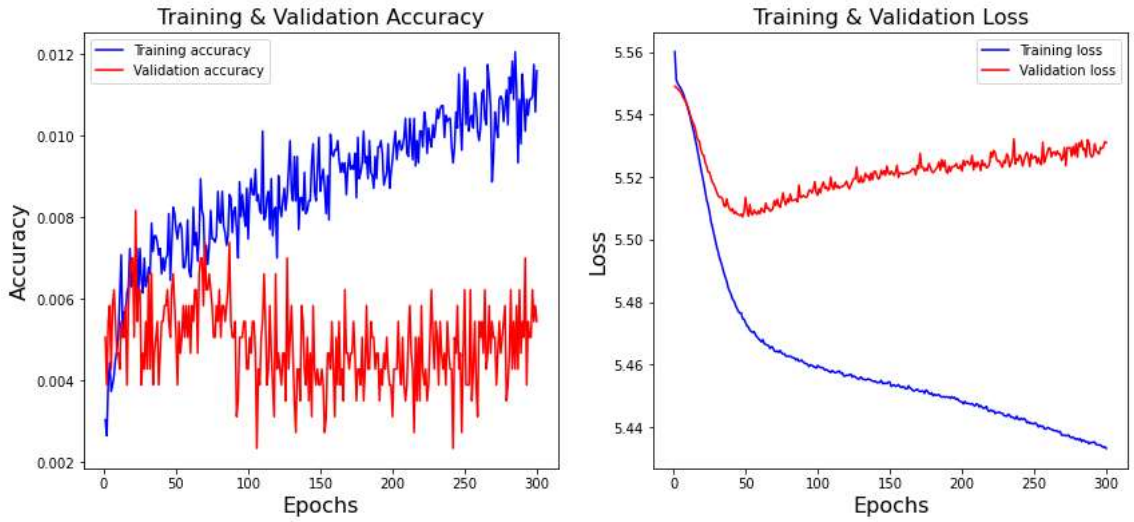

Ilustración 28- Curvas de aprendizaje para el caso de los datos de entrada sin información de modulo y sin early-stopping.

Valor de criterio	con modulo y fase	sin fase	sin modulo
test Precisión	0.9444	0.958	0.0089
Test Loss	0.3541	0.2237	5.5465
N de epochs	431	431	

Tabla 5- Valores finales de Precisión, Loss y Epochs en el caso de aprendizaje estudiados.

Se observa que las evoluciones de Precisión para los datos de entrenamiento y de probación son distintas. El Precisión de los datos de entrenamiento oscilan fuertemente con una tendencia a aumentar de manera linear mientras que la curva de Precisión de datos de probación oscila sin tendencia a aumentar o decrecer.

De manera análoga en las curvas de Loss, las curvas de entrenamiento y de probación no se siguen, aunque en este caso, las curvas no oscilan comparado a las curvas de Precisión.

Eso es característico de una red neuronal que no puede entrenarse por fallo de datos de entrada suficiente. Se concluye que utilizar solamente los datos de fase no es suficiente para entrenar la red y resolver el problema correctamente.

En conclusión, se ha visto que se puede entrenar la red con datos de módulo sin datos de fase de manera similar al entrenamiento con datos de módulos y fases. Como los datos son solamente de dos tipos, fase y modulo, se puede afirmar que los datos de fase no son útiles para el entrenamiento de la red

6.2.2 Variación del número de detectores

Anteriormente, se ha puesto los detectores en 4 puntos del reactor (posiciónes 25,35,223,233) como se puede ver en la ilustración 30:

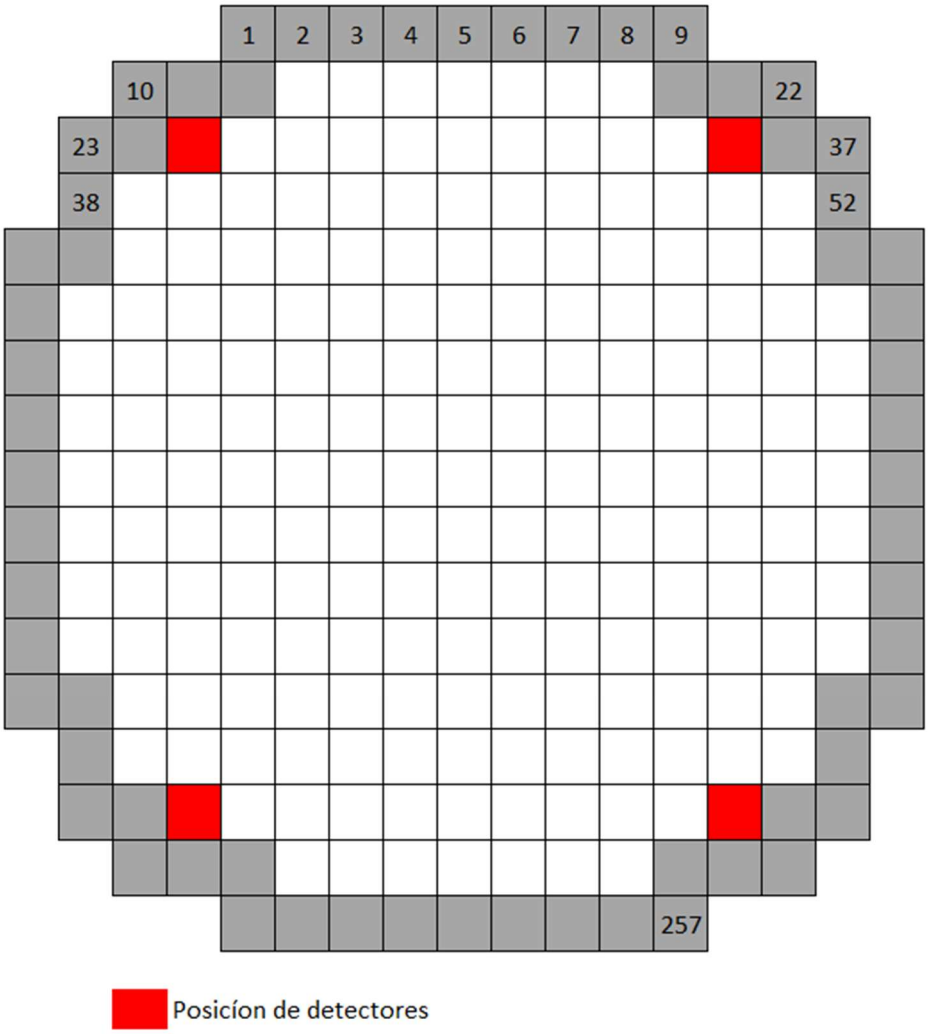

Ilustración 29- Posición de los 4 detectores en el reactor.

En este apartado, se variará el número de detectores con distintos casos para ver la influencia sobre el aprendizaje de la red.

6.2.2.1 Caso 1: espacio de celdas definido entre los detectores

Se pone un detector en todas las 4 posiciones es decir en la posición 4, 8,12,16 … hasta el fin como se puede ver en la ilustración 31:

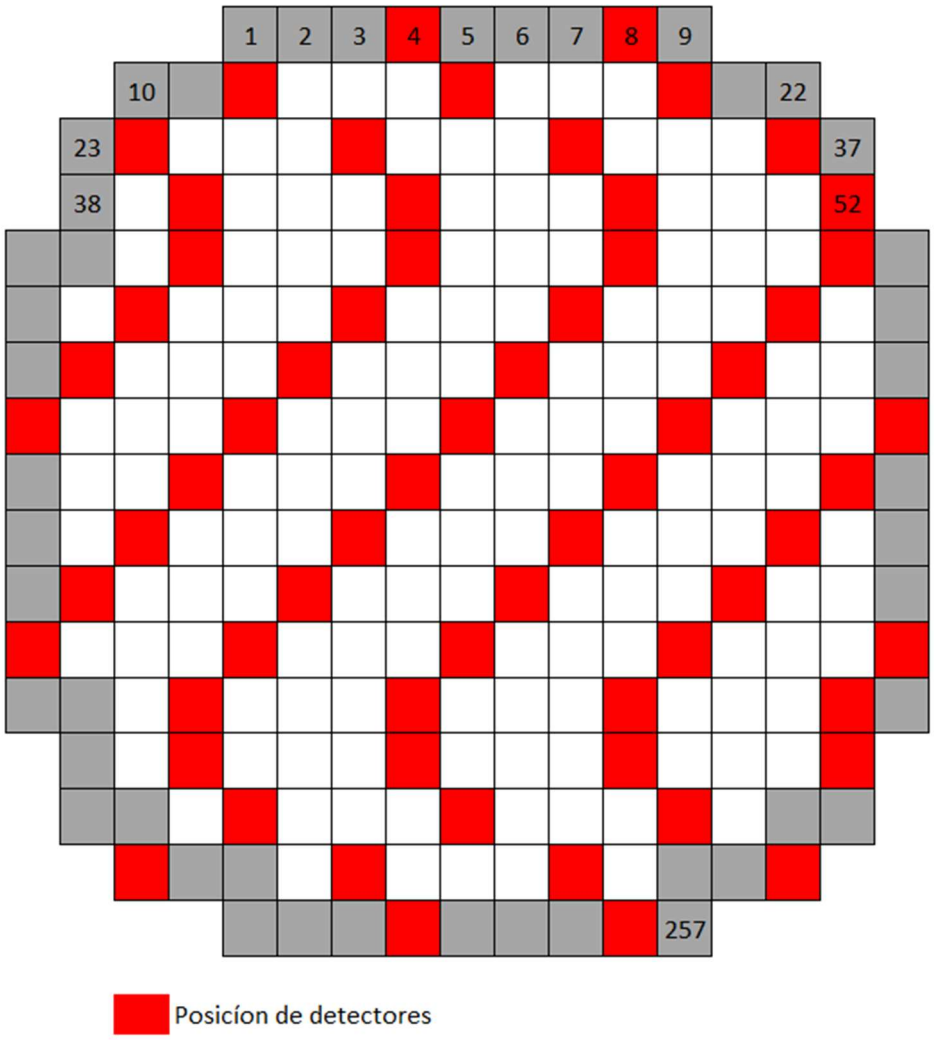

Ilustración 30- Posición de los detectores en el caso de un espacio entre detectores de 3 celdas.

Con esta configuración, obtenemos las curvas de aprendizaje de la Ilustración 31:

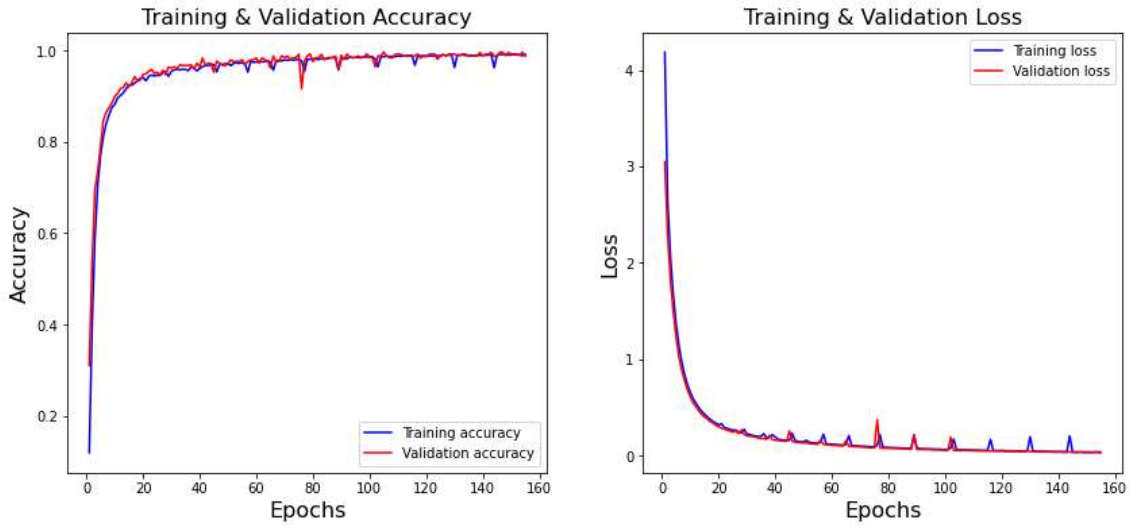

Ilustración 31- Curvas de aprendizaje para el caso 1.

Se alcanza un valor de precisión de 0.98 y un valor de pérdida de 0.09 a partir de 146 epochs.

Entonces se a dividido el tiempo de cálculo por la mitad mientras que se ha multiplicado el número de detectores por 16.

Como este caso con 4 celdas de espacio entre cada detector es arbitrario, se va a poner la curva del número de epochs necesario para alcanzar valores de precisión y pérdida en función del número de espacio entre detectores empezando con 4 celdas de espacio y acabando con 100 celdas de espacio.

