Escuela de ingeniería en Fuenlabrada

Grado en ingeniería aeroespacial en vehículos aeroespaciales

TRABAJO FIN DE GRADO

ESTUDIO DE LA VIABILIDAD DEL USO DE REDES NEURONALES PARA LA EXPANSIÓN DE GEOMETRÍAS DE ASTEROIDES EN ARMÓNICOS ESFÉRICOS

Autor: Juan José González Reig

Tutor: Pablo Solano López

Cotutor: Hodei Urrutxua Cereijo

Curso académico 2023/2024

Dedicado a

mi familia

Agradecimientos

En primer lugar, quiero agradecer a mis padres por todo el apoyo que he recibido desde siempre y, en especial, durante mi etapa en la universidad. Sé que no ha sido fácil y que han tenido que hacer muchos sacrificios para que yo pudiera estudiar lo que quería. En general, agradecer a toda mi familia que siempre ha estado ahí, han creído en mí y han hecho que en todo momento me sienta arropado por ellos. A mi abuela Mila, por ser la persona más bondadosa y altruista que conozco. Me hubiera encantado que el resto de mis abuelos hubieran estado para presenciar este momento, pero sé que estarían muy orgullosos de mí y de la persona en la que me he convertido desde que ellos no están.

A mis amigos Alejandro, Javier y Jonathan por estar a mi lado siempre para celebrar los buenos momentos pero sobretodo por estar en los malos y evitar que me caiga. Sin vosotros esto no hubiera sido posible.

A mi perro Lolo, por ser la alegría de la casa y por haber estado innumerables horas a mi lado mientras hacía este proyecto (mientras escribo esto está aquí a ver si le tiro la pelota como siempre).

En general, gracias a todo el que creyó en mí en algún momento y me alentó a lo largo de mi camino. Este trabajo no es más que el resultado de la colaboración y ayuda de multitud de personas.

Por los que están y por los que estuvieron.

Resumen

Debido a la complejidad de obtener los coeficientes resultantes de la expansión en armónicos esféricos de cualquier geometría y al gran potencial de las redes neuronales, en este proyecto se estudiará la combinación de ambas temáticas. En concreto, se abordará el análisis de geometrías de asteroides con el objetivo de aprovechar la rápida capacidad de respuesta de las redes neuronales para llevar a cabo dicha expansión en armónicos esféricos en misiones espaciales.

Para realizar este estudio, se hará una introducción de los armónicos esféricos y de las redes neuronales. A continuación, se escogerán los tipos de estructura con mayor compatibilidad para el propósito del proyecto. Después, se explicará de manera detallada el código desarrollado, el cual, a *grosso modo*, se encargará de: producir los asteroides de forma aleatoria que se usarán en el proyecto, estandarizar y procesar los datos, crear y entrenar las redes neuronales y calcular las métricas de dichas redes.

Posteriormente, se realizarán varios análisis paramétricos de las redes neuronales escogidas con el objetivo de encontrar relaciones y dependencias entre dichos parámetros. Además, también se obtendrá de esta manera unas combinaciones de parámetros que produzcan mejores métricas.

Finalmente, se hará una comparativa entre las redes neuronales estudiadas y se concluirá si es viable o no el uso de redes neuronales con el fin de expandir geometrías de asteroides en armónicos esféricos. Además, se establecen ciertos posibles futuros trabajos y aplicaciones prácticas del proyecto realizado.

Abstract

Due to the complexity of obtaining the coefficients resulting from the spherical harmonics expansion of any geometry and the great potential of neural networks, this project will study the combination of both themes. Specifically, the analysis of asteroid geometries will be addressed with the aim of taking advantage of the fast response capacity of neural networks to perform such spherical harmonic expansion in space missions.

In order to carry out this study, an introduction to spherical harmonics and neural networks will be made. Next, the types of structure with the best compatibility for the project's purpose will be chosen. Subsequently, a detailed explanation of the developed code will be given, which, in a broad sense, will be in charge for: generating the randomly shaped asteroids to be used in the project, standardizing and processing the data, creating and training the neural networks, and calculating the metrics of these networks.

Following this, several parametric analyses of the chosen neural networks will be performed in order to find relationships and dependencies among these parameters. Additionally, this approach will yield parameter combinations that produce better metrics.

Finally, a comparison will be made between the studied neural networks, determining whether the use of neural networks to expand asteroid geometries into spherical harmonics is viable or not. Furthermore, some possible future works and practical applications of the project carried out are established.

Índice general

Capítulo 1

Introducción

El estudio de los asteroides y su movimiento orbital es uno de los campos más estudiados de la astronomía actual. Los asteroides son los vestigios de la formación de nuestro sistema solar de hace unos 4600 millones de años [1]. Estos están compuestos por pedazos de rocas de silicato y arcilla, junto con algunos metales como el hierro o el níquel. El estudio de estos cuerpos rocosos es de gran interés puesto que a través de ellos se puede llegar a entender las primeras épocas de la formación de nuestro sistema solar. Además, en particular el estudio de los asteroides cercanos a la Tierra (NEA, *near-Earth asteroids*) es de especial importancia por la posibilidad de impactos catastróficos sobre la superficie terrestre. De hecho, la colisión de un asteroide cercano a la Tierra de unos $12km$ de diámetro fue el causante del final del Cretácico hace 65 millones de años [2].