Se obtiene la tabla y las curvas siguiente:

Periodicidad de la posición de los detectores	N_epochs	Val_precisión	Val_loss
4	146	0.99	0.06
5	203	0.9922	0.0621
6	149	0.9844	0.0816
7	196	0.9844	0.097
8	163	0.9759	0.1157
9	224	0.9833	0.018
10	224	0.9895	0.0751
11	280	0.9868	0.0824
12	263	0.984	0.083
13	226	0.9817	0.1082
14	220	0.9732	0.1333
16	320	0.9638	0.1667
19	254	0.984	0.1176
20	144	0.956	0.2411

23	236	0.9696	0.1573
25	388	0.9704	0.1615
28	194	0.9537	0.2299
30	335	0.9603	0.1928
35	261	0.9486	0.2725
40	352	0.942	0.2565
50	315	0.8899	0.4634
65	348	0.9248	0.7031
85	276	0.5821	1.3557

Tabla 6- Valores finales de precisión, pérdida y de Epochs para casos de espacio entre detectores diferente.

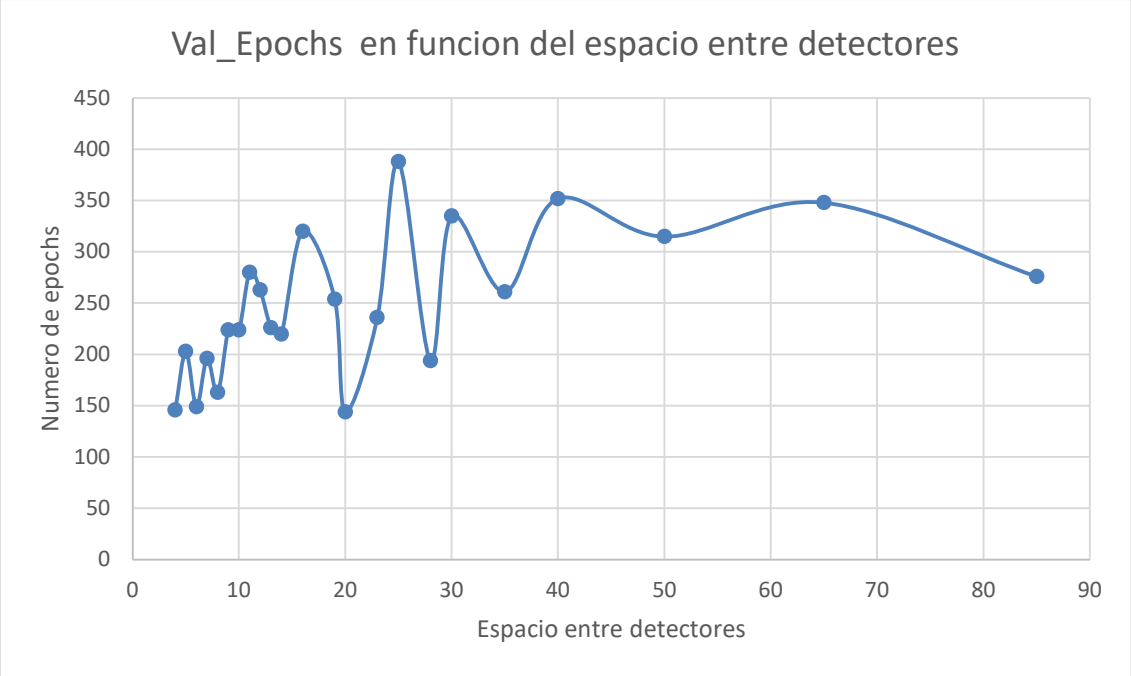

Ilustración 32- Número de epochs en función de la periodicidad de la posición de los detectores.

Esta curva traduce el tiempo de cálculo (N_epochs) por el entrenamiento de la red en función del espacio entre cada detector.

Se recuerda que el entrenamiento de la red tiene una parte aleatoria con el valor inicial de los pesos. Además, los parámetros del early-stopping no permiten alcanzar los mismos valores de precisión y péridida en cada caso sino valores que al menos satisfacen el pliego de condiciones Entonces se analizará las tendencias de las curvas sin tomar en cuenta las valores que son por debajo o por encima de la tendencia. Un análisis riguroso se haría haciendo un análisis estadístico con al menos 100 entrenamiento por cada caso lo que cuesta un tiempo de cálculo demasiado grande para obtener conclusiones similares al del análisis de la tendencia.

En la curva arriba, se puede observar que la tendencia es de aumentar el número de epochs (ie: el tiempo de cálculo) en función del espacio entre los detectores (ie: cuanto menos detectores hay)

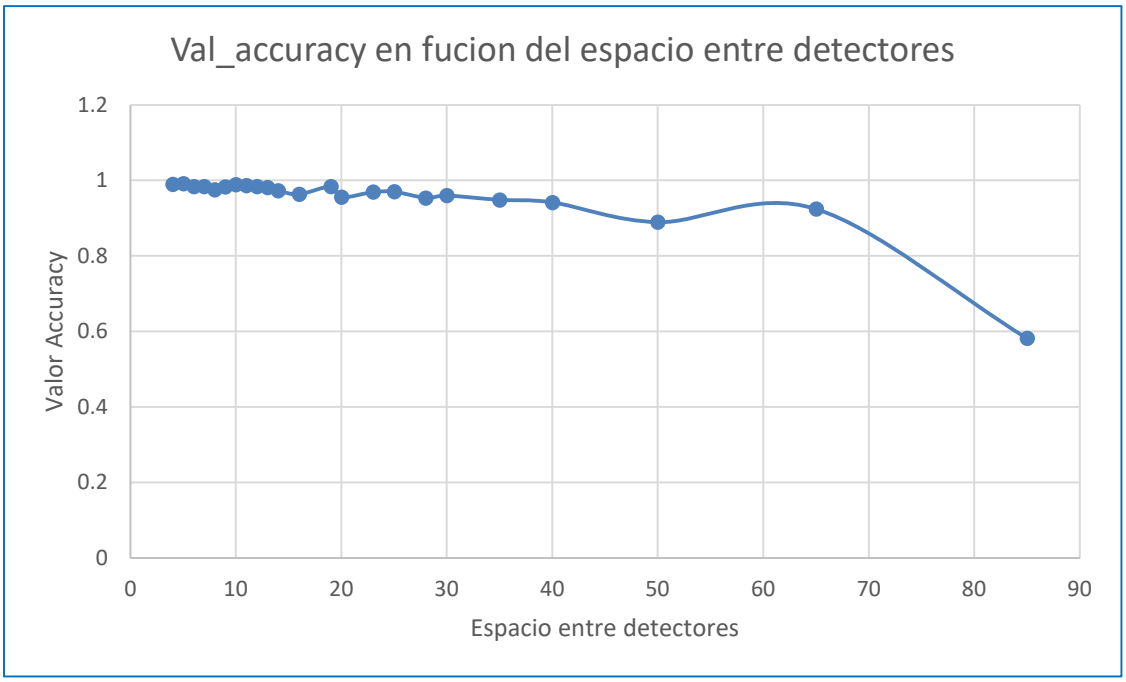

Ilustración 33- Valor final de precisión en función de la periodicidad de la posición de los detectores.

Esta curva representa el valor de precisión obtenido (con los criterios de early-stopping) en cada caso. Se puede ver que este valor decrece cuanto menos detectores hay. Por ejemplo, el último caso está por debajo de los otros valores por tener solamente 2 detectores.

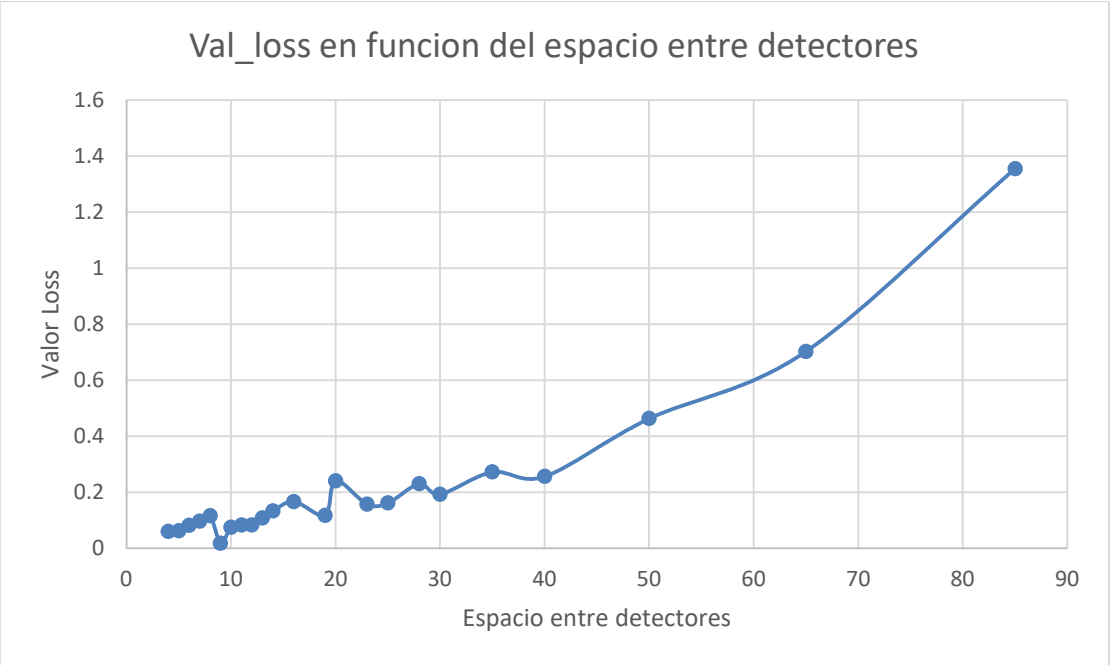

Ilustración 34- Valor final de pérdida en función de la periodicidad de la posición de los detectores.

De manera inversa a la curva de Precisión, el valor de Loss crece cuanto menos detectores hay.

Conclusión:

Con este estudio, vemos que el aumento del número de detectores permite reducir el tiempo de computación, aumentar el valor de Precisión y disminuir el valor de Loss. No podemos concluir sobre el número de detectores que poner ya que los reactores tienen un numero definido de detectores y que se analiza las tendencias de las curvas. Además, esta disposición no conforme a la realidad ya que los detectores no se ponen en función del espacio entre uno y otros.

Se analizará un caso cerca de la realidad en el ejemplo siguiente.

6.2.2.2 Caso 2: caso real de acuerdo con la literatura

Se puede encontrar una disposición que conforme a la realidad en el estudio de Francesco Calivá titulado: “Anomaly detection in nuclear reactors using Deep learning”. (Francesco Caliva, 2018)

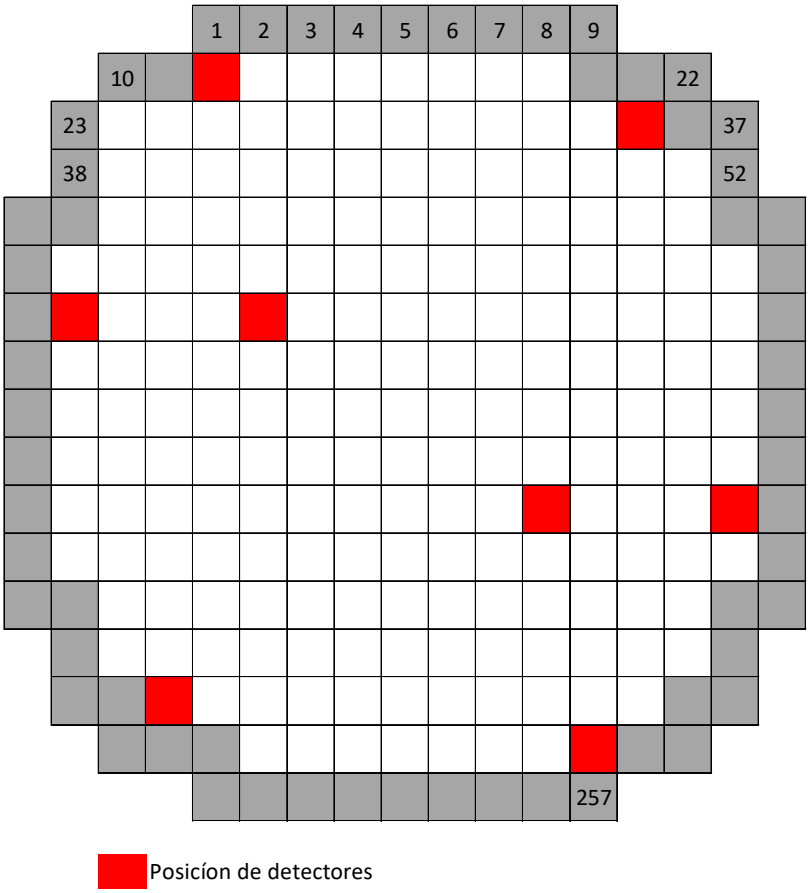

Ilustración 35- Posición de detectores en un caso real.

Se pone detectores en las posiciones 12, 35,88, 92, 166, 170, 223, 246 como se puede ver en la imagen arriba.

Se va a probar si con esta disposición si la red puede obtener resultados que satisfacen los criterios de predicción.

Se obtiene el resultado de la tabla 6:

N_epochs	V_Precisión	V_Loss
234	0.9724	0.1799

Tabla 7- Valores finales de precisión, pérdida y Epochs para posiciones de detectores reales.