La mayoría de los conocimientos que se tiene sobre estos cuerpos proviene de observaciones desde la Tierra, análisis de relaciones con los meteoritos y de la modelización teórica de la dinámica, estructura y evolución térmica de los asteroides [3]. Sin embargo, en las últimas décadas, estos conocimientos se han visto complementados, mejorados y, en algunos casos, revolucionados por las misiones espaciales dedicadas. Estas misiones las componen, de forma genérica, telescopios que orbitan la Tierra y, de forma más concreta, misiones *in situ* de exploración de asteroides. En las últimas décadas se han realizado un total de 8 misiones de exploración, donde la mayoría de estas han sido misiones de *flyby*. Además de esto, también se han realizado dos misiones de órbita y una de órbita y recogida de muestras [4]. El resultado de estas misiones ha permitido, entre muchos otros resultados, el nacimiento de la geología de los asteroides, mediante la cual se puede conocer la historia de estos a través de los

cráteres superficiales originados por colisiones. Por otro lado, estas misiones *in situ*, también han conseguido la obtención de información relativa a la geometría de los asteroides con una precisión muy superior a la que se puede obtener desde observaciones de la Tierra. Un gran ejemplo de este tipo de misiones es la misión NEAR (*Near Earth Asteroid Rendezvous*) realizada sobre el asteroide 433 Eros [5] con el objetivo de determinar su geometría, gravedad y estado rotacional. Antes de la realización de esta misión, poco era conocido del asteroide Eros, sólo su órbita, velocidad de rotación y orientación de sus polos, todo conseguido mediante observaciones telescópicas terrestres. Sin embargo, todo esto pudo ser documentado gracias a esta misión y, en concreto, al instrumento NEAR *laser rangefinder* (NLR). El cual consiguió obtener, entre otras cosas, los coeficientes de los armónicos esféricos que permiten describir la geometría del 433 Eros, los cuales serán de gran importancia en este proyecto.

Todo lo expuesto pone en evidencia los grandes esfuerzos realizados para poder determinar la geometría de un asteroide. Por ello, el objetivo de este proyecto es estudiar la viabilidad del uso de redes neuronales mediante *Machine Learning* (ML) para la obtención de los coeficientes de los armónicos esféricos de asteroides a partir de una nube de puntos.

Esta expansión en armónicos esféricos de una nube de puntos representativa de la superficie de un asteroide se puede realizar de varias maneras. Uno de estos métodos es mediante el uso de la transformada de Fourier, la cual permite una rápida obtención de los armónicos esféricos. Sin embargo, este método puede ser susceptible a errores numéricos debido a la precisión finita de las operaciones de coma flotante. Otra de las formas en las que se puede calcular los armónicos esféricos a partir de una nube de puntos y que, además, es más precisa que la transformada de Fourier es el uso de técnicas numéricas de integración directa como el método de Montecarlo. El problema de este método es que puede llegar a ser computacionalmente muy costoso.

Por otro lado, actualmente existe una gran capacidad computacional que ha permitido la creación y el desarrollo de la Inteligencia Artificial. Una rama de esta tecnología es el *Machine Learning*, mediante el cual se pueden entrenar diversas formas de redes neuronales para que cumplan ciertas tareas que se desee. Estos entrenamientos se consiguen con una gran cantidad de datos y mediante unas redes neuronales cuya estructura se inspira en las

redes neuronales del cerebro. De esta manera, los modelos de redes neuronales son capaces de aprender a realizar una tarea.

Esta tecnología cuenta con el potencial de proporcionar una forma eficiente y precisa de calcular los coeficientes de los armónicos esféricos para cualquier geometría de asteroide dada. Es eficiente debido al hecho de que una vez la red neuronal ha sido entrenada, la obtención de una predicción de ésta, es decir, el resultado de introducir nuevos datos para que la red desempeñe su tarea, se realiza de forma casi inmediata. La precisión de estas redes neuronales dependerá del conjunto de datos utilizado para sus entrenamientos y de las estructuras internas de las propias redes empleadas.

1.1. Objetivos generales

Los objetivos generales de este proyecto son la selección y el estudio de viabilidad de distintas redes neuronales para que desempeñen la tarea de convertir geometrías de asteroides, en forma de nubes de puntos, en los coeficientes de los armónicos esféricos propios de dichos asteroides. Estos armónicos esféricos permiten la representación de esas geometrías con una precisión variable dependiente del número de armónicos empleados. Además, con ello también se puede caracterizar de la misma manera el potencial gravitatorio de los asteroides.

1.2. Objetivos específicos

En primer lugar, como objetivo específico se tiene la selección de las estructuras de redes neuronales que se van a utilizar para el proyecto. Previamente a eso, se realizará un desarrollo de cómo se puede hacer una expansión en armónicos esféricos de una superficie cualquiera. Esto, además, ayudará para seleccionar las mejores estructuras posibles de las redes.

El siguiente objetivo específico será la creación de una base de datos de asteroides que permita el entrenamiento y evaluación de las redes neuronales generadas. Seguidamente, se desarrollará un código capaz de utilizar esa base de datos para entrenar a las redes neuronales y analizar su precisión.

Una vez se tenga esa base, el siguiente objetivo será realizar estudios paramétricos de las redes neuronales para determinar el impacto de estos en los rendimientos de las redes.

El último objetivo será realizar una comparativa de los rendimientos de las diferentes redes neuronales entrenadas bajo la misma base de datos. Para ello, debido al hecho de que se tendrá diferentes estructuras de redes neuronales y estas se entrenan de distintas formas, se procurará llegar con ellas al mismo rango de precisión para poder compararlas.

1.3. Estructura de la memoria

La estructura de la memoria está representada de forma gráfica en la Figura 1.1. En el Capítulo 1 se ha realizado una introducción al proyecto realizado. En concreto, se hecho una puesta en contexto del problema abordado y su importancia. Además, se han estipulado los objetivos generales y específicos del proyecto.

En el Capítulo 2 se introducen los armónicos esféricos. En particular, primero se introducen desde el punto de vista del potencial gravitatorio y, después, se especifica cómo se realiza una expansión en armónicos esféricos de la geometría de un cuerpo.

A continuación, en el Capítulo 3 se realiza una breve introducción al *Machine Learning*. Además, se explica cómo funciona internamente una red neuronal y los distintos tipos de estructuras de éstas. Finalmente, se eligen las estructuras de las redes neuronales que se van a analizar y comparar para cumplir los objetivos del proyecto.