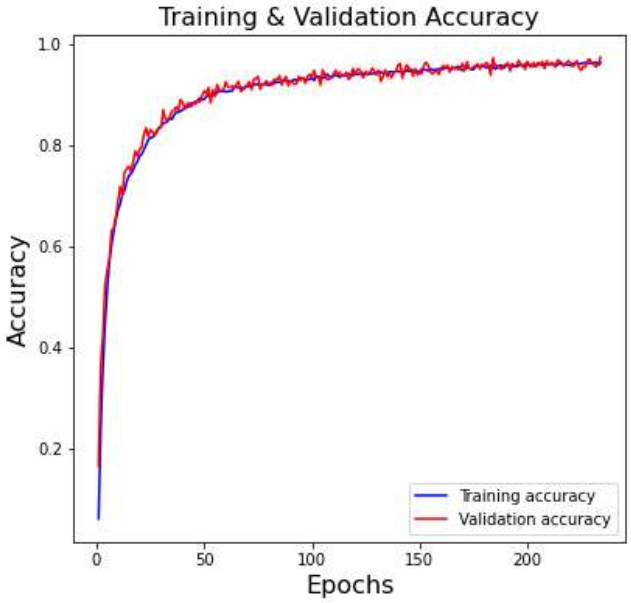

Ilustración 36- Curva de evolución de precisión en fusión del número de Epochs.

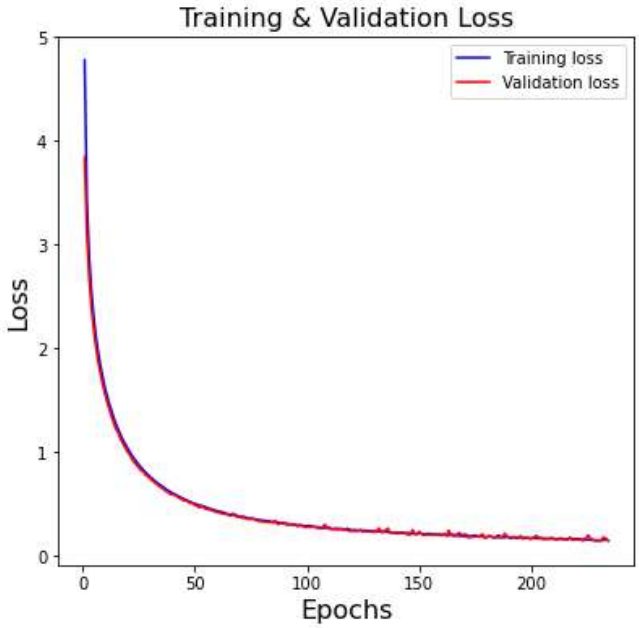

Ilustración 37- Curva de evolución de pérdida en función del número de Epochs

Como se podía esperar añadiendo detectores en el reactor, el entrenamiento de la red funciona perfectamente y se podía utilizar esta red en un funcionamiento real.

6.3 Variación de la función Optimizador

Como recordatorio, los algoritmos de optimización son responsables de reducir las pérdidas y proporcionar los resultados más precisos posibles. En la primera red construida en este estudio, se utilizó el optimizador Adam.

En este aparatado, se definirá los otros optimizadores disponibles en la biblioteca Keras y el resultado correspondiente a la red.

6.3.1 Adagrad

El optimizador Adagrad es un optimizador de primer orden como descenso de gradiente, pero con algunas modificaciones. En lugar de tener una tasa de aprendizaje global, la tasa de aprendizaje se normaliza para cada dimensión de la que depende la función de costo. La tasa de aprendizaje en cada iteración es la tasa de aprendizaje global dividida por la norma l2 de los gradientes anteriores hasta la iteración actual para cada dimensión.

Si la función de costo $c(\theta)$ donde $\theta = [\theta_1\theta_2 \dots \theta_n]^T \in \mathbb{R}^{nx1}$, entonces la regla de actualización para θ es la siguiente:

$$\theta_i^{t+1} = \theta_i^t - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} \theta_i^{(\tau)_2} + \epsilon}} \frac{\partial C^{(t)}}{\partial \theta_i} \tag{24}$$

Donde η es la tasa de aprendizaje y $\theta_i^{(t)}$ y $\theta_i^{(t+1)}$ son los valores del ieme parámetro de la iteración t y t+1 respectivamente.

Este es un buen optimizador para usar en aplicaciones con procesamiento de lenguaje natural y procesamiento de imágenes donde los datos son escasos.

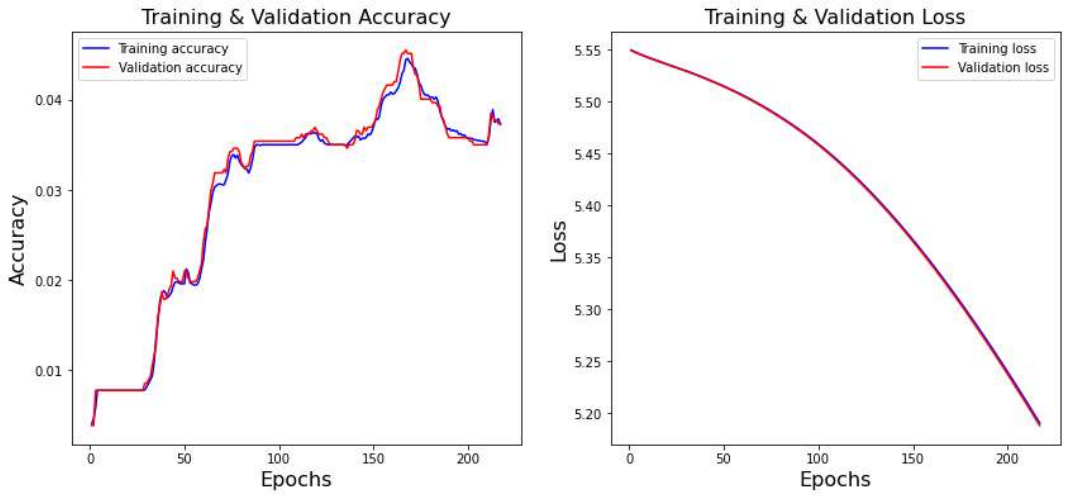

Ilustración 38- Curvas de aprendizaje con el optimizador Adagrad y con Early-stopping.

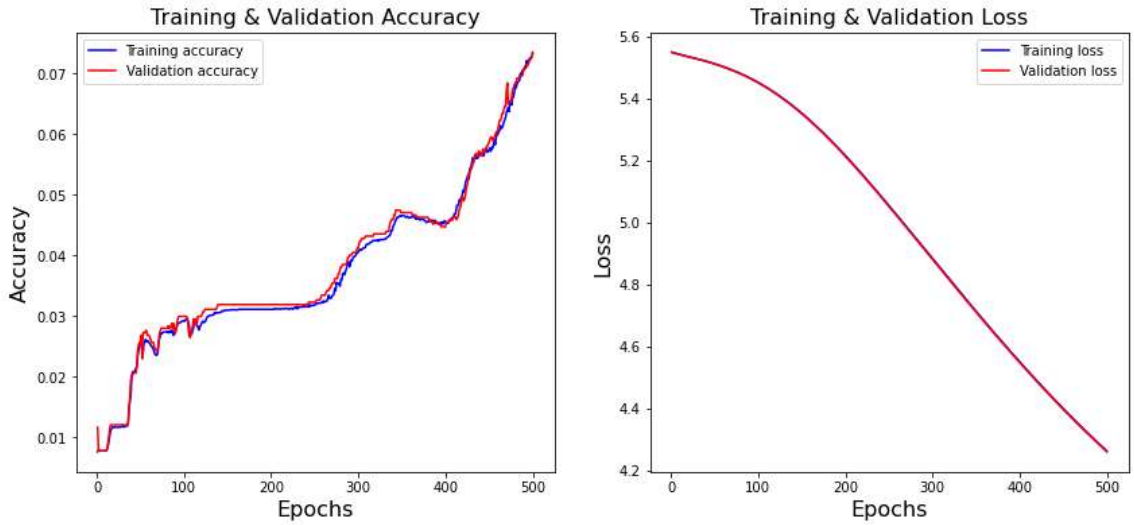

Ilustración 39- Curvas de aprendizaje con el optimizador Adagrad y sin Early-stopping.

Se ha hecho dos redes neuronales para el caso de Adagrad. La Ilustración 38Ilustración 39- Curvas de aprendizaje con el optimizador Adagrad y sin Early-stopping. Es la red con el "Early Stopping". Como se ha acabado el aprendizaje sin que alcanca un valor suficiente para decir que la red aprendí (alrededor de 80%-90%), se ha hecho una red sin "Early Stopping" para ver si el problema viene de los criterios de parada del "Early Stopping": es la Ilustración 39

Al final, la red alcance un valor de precisión de 7% y de pérdida de 4.25. Además, como en el primer caso, las curvas no tienen formas características de un buen aprendizaje (un aprendizaje rápido seguido por un aprendizaje lento) sino que tiene una curva de precisión con variaciones bruscas y una curva de pérdida con un decrecimiento casi constante.

Entonces, se puede decir que el optimizador no conviene al aprendizaje de la red.

6.3.2 RMSprop,

RMSprop es una técnica de optimización que funciona mejor para el aprendizaje por lotes. Rprop resuelve el problema de los gradientes que no apuntan al mínimo en los casos en que los contornos de la función de coste son elípticos. Como discutimos anteriormente, en tales casos, en lugar de una regla de aprendizaje global, una regla de actualización adaptativa separada para cada peso conduciría a una mejor convergencia. Lo especial con Rprop es que no utiliza la magnitud de los gradientes del peso, sino solo los signos para determinar cómo actualizar cada peso.

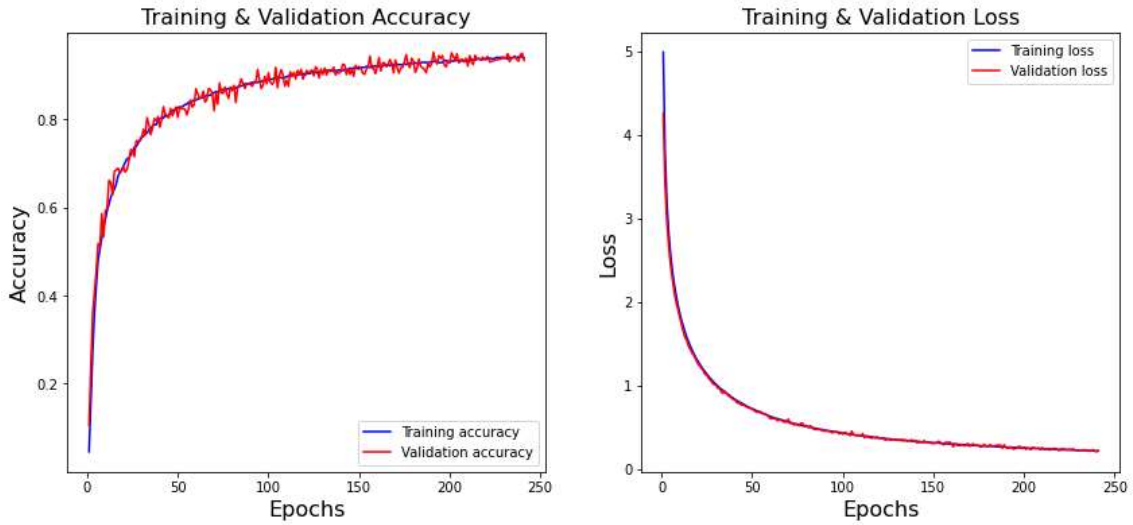

Ilustración 40- Curvas de aprendizaje con el optimizador RMSprop y con Early-stopping.

El valor final de precisión es del 98.6% y el valor final de pérdida es de 0.289. Además, las curvas tienen las características de un buen aprendizaje; es decir, una fase de aprendizaje rápida seguida por una fase de aprendizaje lento.

Entonces, se puede decir que la red aprende con este optimizador.

6.3.3 Adadelta,

El optimizador Adadelta es una variante de Adagrad que es menos agresiva en la reducción de la tasa de aprendizaje. Para cada conexión de peso, Adagrad escala la constante de tasa de aprendizaje en una iteración dividiéndola por el cuadrado medio raíz de todos los gradientes pasados para ese peso hasta esa iteración. Por lo tanto, la tasa de aprendizaje eficaz para cada peso es una función monótona decreciente del número de iteración, y después de un número considerable de iteraciones la tasa de aprendizaje se vuelve infinitamente pequeña. Adagrad supera este problema al tomar la media de los gradientes cuadrados en decadencia exponencial para cada peso o dimensión.

Por lo tanto, la tasa de aprendizaje eficaz en Adadelta sigue siendo más de una estimación local de sus gradientes actuales y no se reduce tan rápido como el método Adagrad. Esto asegura que el aprendizaje continúe incluso después de un número considerable de iteraciones o épocas. La regla de aprendizaje para Adadelta se puede resumir de la siguiente manera:

$$g_{ij}^{(t)} = \gamma\, g_{ij}^{(t-1)} + (1-\gamma)\left(\frac{\partial C^{(t)}}{\partial w_{ij}}\right)^2 \tag{25}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \frac{\eta}{\sqrt{g_{ij}^{(t)} + \epsilon}}\frac{\partial C^{(t)}}{\partial w_{ij}} \tag{26}$$

Donde γ es la constante de decaimiento exponencial, η es una constante de tasa de aprendizaje y $g_{ij}^{(t)}$ representa la gradiente cuadrado medio efectivo en iteración t. Podemos denotar el término $g_{ij}^{(t)}+\in$ como $RMS\left(g_{ij}^{(t)}\right)$, que da la regla de actualización de la siguiente manera:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \frac{\eta}{RMS\left(g_{ij}^{(t)}\right)} \frac{\partial C^{(t)}}{\partial w_{ij}} \tag{27}$$

Sin detallar las ecuaciones de este método, se sabe que se puede eliminar la tasa de aprendizaje η.

Una ventaja significativa de Adadelta es que elimina por completo la constante de tasa de aprendizaje. Si comparamos Adadelta y RMSprop, ambos son iguales si dejamos de lado la eliminación constante de la tasa de aprendizaje. Adadelta y RMSprop se desarrollaron independientemente alrededor del mismo tiempo para resolver el problema de decaimiento de la tasa de aprendizaje rápido de Adagrad.

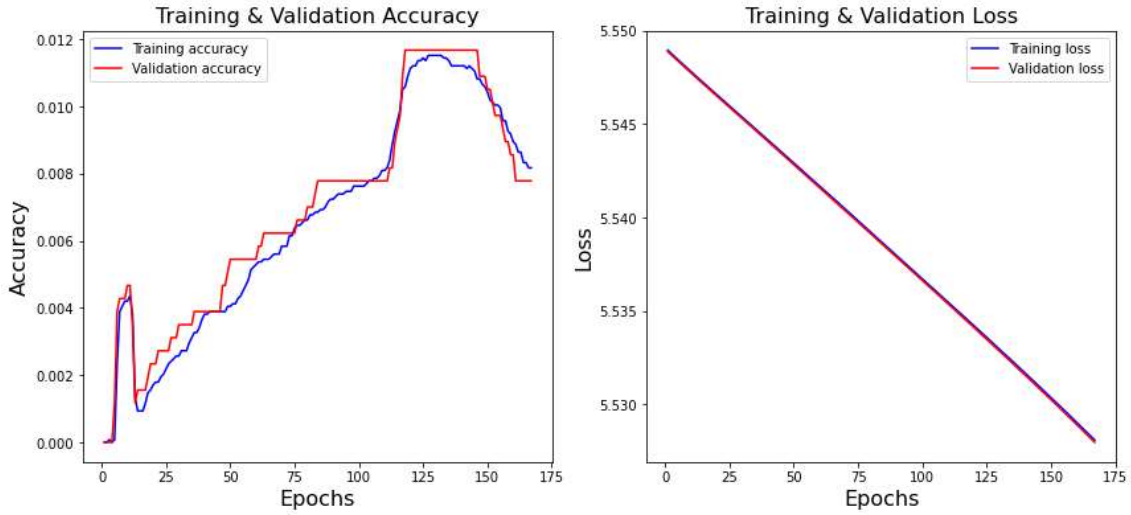

Ilustración 41- Curvas de aprendizaje con el optimizador Adadelta y con Early-stopping.

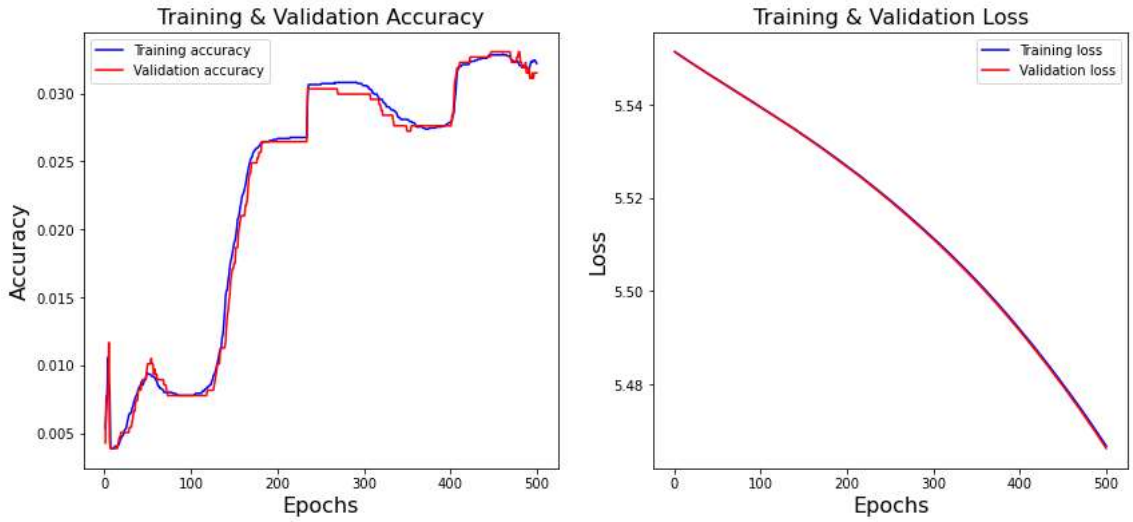

Ilustración 42- Curvas de aprendizaje con el optimizador Adadelta y sin Early-stopping.

Se ha hecho dos redes neuronales para el caso de Adadelta. La Ilustración 41: Es la red con el "Early Stopping". Como se ha acabado el aprendizaje sin que alcance un valor suficiente para decir que la red aprendí (alrededor de 80%-90%), se ha hecho una red sin "Early Stopping" para ver si el problema viene de los criterios de parada del "Early Stopping" : es la Ilustración 42.

Al final, la red alcance un valor de precisión de 3% y de pérdida de 5.46. Además, como en el primer caso, las curvas no tienen formas características de un buen aprendizaje (un aprendizaje rápido seguido por un aprendizaje lento) sino que tiene una curva de precisión con variaciones bruscas y una curva de pérdida con un decrecimiento casi constante.

Entonces, se puede decir que este optimizador no conviene al aprendizaje de la red.

6.3.4 Adam

Recordatorio de las curvas de aprendizaje con el optimizador ADAM.

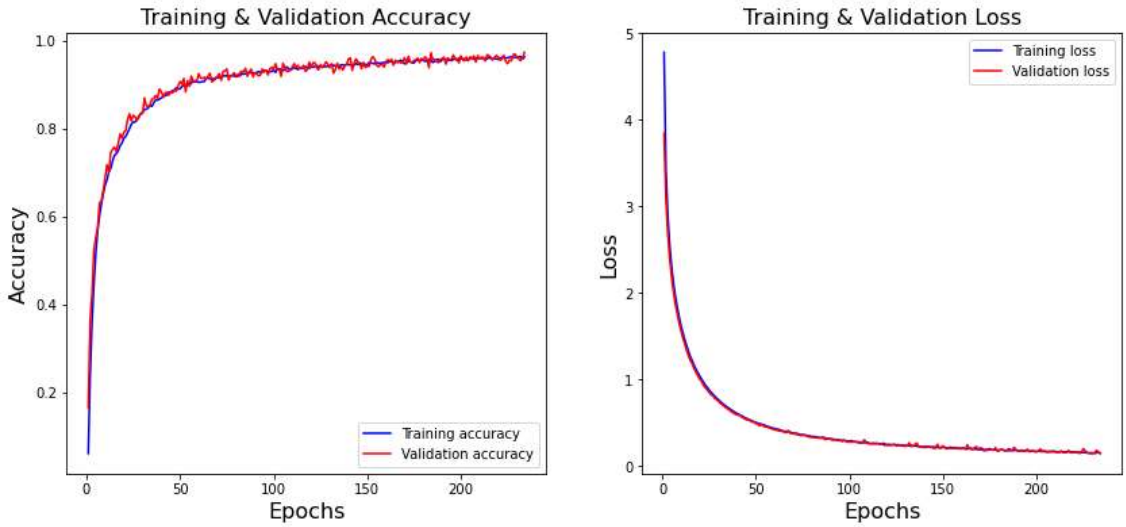

Ilustración 43- Curvas de aprendizaje con el optimizador Adam y con Early-stopping.

El valor final de precisión es de 97.2% y el valor final de pérdida es de 0.179. Además, las curvas tienen las características de un buen aprendizaje; es decir, una fase de aprendizaje rápida seguida por una fase de aprendizaje lento.

Entonces, se puede decir que la red aprendí con este optimizador.

6.3.5 Adamax

Optimizador que utiliza el algoritmo Adamax. Es una extensión de Adam que generaliza el enfoque de la norma infinita y puede resultar en una optimización más eficaz en algunos problemas.

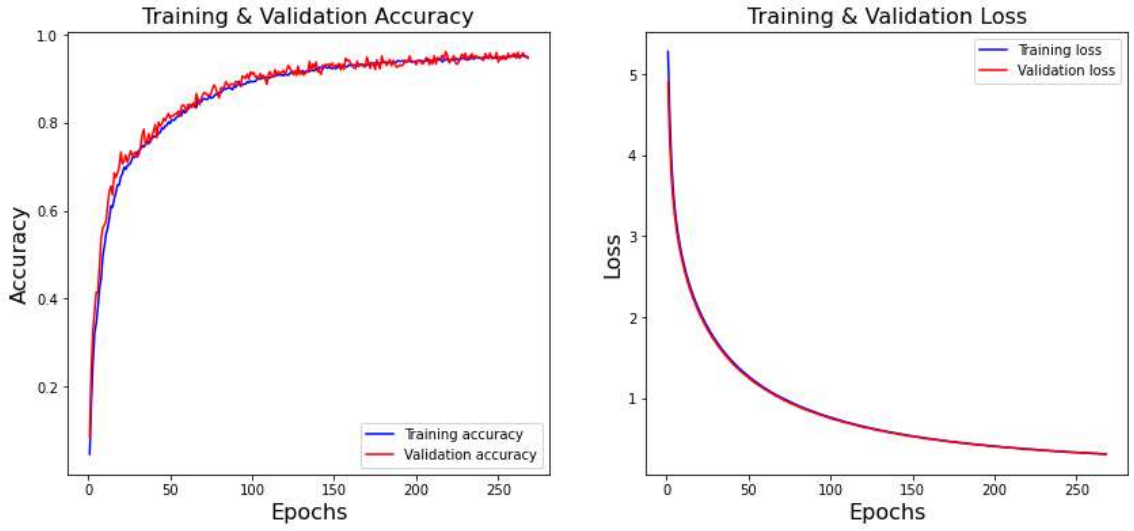

Ilustración 44- Curvas de aprendizaje con el optimizador Adamax y con Early-stopping

El valor final de precisión es de 98.3% y el valor final de pérdida es de 0.168. Además, las curvas tienen las características de un buen aprendizaje; es decir, un fase de aprendizaje rápida seguida por una fase de aprendizaje lento.

Entonces, se puede decir que la red aprendí con este optimizador.

6.3.6 Ftrl

“Follow The Regularized Leader" (FTRL) es un algoritmo de optimización desarrollado en Google para la predicción de la tasa de clics a principios de la década de 2010. Es más adecuado para modelos poco profundos con espacios de características grandes y escasos. El algoritmo es descrito por McMahan (McMahan, 2013).

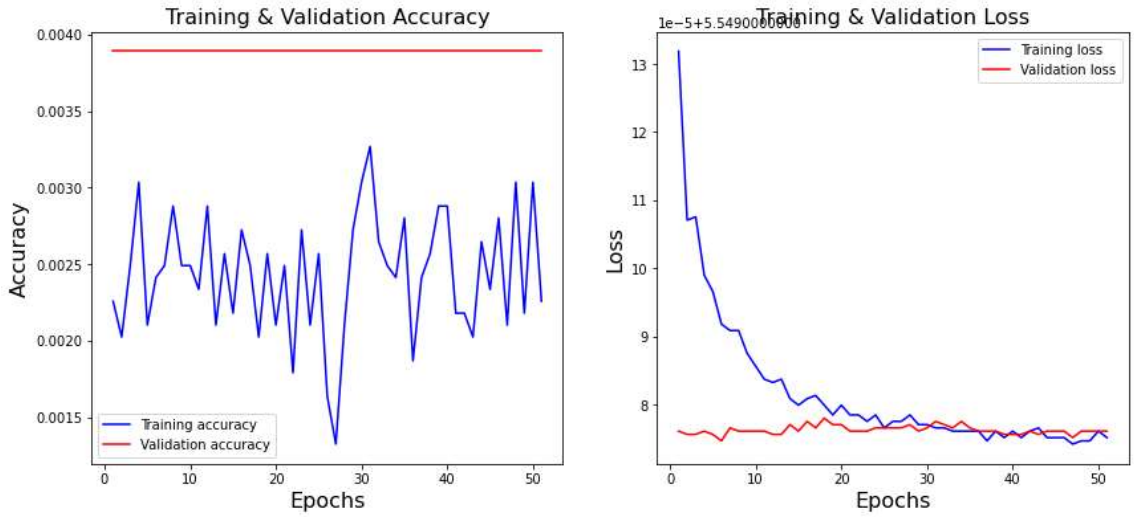

Ilustración 45- Curvas de aprendizaje con el optimizador Ftrl y con Early-stopping.