En el Capítulo 4 se presenta la estructura general del código del proyecto. En concreto, se explicará en detalle cómo se genera la base de datos de los entrenamientos, cómo se construyen las distintas redes neuronales seleccionadas y cómo se van a evaluar los rendimientos de éstas.

Posteriormente, en el Capítulo 5 primero se presentarán los distintos análisis paramétricos e hiperparamétricos realizados a las redes neuronales escogidas. Después, se hará una comparativa final entre las redes seleccionadas. Para ello, se entrenarán las redes neuronales de la forma más similar posible bajo la misma base de datos y se comprobará cuáles son las que mejores resultados presentan y sus diferencias.

Finalmente, en el Capítulo 6 se comentarán las conclusiones del proyecto. Dados los

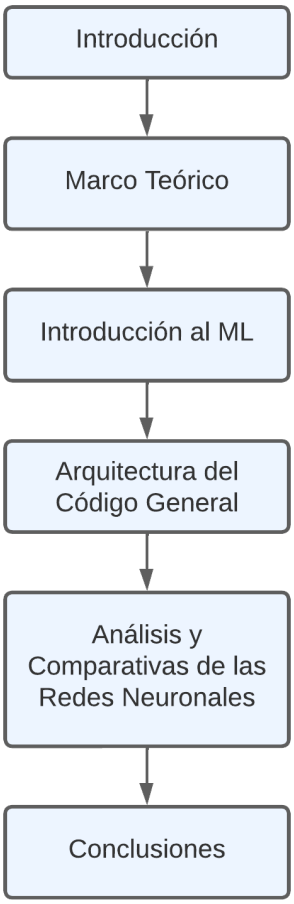

Figura 1.1: Diagrama de la estructura de la memoria. Fuente: Elaboración propia.

resultados obtenidos se evaluará si se han conseguido los objetivos establecidos o no. Además, se establecerán algunos posibles trabajos futuros a realizar para bien profundizar más en la temática del proyecto o para tomar otros enfoques distintos.

Por otro lado, en la Figura 1.2 se puede observar un diagrama de Gantt con el transcurso del tiempo empleado en el proyecto dividido por tareas y según la carga de trabajo empleada en cada una de ellas.

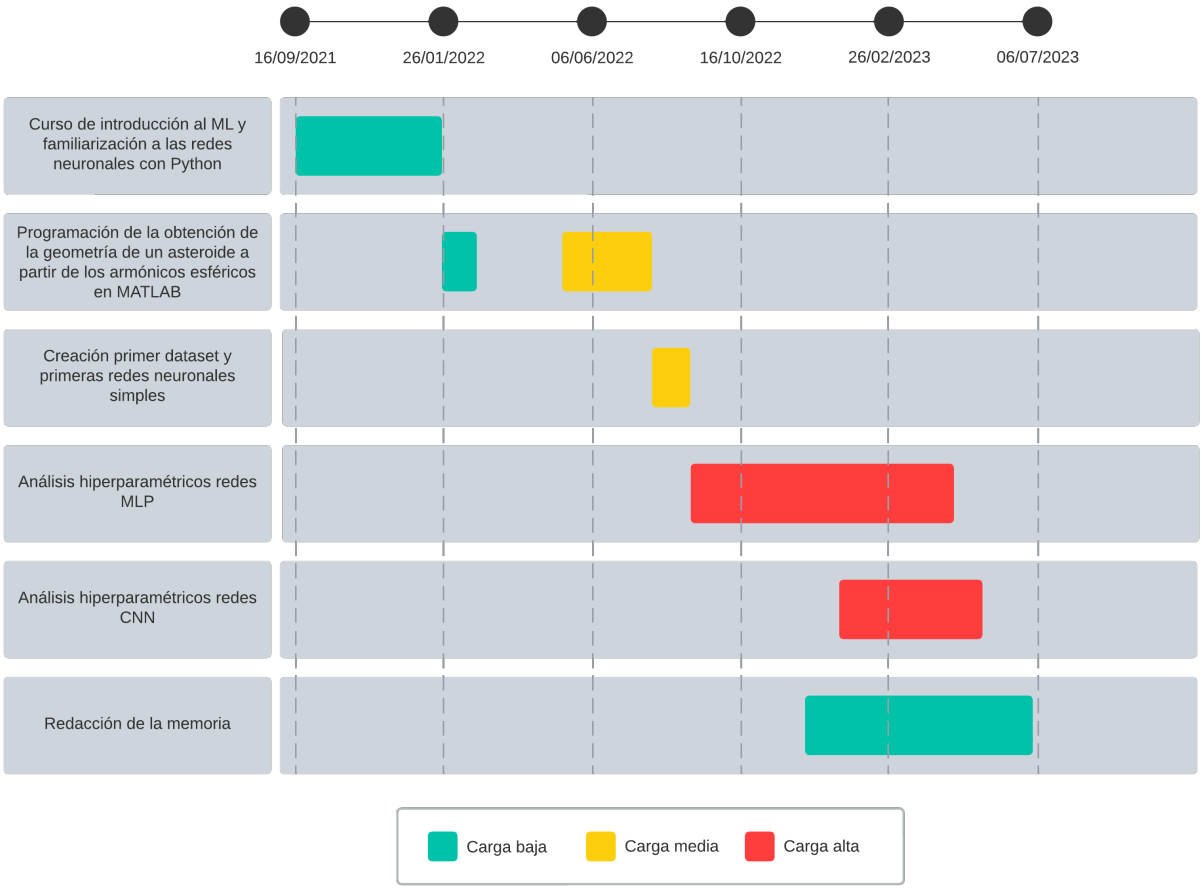

Figura 1.2: Diagrama de Gantt del transcurso del proyecto. Fuente: Elaboración propia.

Capítulo 2

Marco teórico

En este capítulo se van a introducir los conceptos teóricos necesarios para poder comprender el trabajo realizado. Primero, se realizará una introducción a la ley de gravitación universal de Newton para el problema de dos cuerpos.

Después, se generalizará para el caso de un movimiento no Kepleriano y cómo se llega a la función potencial gravitatoria no uniforme. A continuación, se verá cómo se puede expandir dicha función en armónicos esféricos.

Finalmente, a partir de lo desarrollado, se verá cómo se puede expandir cualquier geometría en armónicos esféricos.

2.1. Ley de Gravitación Universal de Newton

Una fuerza es la acción que ejerce un cuerpo físico sobre otro, ya sea mediante contacto directo o a distancia. La Ley de Gravitación Universal de Newton [6] explica la fuerza de atracción a distancia que existe entre dos cuerpos. Dicha fuerza, para dos cuerpos de masas M_1 y M_2 separados una distancia r se escribe mediante la ecuación

$$\mathbf{F} = -G\frac{M_1 M_2}{r^2}\overline{\mathbf{r}} \tag{2.1}$$

donde G es la constante de gravitación universal, que tiene un valor de $6{,}6742{\cdot}10^{-11} m^3 kg^{-1} s^{-2}$ [7] y $\overline{\mathbf{r}}$ es el vector unitario con la dirección de la fuerza. El signo negativo de la ecuación indica el sentido de la fuerza, el cual apunta al centro de gravedad del cuerpo primario con masa M_1, siendo este el origen de coordenadas.

Con esta ley ya es posible describir, por ejemplo, las principales características de la órbita de un satélite orbitando alrededor de la Tierra. Esto es debido a que la fuerza de atracción que ejerce la Tierra sobre el satélite es varios órdenes de magnitud mayor que el resto de fuerzas que actúan sobre él.

Para obtener la aceleración que sufriría dicho satélite basta con combinar la Ecuación 2.1 con la Segunda Ley de Newton y se obtiene

$$\ddot{\mathbf{r}} = -G\frac{M_1}{r^2}\bar{\mathbf{r}} \tag{2.2}$$

Donde el término GM_1 comúnmente se agrupa y se le denomina parámetro gravitacional μ, ya que es un invariante dentro del problema de dos cuerpos. En el caso de la Tierra, el parámetro gravitacional μ tiene un valor de $398600{,}4405 \pm 0{,}001 km^3 s^{-2}$ [8].

2.2. Potencial gravitatorio

En la sección anterior se explican los conceptos más básicos de las fuerzas que actúan en un sistema de dos cuerpos o *movimiento kepleriano*. Sin embargo, si se quieren hacer cálculos más precisos, es necesario tener en cuenta el resto de fuerzas implicadas y entonces se deja de tener un *problema kepleriano* y se pasa a uno *no kepleriano*.

Esto no es más que añadir un término adicional a la Ecuación 2.2 que recoja los efectos del resto de fuerzas, es decir,

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^2}\bar{\mathbf{r}} + \ddot{\mathbf{r}}_p \tag{2.3}$$

donde la componente $\ddot{\mathbf{r}}_p$ representa el sumatorio de las aceleraciones causadas por las perturbaciones. Dichas perturbaciones pueden tener multitud de orígenes, tales como el achatamiento de la Tierra o el cuerpo primario en cuestión, la existencia de otro cuerpo en el sistema o el efecto de la resistencia de la atmósfera entre otros [8].

De ahora en adelante, es conveniente usar una expresión equivalente a la Ecuación 2.2 en forma de gradiente de la función potencial $U = \frac{\mu}{\mathbf{r}}$, por lo que queda como

$$\ddot{\mathbf{r}} = \nabla U \tag{2.4}$$

cuya expresión cumple la ecuación de Laplace y que en coordenadas cartesianas es

$$\nabla^2 U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = 0 \tag{2.5}$$

La Ecuación 2.4 se puede generalizar para una distribución de masas cualquiera sumando elementos diferenciales de masa $dm = \rho(\mathbf{s})d^3\mathbf{s}$, donde $\rho(\mathbf{s})$ es la densidad en un punto $\mathbf{s}$, y el potencial gravitatorio queda entonces

$$U = G \int \frac{\rho(\mathbf{s})d^3\mathbf{s}}{|\mathbf{r} - \mathbf{s}|} \tag{2.6}$$

donde $|\mathbf{r} - \mathbf{s}|$ es la distancia del cuerpo secundario al elemento diferencial de masa dm del primario (Figura. 2.1).

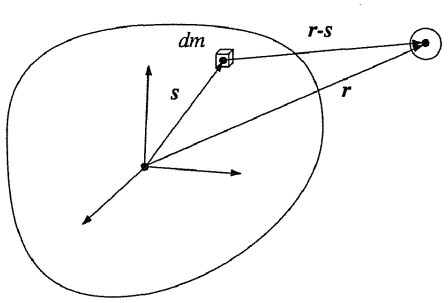

Figura 2.1: Distancia de un elemento diferencial de masa del cuerpo primario al cuerpo secundario. Fuente: [8].

Lo más conveniente a continuación es definir el cambio de coordenadas cartesianas a coordenadas esféricas, entonces

$$\begin{aligned} x &= r\cos\phi\cos\lambda \\ y &= r\cos\phi\sin\lambda \\ z &= r\sin\phi \end{aligned} \tag{2.7}$$

donde r representa la distancia radial desde el origen y los ángulos ϕ y λ representan la latitud y longitud (Figura. 2.2).