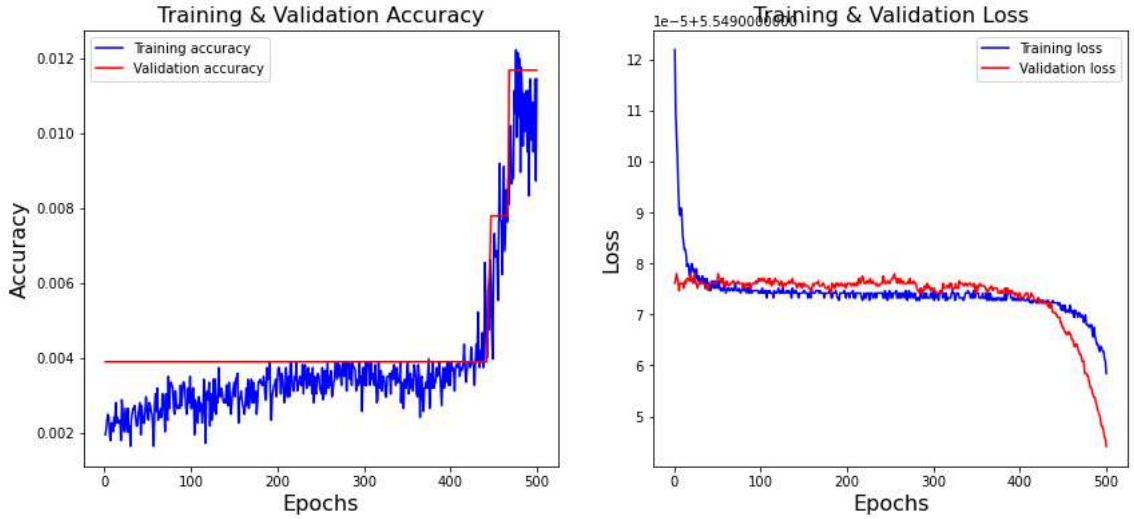

Ilustración 46- Curvas de aprendizaje con el optimizador Ftrl y sin Early-stopping.

Se ha hecho dos redes neuronales para el caso de Ftrl. La Ilustración 45 es la red con el “Early Stopping”. Como se ha acabado el aprendizaje sin que alcance un valor suficiente

para decir que la red aprendí (alrededor de 80%-90%), se ha hecho una red sin “Early Stopping” para ver si el problema viene de los criterios de parada del “Early Stopping” : es la Ilustración 46.

Al final, la red alcance un valor de precisión de 1.2% y de pérdida de 4.32. Además, como en el primer caso, las curvas no tienen formas características de un buen aprendizaje (un aprendizaje rápido seguido por un aprendizaje lento) sino que tiene una curva de precisión con variaciones bruscas y una curva de pérdida casi constante.

Entonces, se puede decir que el optimizador no conviene al aprendizaje de la red.

6.3.7 Nadam

Optimizador que implementa el algoritmo NAdam. Al igual que Adam es esencialmente RMSprop con momentum, Nadam es Adam con momentum Nesterov.

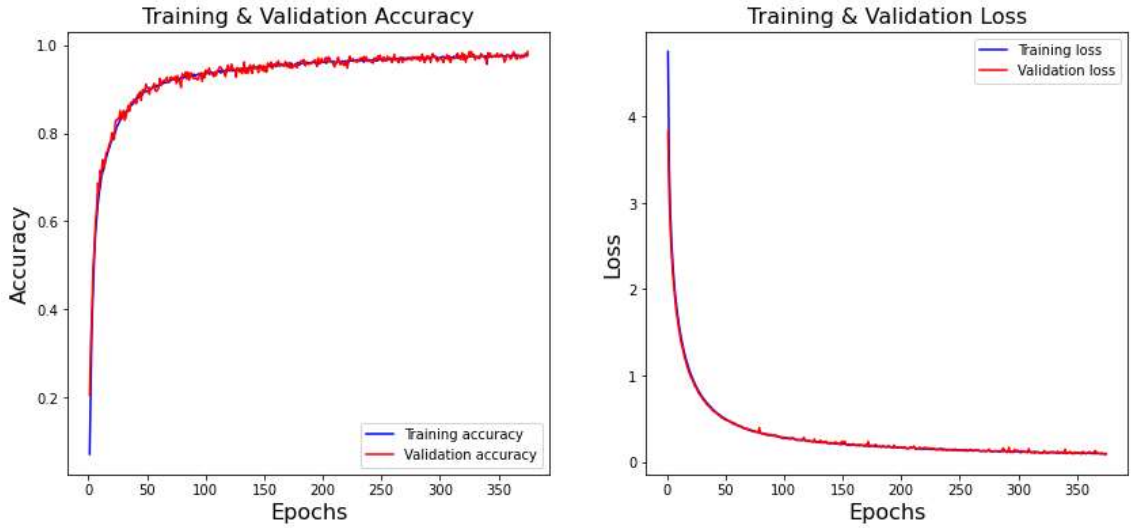

Ilustración 47- Curvas de aprendizaje con el optimizador Nadam y con Early-stopping.

El valor final de precisión es de 97.5% y el valor final de pérdida es de 0.159. Además, las curvas tienen las características de un buen aprendizaje; es decir, una fase de aprendizaje rápida seguida por una fase de aprendizaje lento.

Entonces, se puede decir que la red aprendí con este optimizador.

6.3.8 Gradient descent (SGD)

El descenso de gradiente es el algoritmo de optimización más básico y de primer orden que depende de la derivada de primer orden de una función de pérdida. Calcula en qué dirección se deben alterar los pesos para que la función pueda alcanzar un mínimo. A través de la retropropagación, la pérdida se transfiere de una capa a otra y los parámetros del modelo también conocidos como pesos se modifican en función de las pérdidas para que la pérdida se pueda minimizar.

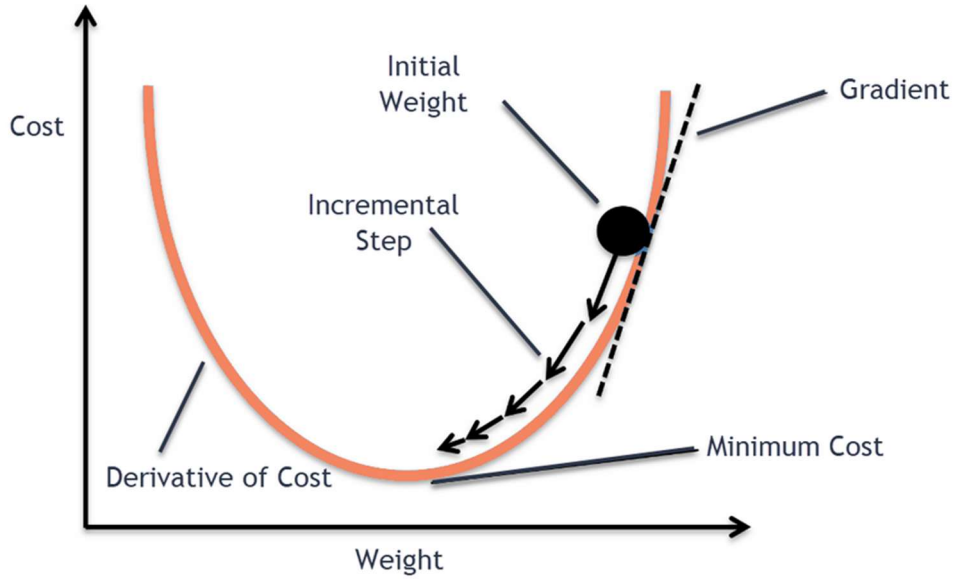

Ilustración 48 - Curva de funcionamiento de Gradient descent (Dawar, 2020)

Este algoritmo es el más básico de los optimizadores. Tiene las desventajas de tomar un conjunto de datos completo de n-puntos a la vez para calcular la derivada para actualizar los pesos, lo que requiere una gran cantidad de memoria. Además, se alcanza el mínimo después de mucho tiempo o nunca se alcanza y se puede pegar en el punto mínimo local o de la silla de montar.

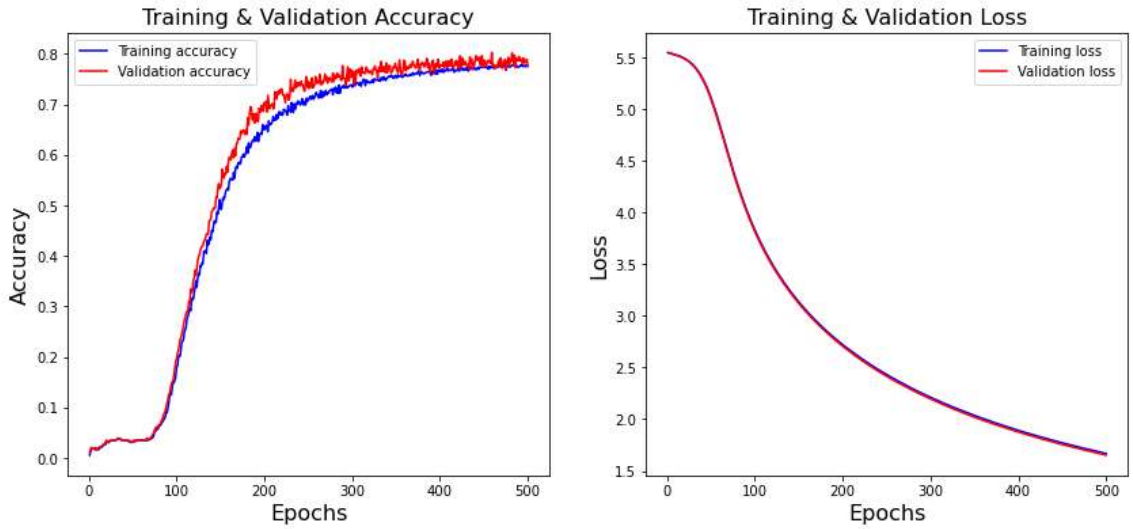

Ilustración 49- Curvas de aprendizaje con el optimizador SGD y con Early-stopping.

El valor final de precisión es de 77.9% y el valor final de pérdida es de 1.65. Además, las curvas tienen las características de un buen aprendizaje; es decir, una fase de aprendizaje rápida seguida por una fase de aprendizaje lento.

Entonces, se puede decir que la red aprendí con este optimizador.

6.3.9 Resumen de las curvas de aprendizaje para las distintas combinaciones de optimizador y conclusión

La Tabla 8 muestra los resultados de precisión, los y el número de epochs final para cada optimizador.

Nombre del Optimizador	Precisión	Loss	Epochs
Adagrad	0.07	4.25	500
RMSprop	0.986	0.289	245
Adaldelta	0.03	5.46	500
Adam	0.972	0.179	234
Adamax	0.983	0.168	273
Ftrl	0.012	4.32	500
Nadam	0.975	0.159	370
SGD	0.779	1.65	500

Tabla 8- Valores finales de Precisión, Loss y Epochs según el Optimizador.

Se pone los datos anteriores en histogramas de Precisión, Loss y numero de Epochs según el optimizador respectivamente en los histogramas Ilustración 50, Ilustración 51, Ilustración 52.

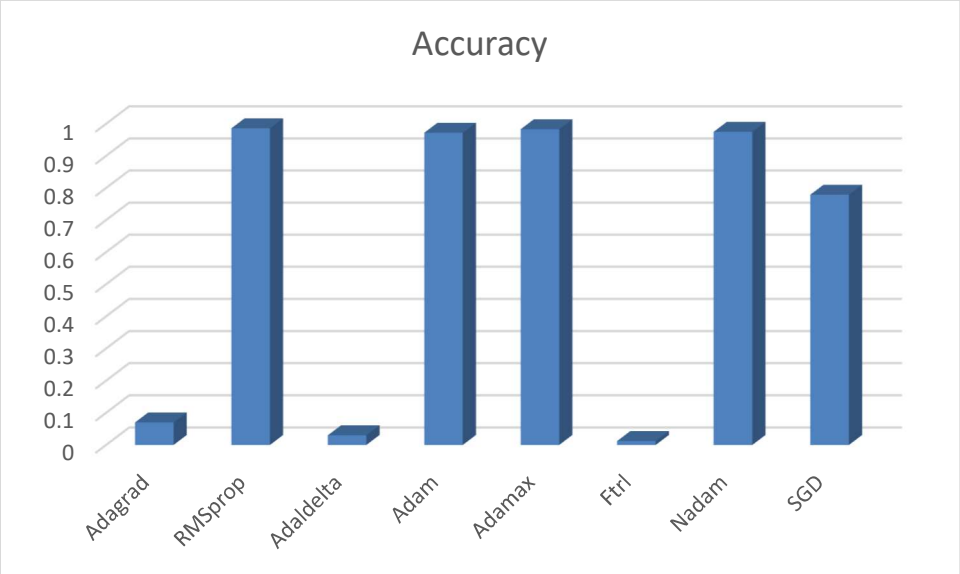

Ilustración 50- Histograma de los valores finales de Precisión según el Optimizador