Finalmente, aplicando y desarrollando el cambio de coordenadas cartesianas a esféricas,

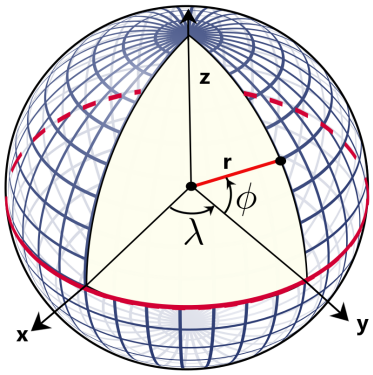

Figura 2.2: Representación de las coordenadas esféricas. Fuente: [9].

la ecuación de Laplace anteriormente mencionada 2.5, queda de la forma

$$r^2\nabla^2 U = \frac{\partial}{\partial r}\left(r^2\frac{\partial U}{\partial r}\right) + \frac{1}{\cos\phi}\frac{\partial}{\partial\phi}\left(\cos\phi\frac{\partial U}{\partial\phi}\right) + \frac{1}{\cos^2\phi}\frac{\partial^2 U}{\partial\lambda^2} = 0 \tag{2.8}$$

2.3. Expansión del potencial gravitatorio en armónicos esféricos

Tomando como punto de partida la Ecuación 2.8, los pasos a seguir para obtener la expansión del potencial gravitatorio en armónicos esféricos son los siguientes.

Primero, se presupone que la función potencial gravitatoria tiene la forma [10]

$$U = R(r)\Phi(\phi)\Lambda(\lambda) \tag{2.9}$$

Sustituyendo ahora la Ecuación 2.9 en la Ecuación 2.8, se tiene

$$\frac{1}{R}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) + \frac{1}{\Phi\cos\phi}\frac{d}{d\phi}\left(\cos\phi\frac{d\Phi}{d\phi}\right) + \frac{1}{\Lambda\cos^2\phi}\frac{d^2\Lambda}{d\lambda^2} = 0 \tag{2.10}$$

Se observa de la ecuación anterior que el primer término es el único que es dependiente de r, por lo que debe ser constante para que cumpla la ecuación. Se escribe entonces, por conveniencia, ese término como

$$\frac{1}{R}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right) = n\,(n+1) \tag{2.11}$$

donde $n \in \mathbb{N}$. Desarrollando la diferenciación queda

$$r^2 \frac{d^2R}{dr^2} + 2r\frac{dR}{dr} - nR\,(n+1) = 0 \tag{2.12}$$

La forma de la Ecuación 2.12 sugiere que R tiene una forma similar a r^k. Sustituyendo este valor de R en 2.12 y resolviendo para k, se tiene unos resultados de n y $-n-1$ o lo que es lo mismo

$$R(r) = Ar^n + Br^{-n-1} \tag{2.13}$$

donde A y B son constantes arbitrarias. Como condición de contorno se tiene que la función potencial se anula en el infinito, lo cual establece que $A = 0$.

A continuación, se sustituye la Ecuación 2.11 en 2.10 y se multiplica por $\cos^2\phi$ para aislar el término de Λ y se tiene

$$n\,(n+1)\cos^2\phi + \frac{\cos\phi}{\Phi}\frac{d}{d\phi}\left(\cos\phi\frac{\partial\Phi}{\partial\phi}\right) + \frac{1}{\Lambda}\frac{d^2\Lambda}{d\lambda^2} = 0 \tag{2.14}$$

De nuevo, se puede observar que el último término de la Ecuación 2.14 es el único dependiente de λ, por lo que debe ser también una constante. Si se hace que esa constante tenga un valor de $-m^2$ y se resuelve la ecuación diferencial, se tiene una solución para Λ del tipo

$$\Lambda(\lambda) = a\cos m\lambda + b\sin m\lambda \tag{2.15}$$

donde a y b son constantes arbitrarias y $m \in \mathbb{N}$. Sustituyendo ahora el término de Λ por $-m^2$ en la Ecuación 2.14 y multiplicando por el factor $\frac{\Phi}{\cos^2\phi}$, se tiene una ecuación que solamente es dependiente de ϕ

$$\frac{d}{d\phi}\left(\cos\phi\frac{d\Phi}{d\phi}\right) + \left[n\,(n+1) - \frac{m^2}{\cos^2\phi}\right]\Phi = 0 \tag{2.16}$$

Como solución a la anterior ecuación, para unos índices n y m positivos y enteros denominados *grado* y *orden*, se tiene la llamada *Función asociada de Legendre* $P_{nm}\,(\sin\phi)$ [11], es decir

$$\Phi\,(\phi) = P_{nm}\,(\sin\phi) \tag{2.17}$$

Esta función, de forma generalizada $P_{nm}\,(u)$ puede ser definida de varias formas [8], [10],

[12] entre las cuales se elige

$$P_{nm}(u) = \left(1-u^2\right)^{m/2} \frac{d^m}{du^m} P_n(u) \tag{2.18}$$

donde $P_n(u)$ es el polinomio de Legendre de grado n para $m = 0$, el cual se puede escribir como

$$P_n(u) = \frac{1}{2^n n!} \frac{d^n}{du^n} \left(u^2-1\right)^n \tag{2.19}$$

o de forma recursiva

$$P_n(u) = -\frac{n-1}{n} P_{n-2}(u) + \frac{2n-1}{n} u P_{n-1}(u) \tag{2.20}$$

Con todo lo desarrollado hasta ahora, la solución a la expansión del potencial gravitatorio en armónicos esféricos de la Ecuación 2.8, implementando la Ecuación 2.13 con $A = 0$ y $B = 1$ y las Ecuaciones 2.15 y 2.18, se tiene

$$U(r,\lambda,\phi) = \sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{1}{r^{n+1}} P_{nm}(\sin\phi) \left[a_{nm}\cos m\lambda + b_{nm}\sin m\lambda\right] \tag{2.21}$$

donde a_{nm} y b_{nm} son los coeficientes de Stokes de grado y orden n y m, respectivamente, con $n, m \in \mathbb{N}$, que describen al modelo específico. Además de estas soluciones reales, existen soluciones imaginarias pero no son relevantes para el caso de estudio de este proyecto.