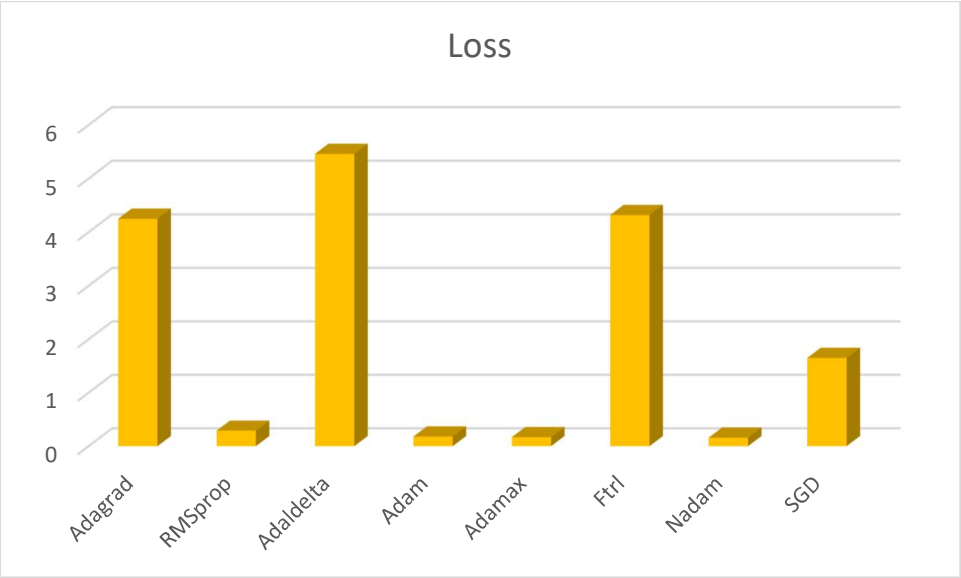

Ilustración 51- Histograma de los valores finales de Loss según el Optimizador

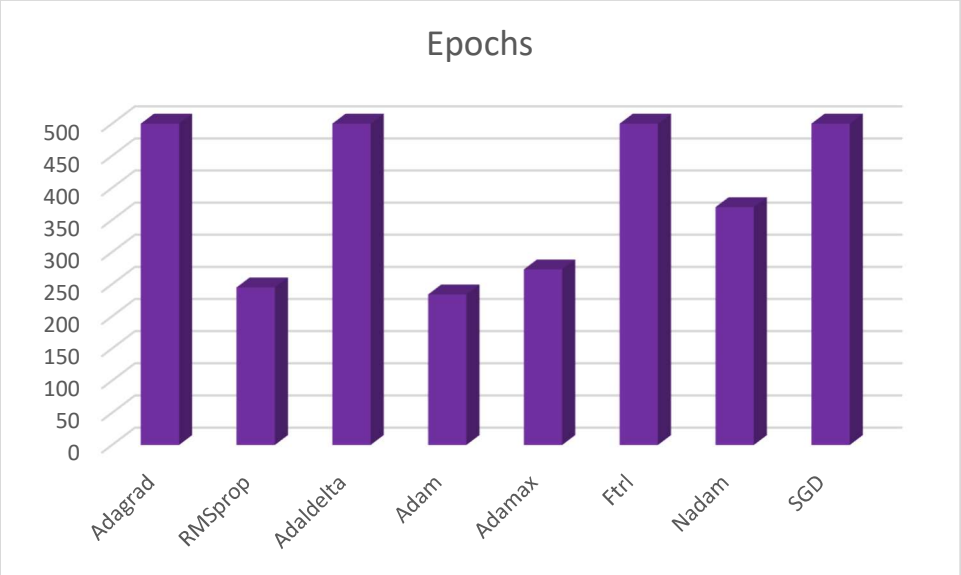

Ilustración 52- Histograma de los valores finales de Epochs según el Optimizador

En conclusión, el análisis de cada curva, se ha dicho que los optimizadores Adagrad, Adaldelta y Ftrl no permiten a la red de aprender. Además, se debe recordar que se ha establecido un criterio de precisión de 95%. El único optimizador (entre los que permiten a la red de aprender) que no permite alcanzar este valor (después de 500 Epochs) es el SGD. No quiere decir que no la red no puede alcanzar un valor de precisión de 95% sino que quiere decir que la red aprende de manera más lenta con este optimizador. En media, los optimizadores superan el 95% de precisión después de 280 Epochs. El optimizador SGD no puede alcanzar este valor después de 500 Epochs. Entonces, se puede eliminar el optimizador SGD en este caso ya que el tiempo de computación es más grande comparado con los otros optimizadores.

Entonces, no se utilizará los optimizadores Adagrad, Adadelta, Ftrl, y SGD para la red.

Ahora, se puede preguntar: ¿Que optimizador elegir?

Se puede notar diferencia de desempeño en los valores de precisión, pérdida y "Nº Epochs".

Nombre del Optimizador	Precisión	Loss	Nº Epochs
RMSprop	0.986	0.289	245
Adam	0.972	0.179	234
Adamax	0.983	0.168	273
Nadam	0.975	0.159	370
Diferencia Media	**0.0055**	**0.045125**	**44.75**

Tabla 9- Diferencias medias de Precisión, Perdida y Epochs según los optimizadores seleccionados.

En la Tabla 9 , se resume las diferencias medias de precisión, pérdida y "Epochs". Las diferencias medias representan una parte despreciable del valor medio de los valores. Entonces, por el carácter aleatorio de la inutilización de las redes neuronales, se podría hacer conclusiones sobre el mejor optimizador con un estudio estadístico con el cálculo de cientos de redes neuronales. No sería renta hacer este estudio ya que el tiempo ganado seria despreciable comparado al tiempo de cálculo para el estudio.

6.4 Variación de la función de activación

Como recordatorio, la función de activación de un nodo en una red de neuronas define la salida de ese nodo dada una entrada o conjunto de entradas.

En esta parte, se estudia las diferentes funciones de activación en las distintas capas. Se recuerda que se define una función de activación para las neuronas en la capa oculta y la capa. Entonces, en esta parte, se estudia todas las combinaciones posibles con las distintas funciones de activación disponibles en Keras. Es decir que se varia la función de activación en la capa oculta y en la capa de salida. Se considera que la combinación de la función X con la función Y en la capa oculta y la capa de salida respectivamente es diferente de la combinación función Y/función X. Las 7 funciones de activación de Keras permiten obtener 49 combinaciones diferentes.

Abajo, se define las funciones de activación de Keras y el desempeño de la red correspondiente.

6.4.1 Función de activación lineal:

En una neurona lineal, la salida depende linealmente de las variables de entrada. Si la neurona tiene como entrada $x_1, x_2\ y\ x_3$,la salida y seria $y = w_1x_1 + w_2x_2 + w_3x_3 + b$, donde $w_1, w_2\ y\ w_3$son los pesos sinápticos para las variables x_1, x_2, x_3 respectivamente y b b es el sesgo en la unidad neuronal.

6.4.2 Binary Threshold:

En una binary threshold neurona (ver figura), si la entrada neta a la neurona excede un umbral especificado, entonces, la neurona se activa; es decir, salidas 1 más que salidas 0. Si la entrada lineal neta a la neurona es $z = w^T + b$ y k es el umbral más allá del cual la neurona se activa, entonces,

$$y = 1 \; si \; z > k \tag{28}$$

$$y = 0 \; si \; z \leq k \tag{29}$$

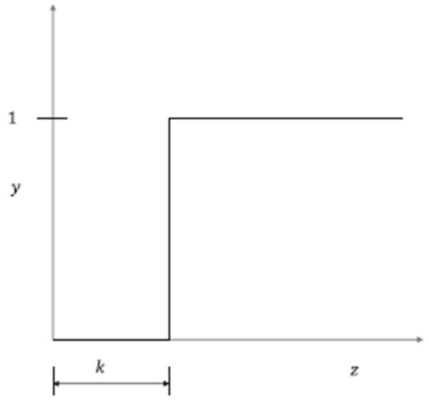

Ilustración 53 - Función Binary Treshold (Layer activation function, s.f.)

6.4.3 Sigmoid:

Se recuerda la Sigmoid que se ha definido en la parte 2 del apartado 2 del capítulo 5

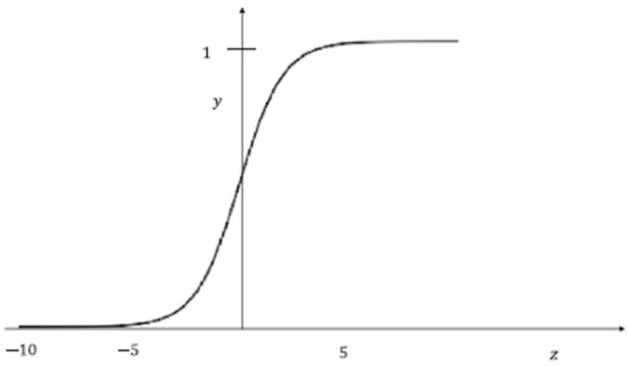

Ilustración 54 - función Sigmoid (Layer activation function, s.f.)

6.4.4 SoftMax:

La función de activación SoftMax es una generalización de la función sigmoide y es más adecuada para problemas de clasificación de múltiples clases. Si hay k clases de

salida y el vector de peso para la clase i es w(i), entonces la probabilidad predicha para la clase *i* dada la entrada vector $x \in \mathbb{R}^{nx1}$ viene dada por lo siguiente:

$$P\left(y_i = \frac{1}{x}\right) = \frac{e^{w^{(i)T}x+b^i}}{\sum_{j=1}^{k} e^{w^{(j)T}x+b^{(j)}}} \tag{30}$$

La particularidad de esta función al contrario de la sigmoid es que utiliza distintos conjuntos de pesos para dos salidas diferentes como se muestra en la figura siguiente:

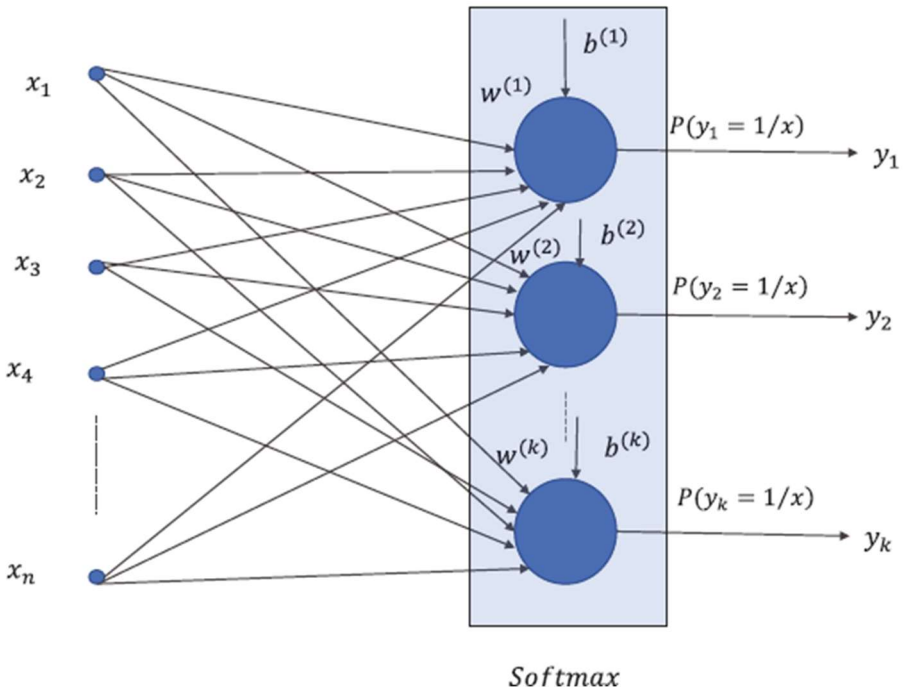

Ilustración 55 - Esquema funcionamiento de la función SoftMax (Layer activation function, s.f.)**.**

6.4.5 Rectified Linear Unit(ReLU) Activation Function

Se recuerda la función de activación RELU que se ha definido en la parte 2 del apartado 2 del capítulo 5

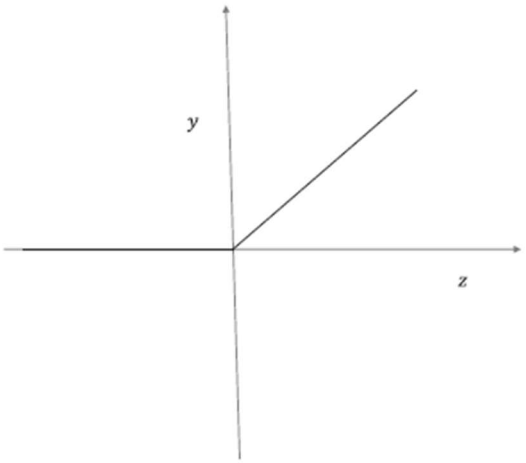

Ilustración 56 - función ReLU (Layer activation function, s.f.)**.**

6.4.6 Tanh

La función se define de la siguiente manera:

$$y = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{32}$$

Donde $z = w^{T}x + b$ es la entrada neta da la función.

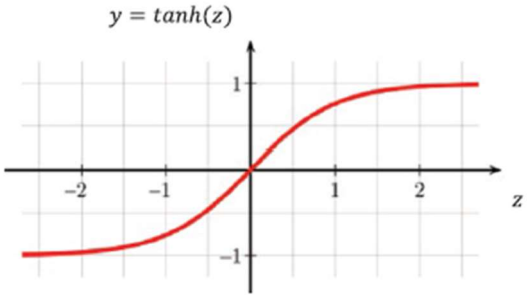

Ilustración 57 - Función Tanh (Layer activation function, s.f.)**.**

Si la entrada neta es un numero positivo grande, $\lim_{z\to\infty} y = 1$

Si la entrada neta es un numero negativo grande, $\lim_{z\to-\infty} y = -1$

Si z=0, $y = 0$

La función de activación sigmoide se satura alrededor de la salida 0. Mientras se entrena la red, si las salidas en la capa están cerca de cero, el gradiente desaparece y el entrenamiento se detiene. La función de activación *tanh* satura a valores -1 y + 1 para la salida y tiene gradientes bien definidos en torno al valor 0 de la salida. Por lo tanto, con las funciones de activación de *tanh* tales problemas de gradiente de fuga se pueden evitar alrededor de la salida 0.