A continuación, para añadir las dimensiones y escalar el potencial para el cuerpo en cuestión

$$U(r,\lambda,\phi) = \frac{\mu}{r} \sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{R^n}{r^n} P_{nm}(\sin\phi) \left[a_{nm}\cos m\lambda + b_{nm}\sin m\lambda\right] \tag{2.22}$$

donde R es el radio de referencia del cuerpo y μ su parámetro gravitacional.

Debido a la posibilidad de que los modelos con los que se trabajen tengan unos grados y órdenes altos, es recomendable utilizar una normalización tanto para los coeficientes de

Stokes como para los polinomios de Legendre. Esta normalización se calcula mediante [13]

$$\begin{Bmatrix} \overline{a}_{nm} \\ \overline{b}_{nm} \end{Bmatrix} = \sqrt{\frac{(n+m)!}{(2-\delta_{0m})\,(2n+1)\,(n-m)!}} \begin{Bmatrix} a_{nm} \\ b_{nm} \end{Bmatrix}$$
$$\overline{P}_{nm} = \sqrt{\frac{(2-\delta_{0m})\,(2n+1)\,(n-m)!}{(n+m)!}} P_{nm} \tag{2.23}$$

donde δ_{0m} es la delta de Kronecker y cuyo valor es

$$\delta_{0m} = \begin{cases} 1 & \text{si } m = 0 \\ 0 & \text{si } m \neq 0 \end{cases} \tag{2.24}$$

La Ecuación 2.22 con los coeficientes de Stokes y los polinomios de Legendre normalizados queda

$$U\,(r,\lambda,\phi) = \frac{\mu}{r} \sum_{n=0}^{\infty} \sum_{m=0}^{n} \frac{R^n}{r^n} \overline{P}_{nm}\,(\sin\phi)\left[\overline{a}_{nm}\cos m\lambda + \overline{b}_{nm}\sin m\lambda\right] \tag{2.25}$$

2.4. Obtención de una geometría a partir de armónicos esféricos

En la sección anterior se ha realizado el desarrollo del potencial gravitatorio y su expansión en harmónicos esféricos, sin embargo, cabe destacar que lo realmente necesario para el desarrollo del proyecto es la geometría de un cuerpo en vez de su función potencial gravitatoria.

Para ello, se debe definir qué es exactamente un **armónico esférico** y en qué se diferencia de una función armónica.

De forma genérica, una función armónica es cualquier solución de la ecuación de Laplace $\nabla^2 f = 0$. Ecuación que anteriormente se desarrolló en coordenadas esféricas en la Ecuación 2.8.

Por otro lado, un armónico esférico no es más que el producto de dos de las tres soluciones parciales, las soluciones ligadas a los ángulos, de la ecuación de Laplace en coordenadas esféricas. Es decir, si la solución de la ecuación de Laplace en coordenadas esféricas se suponía

de la forma

$$f = R(r)\Phi(\phi)\Lambda(\lambda) \tag{2.26}$$

Un armónico esférico es el producto de $\Phi(\phi)\Lambda(\lambda)$, el cual se renombra comúnmente como

$$Y(\lambda, \phi) = \Phi(\phi)\Lambda(\lambda) \tag{2.27}$$

Tal y como se vio en la sección anterior, la función que es solo dependiente de ϕ tiene como solución la función asociada de Legendre

$$\Phi(\phi) = P_{nm}(\sin\phi) \tag{2.28}$$

La función que es solo dependiente de λ y que anteriormente se desarrolló en la Ecuación 2.15, también se puede expresar de forma más compacta con números complejos [14] como

$$\Lambda(\lambda) = e^{im\lambda} \quad \text{con } m = 0, \pm1, \pm2, \ldots, \pm n \tag{2.29}$$

Así pues, el armónico esférico se reconstruye como

$$Y_{nm}(\lambda, \phi) = P_{nm}(\sin\phi)\, e^{im\lambda} \tag{2.30}$$

Y expandiendo la exponencial

$$Y_{nm}(\lambda, \phi) = P_{nm}(\sin\phi)\left[\cos m\lambda + i \sin m\lambda\right] \tag{2.31}$$

A continuación, para dotar al armónico esférico de escala y de rotación, este se multiplica por un factor complejo $(C_{nm} - i \cdot S_{nm})$, donde C_{nm} y S_{nm} son constantes que dependen del grado n y orden m. Multiplicando este factor por el lado derecho de la ecuación anterior se tiene

$$\begin{aligned} Y_{nm}(\lambda, \phi) &= P_{nm}(\sin\phi)\left[\cos m\lambda + i \sin m\lambda\right](C_{nm} - i \cdot S_{nm}) \\ &= P_{nm}(\sin\phi)\left[C_{nm}\cos m\lambda + iC_{nm}\sin m\lambda - iS_{nm}\cos m\lambda - i^2 S_{nm}\sin m\lambda\right] \\ &= P_{nm}(\sin\phi)\left[C_{nm}\cos m\lambda + iC_{nm}\sin m\lambda - iS_{nm}\cos m\lambda + S_{nm}\sin m\lambda\right] \end{aligned} \tag{2.32}$$

Los términos imaginarios carecen de sentido físico, por lo que se procede a eliminarlos.

Finalmente, el armónico esférico de grado n y orden m, queda definido como

$$Y_{nm}(\lambda,\phi) = P_{nm}\left(\sin\phi\right)\left[C_{nm}\cos m\lambda + S_{nm}\sin m\lambda\right] \tag{2.33}$$

Se puede demostrar que dos armónicos esféricos cualesquiera son completamente ortogonales mediante la integral de superficie sobre la esfera de radio unitario como

$$\int_{\lambda=0}^{2\pi}\int_{\phi=0}^{\pi} Y_{n_1m_1}(\lambda,\phi)Y_{n_2m_2}(\lambda,\phi)\sin\phi d\phi d\lambda = \delta_{n_1n_2}\delta_{m_1m_2} \tag{2.34}$$

Ahora, además de la completa ortogonalidad de los armónicos, lo cual indica que son linealmente independientes, se tiene la teoría de Sturm-Liouville aplicada a la ecuación de Laplace [15]. Esta teoría dice que para cualquier función $F\left(\lambda,\phi\right)$ representativa de una superficie lo suficientemente suave, es decir, con las suficientes propiedades de continuidad, evaluada sobre la superficie de la esfera de radio unitario, puede ser expandida en una serie doble uniformemente convergente de armónicos esféricos.

Se demuestra [16], [17], por tanto, que se puede tomar $F\left(\lambda,\phi\right)$ como la función que representa la componente radial de una geometría en función de los ángulos de longitud y latitud y esta puede ser expandida en armónicos esféricos como

$$r\left(\lambda,\phi\right) = \sum_{n=0}^{\infty}\sum_{m=0}^{n} P_{nm}\left(\sin\phi\right)\left[C_{nm}\cos m\lambda + S_{nm}\sin m\lambda\right] \tag{2.35}$$

Al igual que en la sección anterior, debido a la posibilidad de desbordamiento numérico de los polinomios de Legendre $P_{nm}(\sin\phi)$, estos se deben normalizar con la expresión de la Ecuación 2.23.

Además, se puede ver que si no se normalizan los coeficientes C_{nm} y S_{nm}, la geometría $r(\lambda,\phi)$ tendrá las mismas unidades que estos coeficientes, lo cual será de gran utilidad en el desarrollo del proyecto. Por lo tanto, la ecuación que se usará en las futuras secciones será

$$r\left(\lambda,\phi\right) = \sum_{n=0}^{\infty}\sum_{m=0}^{n} \overline{P}_{nm}\left(\sin\phi\right)\left[C_{nm}\cos m\lambda + S_{nm}\sin m\lambda\right] \tag{2.36}$$

Así pues, se tiene una función que mediante los coeficiente de Stokes de los armónicos esféricos C_{nm} y S_{nm} particulares de un objeto y los polinomios de Legendre normalizados

$\overline{P}_{nm}$, devuelve la distancia radial desde el origen hasta un punto establecido por su longitud λ y latitud ϕ y con lo cual queda definida la geometría de un cuerpo por medio de su expansión en armónicos esféricos.

Capítulo 3

Introducción al Machine Learning

En este capítulo se pretende hacer una introducción al *Machine Learning (ML)* y a las Redes Neuronales, ya que van a ser las principales herramientas que se van a usar en este proyecto para poder alcanzar los objetivos deseados.

Primero, se hará una explicación de qué es el *ML*, de cómo funciona a nivel interno y de sus principales características.

Después, se enumerarán los principales tipos de Redes Neuronales que existen, cómo se construyen y para qué tipo de aplicaciones se pueden usar.

Por último, después de haber explicado los tipos de redes que hay y sus ventajas, se comentará qué tipos de redes neuronales se han elegido para el desarrollo del proyecto.

3.1. Machine Learning

El *Machine Learning* es una rama de la *Inteligencia artificial (IA)* que, a través de análisis de una gran cantidad de datos, dota a los sistemas computacionales la capacidad de identificar patrones en los datos, **aprender** de ellos y hacer predicciones para nuevos datos con un bajo gasto computacional [18].

La *IA*, al principio de su invención, fue capaz de resolver problemas que son difíciles de resolver para los seres humanos pero que no lo son para los ordenadores. Estos problemas son los que son fácilmente descriptibles por una serie de normas matemáticas. Por ejemplo, problemas como el ajedrez, para un ordenador son muy fáciles de aprender. Esto se vio con el

Deep Blue de *IBM* que pudo derrotar al campeón del mundo Garry Kasparov en 1997 [19].

Lo realmente difícil, aunque parezca lo contrario, es enseñar a los ordenadores a resolver problemas que para los seres humanos son muy fáciles de resolver, problemas que los seres humanos resuelven automáticamente, de manera instintiva y sin pensar. Problemas tales como reconocer voces, caras o simplemente saber cuándo cierto objeto entra dentro de la definición de ese objeto con tan sólo mirarlo.

¿Cómo se le puede **enseñar** entonces a un ordenador? Un algoritmo de *machine learning* aprende cuando a través de la experiencia E sobre cierta tarea T y medida de rendimiento P, aumenta su rendimiento sobre la tarea T, medida con P, con la experiencia E [20].

3.1.1. Tareas, T

Se entiende por tarea el objetivo que se le quiere hacer aprender al ordenador. El aprendizaje es el proceso que se sigue para poder conseguir que el ordenador ejecute la tarea deseada. Por ejemplo, si se le quiere enseñar a un robot a que lance una pelota a una canasta y encestar, entonces encestar la pelota es la tarea. La idea es enseñarle al robot y que aprenda a cómo encestar la pelota, no decirle exactamente cuáles son los pasos a seguir para conseguir encestar.

Las tareas más típicas que se les suelen enseñar a un ordenador son:

- **Clasificación**: En este tipo de tareas lo que se le pide al ordenador es que clasifique cierta entrada (*input*) dentro de las k clases establecidas. Esto es, un algoritmo definido como $f : \mathbb{R}^n \longrightarrow \{1, \ldots, k\}$. Normalmente, f tiene como salida una distribución de probabilidades sobre las k clases y se determina que la clasificación final es la que mayor probabilidad tiene de todas.

  Un ejemplo de una tarea de clasificación es el reconocimiento de objetos, donde la entrada es una imagen, normalmente definida como una matriz de valores donde cada valor es la intensidad de brillo de los píxeles que forman dicha imagen, y la salida, es la clasificación de esa imagen entre las distintas clases de objetos posibles.