6.4.7 Exponencial

Se recurada la función exponencial con la Ilustración 58:

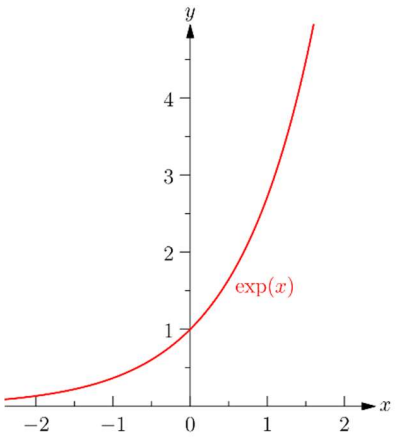

Ilustración 58- función exponencial.

6.4.8 Resultados

Gracias, a las curvas de aprendizaje, se puede determinar las parejas de función de activación que funcionen. Se define los criterios siguientes que permiten decir si la pareja de funciones permite entrenar la red o no:

- Alcanzar un valor de precisión de más de 20%

Se considera que una red que alcance este valor es una red que ha empezado aprender sobre el caso de estudio. Si no se alcance este valor después de 500 Epochs, se considera que la red no aprende.

Se coge los ejemplos de la pareja exponential/ Selu y la de la pareja Selu/Softsign.

El ejemplo Exponencial/Selu es una red que aprende. En la Ilustración 59, se puede ver las curvas de aprendizaje con la fase de aprendizaje rápida y lenta. Al final, alcance un valor de precisión de 98%.

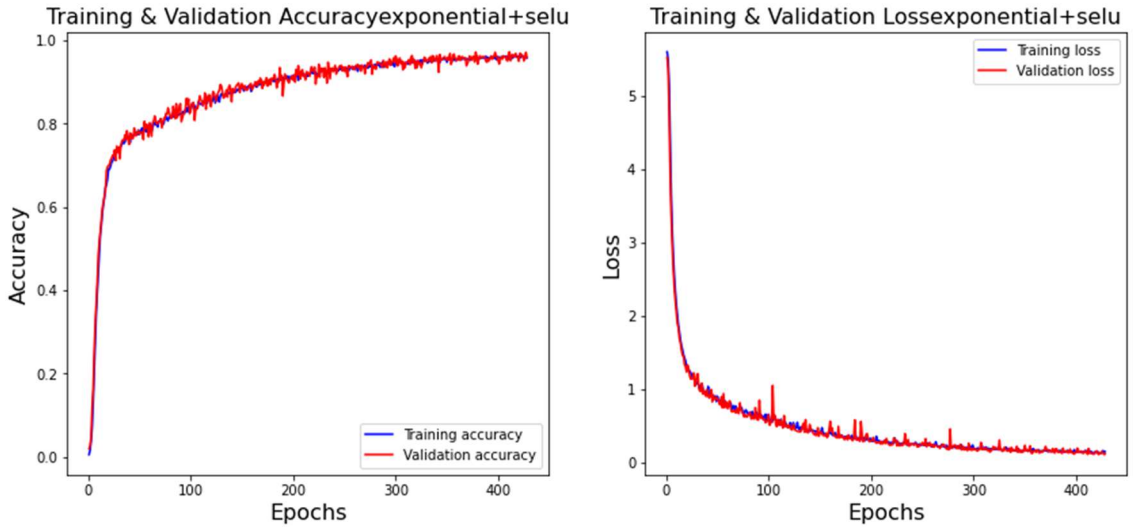

Ilustración 59- curvas de aprendizaje con la pareja exponential + selu.

Al contrario, la pareja Selu/softsign alcanza un valor de precisión menos que 20% lo que no es un valor suficiente para explotar la red o decir que aprendí sobre los datos.

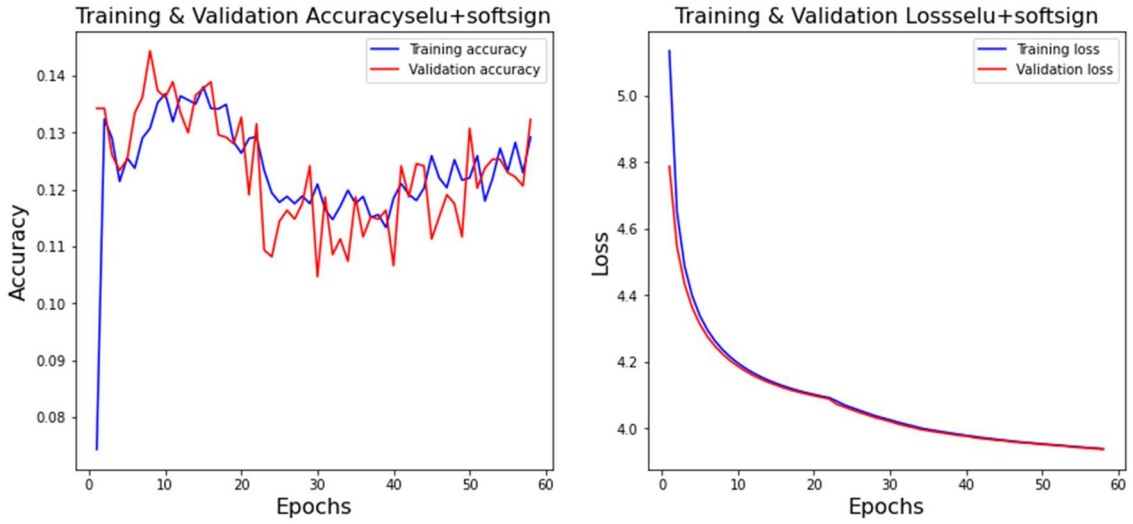

Ilustración 60- curvas de aprendizaje con la pareja selu + softsign.

Las curvas de aprendizaje se encuentran en anexo. La tabla siguiente resume el aprendizaje o no aprendizaje según la pareja de función de activación.

hidden layer function/ last layer function	Relu	Sigmoid	Softplus	Sofsign	Tanh	Selu	Exponential
Relu	0.17	0.943	0.963	0.66	0.72	0.98	0.93
Sigmoid		0.96	0.95	0.6	0.62	0.943	0.93
Softplus		0.92	0.89	0.48	0.5	0.92	0.9
Sofsign	0.278	0.982	0.965	0.62	0.65	0.976	0.948
Tanh	0.25	0.958	0.968	0.667		0.956	0.94
Selu	0.225	0.954	0.974			0.986	0.96
Exponential		0.95	0.965	0.36	0.5	0.97	0.96

Tabla 10- Valores finales de Precisión para cada pareja.

hidden layer function/last layer function	Relu	Sigmoid	Softplus	Sofsign	Tanh	Selu	Exponential
Relu	4.65	0.08	0.09	3.7	3.6	0.08	0.15
Sigmoid		0.23	0.24	3.75	3.68	0.19	0.23
Softplus		0.24	0.23	3.75	3.75	0.295	0.25
Sofsign	4.1	0.078	0.13	3.7	3.6	0.085	0.14
Tanh	4.3	0.09	0.11	3.7		0.096	0.34
Selu	4.43	0.07	0.12			0.08	0.15
Exponential		0.09	0.26	3.75	4.17	0.28	0.34

Tabla 11- Valores finales de Perdida para cada pareja.

Con esta tabla, se concluye que, en cualquier caso, las funciones Tanh y Selu en la capa oculta no permiten a la red de aprender.

Ahora, se selecciona las parejas que permiten alcanzar el criterio de validación de la red, es decir, maximizar la precisión. Dado los datos de la Tabla 11, se pretende alcanzar como mínimo, un valor de 95% de precisión.

Se quitta las funciones RELU, Softsign, Tanh de la tabla anterior (en la capa oculta) y se resalta los valores superiores a 95% de precisión.

hidden layer function/last layer function	Sigmoid	Softplus	Selu	Exponential
Relu	94%	96%	98%	93%
Sigmoid	94%	95%	94%	93%
Softplus	92%	89%	92%	90%
Sofsign	98%	97%	98%	95%
Tanh	96%	97%	96%	94%
Selu	95%	97%	99%	96%
Exponential	95%	97%	97%	96%

Tabla 12- Valores finales de precisión de las parejas seleccionadas. En verde, las parejas que superan el criterio de precisión.

Se puede notar unas combinaciones que no alcancen el valor de 95% de Precisión. Se debe recordar que es por culpa del Early Stopping; no quiere decir que las combinaciones nunca podrían alcanzar el 95% de precisión, sino que, en este caso, no respetan el criterio. Se necesitaría, otra vez, un estudio estadístico sobre el desempeño de cada combinación. Si cada red neuronal se calcula en 4 minutos y se hace 100 veces cada calculo, se necesitaría 13,5 días de computación para seleccionar la red más adecuada. Sin embargo, se puede notar que hay funciones que no alcancen los 95% de Precisión en la mayoría de los casos.

Entonces, no se haría el estudio estadístico ya que el tiempo de computación es demasiado grande. Se supone que los resultados de esta tabla son siempre los mismos pero que se seleccionara las funciones sobre las cuales, en la mayoría de los casos, no se alcance el valor de 95% de precisión con el Early Stopping.

Al final, se quita el caso de la función Expencial en la capa oculta y las funciones Sigmoid y Softplus en la última capa.

hidden layer function/last layer function	Sigmoid	Softplus	Selu
Relu	94%	96%	98%
Sofsign	98%	97%	98%
Tanh	96%	97%	96%
Selu	95%	97%	99%
Exponential	95%	97%	97%

Tabla 13- Valores finales de Precisión para cada pareja que cumplen el criterio de funcionamiento y de Precisión.

Entonces, se queda las combinaciones de la Tabla 13.

Se podría elegir cualquier de las combinaciones disponibles en esta tabla ya que el estudio estadístico no se ha hecho.

6.5 Utilidad del estudio estadístico

En este trabajo, se ha visto que el tiempo de cálculo del aprendizaje de las redes tiene un carácter aleatorio por causa de la inicialización de los pesos en las neuronas. Por eso, sería pertinente hacer un estudio estadístico. Sin embargo, este tipo de estudio cuesta tiempo ya que cada cálculo de red neuronal dura al menos dos minutos. La pregunta de esta parte es: ¿Sería rentable hacer el estudio estadístico?

Suponemos que cada simulación dura 2 minutos (120 segundos). Además, para que el estudio sea relevante, se supone que cada caso se simula 100 veces al mínimo. Si se quiere encontrar el mejor optimizador y la mejor combinación de funciones de

activación, se debería simular 8 casos para el optimizador y 49 casos para la combinación de función de activación. Entonces, se necesitaría simular 570 casos.

Al final, el tiempo de cálculo para hacer el estudio será de 19 horas. Sin embargo, la precisión ganada por el estudio no permitiría alcanzar valores de precisión mas elevadas de las se alcancen ya. Entonces, sería más rentable elegir una de las combinaciones que permiten alcanzar valores de precisión de más de 95% y aumentar el número de iteraciones en la simulación (por ejemplo, de 200 Epochs) para que llegue un valor de precisión suficiente; aumentando de esta manera el tiempo de cálculo de 45 segundos lo que representa 0.06% del tiempo del estudio estadístico.

CAPITULO 7: CONCLUSIONES

7 CONCLUSIONES

En conclusión, se ha utilizado la biblioteca FEMFUSION para recoger casos de estudio de una perturbación neutrónica en cada celda de un reactor de 257 celdas. Sabiendo que los detectores que recogen estos datos tienen una precisión que no puede llegar al 100%, se genera un dataset tomando en cuenta la imprecisión de los detectores.

Luego, se ha definido el tipo de problema de este estudio. En efecto, se debe encontrar la celda de donde proviene la perturbación; es decir que el problema es un problema de localización con una salida de 257 valores posible (0 hasta 256). Esta definición de problema ha permitido fijar parámetros como el número de detectores (4) el número de neurona a la entrada y la salida (8 y 257) y la función de perdida (*Sparse categorical crossentropy*). Se ha elegido los otros parámetros de manera arbitraria para generar una primera red neuronal funcional con sus curvas de aprendizaje

Luego, se estudió la optimización del tiempo de cálculo para obtener una red neuronal. El primer parámetro estudiado fue el número de neuronas en la capa oculta. Se ha generado casos variando el número de neuronas en potencia de 2 desde 8 hasta 256 ya que los expertos se acuerdan para decir que permite obtener tiempos de cálculo más rápidos. Se generó 10 veces cada caso y cogiendo la media de los valores finales de precisión, perdida y de tiempo para limitar el carácter aleatorio del cálculo de una red. Se vio que 10 casos no permiten concluir sobre un valor de numero de neuronas en la capa oculta sino 2 ya que siempre se nota el carácter aleatorio de la inicialización de la red. Entonces, se eligió 128 neuronas en la capa oculta de la red ya que sus valores finales de precisión, perdida y tiempo de cálculo son unas de los mejores de los casos generados.

Además, se ha reducido la cantidad de datos entrando en la red. Se intentó entrenar la red de dos maneras, sin información sobre las fases y sin información sobre los módulos. Al final, se podía entrenar la red solamente con la información sobre los módulos dividiendo por dos la cantidad de datos a la entrada (4 en lugar de 8) y pasando el número de neuronas a la entrada a 4.

A continuación, se ha variado la cantidad de detectores en el reactor de dos maneras. En primer lugar, se ha estudiado la influencia de la cantidad de detectores en el reactor con una distinción entre los casos al nivel de separación de celdas entre detectores. Se ha llegado a la conclusión que cuanto más detectores hay, mayor es el

valor de precisión final y menor el valor de perdida y de tiempo de cálculo. Sin embargo, en el caso de un reactor real, el número de detectores es fijado. Entonces, se estudió en segundo lugar un caso real de posición de detectores. Se aumentó el número de detectores y cambió la disposición de ellos para probar el funcionamiento de la red en un caso real.

Luego, se buscó la influencia del optimizador en el aprendizaje de la red. Se vio que se puede elegir un optimizador entre RMSprop, Adam, Adamax, Nadam pero los optimizadores Adagrad, Adadelta, Ftrl y SGD no permiten a la red de aprender.

De manera análoga, se buscó la combinación de funciones de activación en la capa oculta y la capa de salida. Se encontré 15 combinaciones posibles que permiten entrenar la red.

En las variaciones de los parámetros anterior, se concluye que se necesita un estudio estadístico para encontrar lo mejor de los parámetros en cada caso. Sin embargo, haciendo un estudio con 100 iteraciones, se determinó que no sería rentable hacer el estudio estadístico comparado por el tiempo ganado.

Al final, este Trabajo de Fin de Máster ha triunfado su objetivo inicial, es decir, diseñar una red neuronal para encontrar fuentes de perturbaciones en reactores nucleares. Se ha hecho un estudio de la red y sus parámetros para intentar reducir el tiempo de computación. Sin encontrar los mejores parámetros, se ha establecido un panel de parámetros que permiten obtener una red neuronal para resolver el problema en un tiempo razonable ya que sería necesario hacer un estudio estadístico para determinar los mejores parámetros.

Al final, la resolución del caso en 2 dimensión ha permitido entender el funcionamiento de las redes neuronales y sus distintos parámetros. El siguiente paso lógico de este TFM sería el diseño de una red neuronal en un caso completo para encontrar el tipo y la celda de origen de la perturbación en un caso 4 dimensión: 3 de espacio y 1 de tiempo.

BIBLIOGRAFÍA

8 BIBLIOGRAFÍA

A. Vidal-Ferràndiz, D. G. (2021). A finite element method for neutron noise analysis in hexagonal reactors. *PHYSOR2020 – International Conference on Physics of Reactors: Transition to a Scalable Nuclear.*

Dawar, H. (2020). *Stochastic Gradient Descent*. Obtenido de medium.com: https://medium.com/analytics-vidhya/stochastic-gradient-descent-1ab661fabf89

Diederik P. Kingma, J. B. (2015). *Adam: A Method for Stochastic Optimization.* San Diego: 3rd International Conference for Learning Representations.

Électricité en Espagne. (6 de noviembre de 2022). Obtenido de Wikipedia: https://fr.wikipedia.org/wiki/%C3%89lectricit%C3%A9_en_Espagne

Energía nuclear. (s.f.). Obtenido de Wikipedia: https://fr.wikipedia.org/wiki/%C3%89nergie_nucl%C3%A9aire

Energía nuclear en España. (5 de febrero de 2013). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Energ%C3%ADa_nuclear_en_Espa%C3%B1a?oldid=63557202

Fonctionnement central PWR. (s.f.). Obtenido de EDF: https://www.edf.fr/groupe-edf/espaces-dedies/l-energie-de-a-a-z/tout-sur-l-energie/produire-de-l-electricite/le-fonctionnement-d-une-centrale-nucleaire

Francesco Caliva, F. S. (2018). A deep learning approach to anomaly detection in nuclear reactors. *2018 International joint conference on neural networks (IJCNN).*

Francesco Calivá, P. (2020). Anomaly detection in nuclear reactors using deep learning., (pág. 36).

IBM. (s.f.). Obtenido de Overfitting: https://www.ibm.com/cloud/learn/overfitting

Layer activation function. (s.f.). Obtenido de Keras: https://keras.io/api/layers/activations/

McMahan, H. B. (2013). *Ad click Prediction: A View from the Trenches.*

Moolayil, J. (2019). *Learn Keras for Deep Neural Networks.*

Nacional - Seguimiento de la demanda de energía eléctrica. (08 de octubre de 2022). Obtenido de Demanda.ree: https://demanda.ree.es/visiona/peninsula/nacional/acumulada/2022-10-08

Pattanayak, S. (2017). *Pro deep learning with tensorflow.*

Treiner, J. (18 de octubre de 2017). Obtenido de ResearchGate: https://www.researchgate.net/figure/Un-exemple-de-production-consommation-journaliere-delectricite-On-remarque-que-les_fig3_328164342

Vidal-Ferràndiz, A. (2022). Localizing perturbations in Pressurized water reactors using one-dimensional deep convolutional neural networks. *Sensors*.

PARTE 2: PRESUPUESTO

Este apartado se dedica al cálculo de los costes para hacer este Trabajo de Fin de Máster. Se identificará los costes humanos y los costes de las herramientas utilizadas. A notar que el siguiente presupuesto es una estimación.

1- COSTES HUMANOS

En esta parte, se identifica las personas y las horas dedicadas al estudio.

Las personas que han participado a este trabajo son:

- El estudiante que realizó el TFM: Quentin LEC'HVIEN
- El tutor del estudiante: Antoni VIDAL

Se ha empezado el TFM el 1 marso de 2022 y se le ha acabado el 31 de octubre 2022. Se distinta 3 fases del trabajo:

- Fase 1 desde el 01/03/2022 hasta el 30/04/2022: fase de aprendizaje de las redes neuronales y de los estudios correspondiente a la detección de perturbaciones en redes neuronales.
- Fase 2 desde el 01/05/2022 hasta el 31/07/2022: fase de construcción de la red y su optimización
- Fase 3 desde el 01/08/2022 hasta el 31/10/2022: fase de redacción del TFM

Se define el coste de mano a 20 €/h para el alumno como ingeniero y un coste de 45 €/h para el tutor.

1- Fase1

En esta fase, el alumno se dedicó a la búsqueda de información sobre las redes neuronales y los distintos estudios sobre las perturbaciones en reactores nucleares. El tutor ayudó el alumno para entender el problema y responder a sus respuestas. El alumno se dedicó al estudio

La Tabla 14define las horas dedicadas por el alumno y el tutor y La Tabla 15 muestra el coste de mano de obra para la fase 1.

Concepto	Horas Alumno	Horas Tutor
presentación del trabajo	3	3
planificación del trabajo	2	2
formación FEMFUSION	10	2
formación en Redes Neuronales	15	
formación en Keras	50	
Total	**80**	**7**

Tabla 14- Horas trabajadas por el alumno y el tutor en la fase 1

Designación	Horas	Precio (€/h)	Total (€)
Alumno	80	20	1600 €
Tutor	7	45	315 €
		Total	**1915 €**

Tabla 15- Coste de mano de obra por la fase 1

2- Fase 2

El alumno se dedicó en esta fase, a la construcción de la red neuronal y el estudio de su optimización. El tutor ayudó el alumno para la construcción de la red.

De manera análoga a la fase 1, la Tabla 16muestra las horas trabajadas y la Tabla 17, el precio correspondiente.

Concepto	Horas Alumno	Horas Tutor
construcción de la red neuronal	60	5
Estudio de los parámetros	180	0
Total	**240**	**5**

Tabla 16- Horas trabajadas por el alumno y el tutor en la fase 2

Designación	Horas	Precio (€/h)	Total (€)
Alumno	240	20	4800 €
Tutor	5	45	225 €
		Total	**5025 €**

Tabla 17- Coste de mano de obra por la fase 2

3- Fase 3

El alumno se dedicó a la redacción del TFM y el tutor ha hecho las revisiones del trabajo.

De manera análoga a la fase 1, la Tabla 18muestra las horas trabajadas y la Tabla 19, el precio correspondiente.

Concepto	Horas Alumno	Horas Tutor
redacción del documento	60	0
1er Revisión	40	5
2e Revisión	20	5

Total	**120**	**10**

Tabla 18- Horas trabajadas por el alumno y el tutor en la fase 3

Designación	Horas	Precio (€/h)	Total (€)
Alumno	120	20	2400 €
Tutor	5	45	225 €
		Total	**2625 €**

Tabla 19- Coste de mano de obra por la fase 3

2- COSTE DEL ORDENADOR

El alumno utilizó su propio ordenador y el ordenador del departamento de matemáticas para la realización del trabajo. Se considera que el ordenador del alumno tiene una vida útil de 6 años y el del departamento de matemáticas, una de 10 años. Durante los 7 meses del proyecto, el alumno utilizó el ordenador de departamento de matemáticas durante 2 meses, lo cual se traduce en una amortización de 1.67%. De manera análoga, se calcula una amortización para la amortización del ordenador del alumno de 6.94%.

Entonces, obtenemos la Tabla 20 de coste de los ordenadores

Producto	Cantidad	Coste	Amortización	Depreciación
Ordenador del alumno	1u	800	6.94%	55 €
Ordenador del departamento de matemáticas	1u	750	1.67%	12 €
			Total	**67 €**

Tabla 20- Coste de los ordenadores

3- COSTE DEL SOFTWARE

En esta parte, se cuenta los costes relacionados con los softwares utilizados para el trabajo listados en la Tabla 21. Esta tabla muestra que el coste total es de 0 euros ya que las bibliotecas utilizadas de Python son gratuitas y la universidad permite tener las licencias Microsoft Word y Excel gratuitas.

Producto	Cantidad	Coste
Microsoft Word	1u	0 €
Microsoft Excel	1u	0 €
Anaconda / Spider / Python	1u	0 €
Biblioteca FEMFUSION	1u	0 €
Biblioteca Keras	1u	0 €

Tabla 21- Coste del software

4- COSTE TOTAL

Al final, se añade los costes calculados anterior en la Tabla 22 para obtener el presupuesto del proyecto antes impuestos y beneficios.

PARTE DEL PRESUPUESTO	COSTE TOTAL (€)
MANO DE OBRA	**9565 €**
FASE 1	1915 €
FASE 2	5025 €
FASE 3	2625 €
HARDWARE	**67 €**
SOFTWARE	**0 €**
TOTAL	**9632 €**

Tabla 22- Presupuesto base del proyecto

Luego, se añade el coste de los gastos generales y el beneficio industrial para obtener el presupuesto total sin impuestos:

Parte	Coste (€)
Presupuesto base	9632
Gastos generales (15%)	1445
Beneficio Industrial (6%)	578
Presupuesto sin impuestos	11655

Tabla 23- Coste total antes de impuestos

Teniendo en cuenta los impuestos:

Parte	Coste (€)
Presupuesto sin impuestos	11655
IVA (21%)	2448
Presupuesto con impuestos	14103

Al final, el coste de este trabajo es de 14103 euros.

ANNEXO

1. RELACIÓN DEL TRABAJO CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE DE LA AGENDA 2030

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No procede
OSD 1. **Fin de la pobreza.**			X	
OSD 2. **Hambre cero.**				X
OSD 3. **Salud y bienestar.**			X	
OSD 4. **Educación de calidad.**		X		
OSD 5. **Igualdad de género.**				X
OSD 6. **Agua limpia y saneamiento.**				X
OSD 7. **Energía asequible y no contaminante.**	X			
OSD 8. **Trabajo decente y crecimiento económico**		X		
OSD 9. **Industria, innovación e infraestructuras.**	X			
OSD 10. **Reducción de las desigualdades.**				X
OSD 11. **Ciudades y comunidades sostenibles.**	X			
OSD 12. **Producción y consumo responsables.**				X
OSD 13. **Acción por el clima**.		X		
OSD 14. **Vida submarina.**				X
OSD 15. **Vida de ecosistemas terrestres.**				X
OSD 16. **Paz, justicia e instituciones sólidas.**				X
OSD 17. **Alianzas para lograr objetivos.**				X

Este trabajo permite encontrar fuentes de perturbaciones en reactores nucleares. La detección de las perturbaciones se hace mediante datos simulados, cogiendo los datos pertinentes y calculando una red neuronal. Permite mejorar la seguridad de los reactores nuclear sin poner en riesgo la población y su entorno. Entonces, se considera que está relacionado con el objetivo ODS 7 ya que la energía nuclear es la base de producción de electricidad en números países. Además, su infraestructura y el tratamiento de los residuos radioactivos permite asegurarse de la no contaminación del entorno.

Además, la energía nuclear es una de la más utilizadas en el mundo y tiene una potencia considerable emitiendo pocas cantidades de CO2 por lo que permite mantener el desarrollo sostenible de las ciudades: es el OSD 11.

En fin, la infraestructura de las centrales nucleares con la infraestructura de tratamiento de los residuos nucleares es complejas y completas. Su diseño necesita expertos, ingenieros y mano de obra para su mantenimiento. De esta manera, la localización de las perturbaciones se relaciona con el ODS9.