- **Regresión**: Este tipo de tareas son muy similares a las tareas de clasificación, la diferencia principalmente es la salida del algoritmo que, en este caso, es un valor numérico de predicción. En este caso, la función de salida del algoritmo debe ser $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. Un ejemplo de este tipo de tareas puede ser la predicción de una órbita de un satélite alrededor de un asteroide, es decir, dado el *input* de posición del satélite en el instante de tiempo t, se tiene como *output* la posición del satélite en el instante de tiempo $t + \Delta t$.

3.1.2. Rendimiento, *P*

Para poder discernir si el algoritmo entrenado está cumpliendo o no con el objetivo de la tarea T, se necesita poder evaluar de manera cuantitativa el rendimiento P. Hay multitud de formas distintas de calcular dicho rendimiento [21]. Por ejemplo, para tareas de clasificación, se suele utilizar la proporción de *outputs* correctos respecto del total, el llamado *accuracy*, normalmente acotado entre $[0, 1]$. Un *accuracy* de 0 indica que todos los *outputs* fueron clasificados de forma errónea y un *accuracy* de 1 indica una clasificación perfecta. De manera inversa, también se puede calcular la proporción de *ouputs* erróneos respecto del total o *error rate*, donde un valor de 0 en este caso indicaría la perfección.

Sin embargo, rara vez se tiene una tarea que devuelva de forma categórica una clasificación. Normalmente, se devuelve una distribución de probabilidades y entonces pierde el sentido el cálculo del rendimiento como se ha explicado. En este caso se debe de obtener un rendimiento que ofrezca un valor continuo para cada ejemplo. Esto también es así para las tareas de regresión.

Además, para que los rendimientos calculados sean más fieles al comportamiento que va a tener el algoritmo una vez esté desplegado, estos deben de estar calculados con una serie de datos, *data test*, que el algoritmo no haya visto nunca durante el aprendizaje para así poder cerciorar de que no exista ningún tipo de parcialidad o *bias*.

Así pues, se procede a listar las métricas de rendimiento más comúnmente usadas.

- **Error absoluto medio (*MAE*)**: Quizás la medida de rendimiento más simple y más usada. Su rango de valores es de $[0, \infty)$ y cuanto menor sea este valor, mejor será el

modelo. Matemáticamente se escribe

$$MAE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{3.1}$$

donde y_i es el valor real, $\hat{y}_i$ es el valor predicho y N es el número de datos totales del *data test*.

El problema del *MAE* es que no se pueden comparar dos modelos distintos simplemente con este rendimiento, ya que al ser el error medio, uno puede tener un error muy grande en un único valor y el otro puede tener muchos errores pequeños.

- **Error medio cuadrado (*MSE*)**: De los errores más usados en tareas de regresión lineal. Se calcula

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.2}$$

y gráficamente

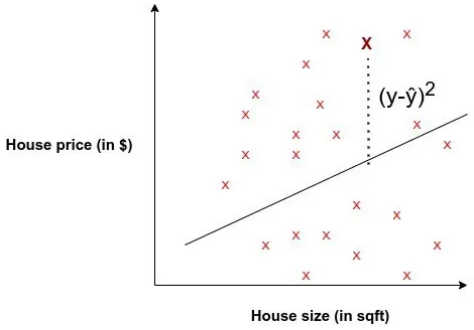

Figura 3.1: Representación gráfica del MSE. Fuente: [21].

Al igual que el *MAE*, también su rango es de $[0, \infty)$. Algunas de sus características son: es más sensible a valores atípicos, puesto que al estar elevado al cuadrado, penalizará mucho más estos datos; también penaliza incluso los pequeños errores, por el mismo motivo que la característica anterior; incluso un buen modelo puede tener un *MSE* alto, por lo que no basta sólo con esta métrica para evaluarlo.

- **Error medio cuadrático (*RMSE*)**: Al igual que los dos errores mencionados anteriormente, el rango del *MSE* es de $[0, \infty)$. De nuevo, cuanto menor sea el *RMSE*, mejor

será el modelo. Su expresión matemática es

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{3.3}$$

Este parámetro de rendimiento es de los que más información se puede extraer. Información como que si el *RMSE* es mucho mayor que el *MAE*, quiere decir que existe una gran varianza del error y que probablemente esto se deba a la existencia de valores atípicos. Además, el *RMSE* posee las mismas unidades que la variable de predicción y. Por otro lado, también es importante conocer que el *RMSE* cumple la siguiente relación con el *MAE*

$$MAE \leqslant RMSE \leqslant \sqrt{N} \cdot MAE \tag{3.4}$$

Por esta relación, si el *RMSE* es igual a $\sqrt{N} \cdot MAE$, entonces seguramente lo que esté pasando es que haya un gran valor atípico o que todo el error esté concentrado en una predicción.

- **Error R al cuadrado (R^2)**: También llamado *coeficiente de determinación*, guarda una estrecha relación con el *MSE*, pero el R^2 carece de escala, por lo que su rango es de $(-\infty, 1]$. Un valor negativo del R^2 indica que el modelo es peor que predecir la media. Un valor de 0 indicaría que la predicción es la misma que la de la media, mientras que un valor de 1 indicaría que el modelo no tiene error. El cálculo del R^2 se hace a partir de

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{3.5}$$

donde $\bar{y}$ es la media de los valores reales, calculada como

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.6}$$

En este caso, un valor del R^2 cercano a 1 es lo deseable, pero se debe de tener especial precaución porque no siempre esto indicará un mejor modelo. Un valor cercano a la unidad puede querer decir que el modelo posee un alto grado de **sobreajustado**. Esto quiere decir que el modelo está altamente condicionado a los datos con los que ha sido

entrenado y posiblemente generalice mal al ser desplegado en una situación real con mayor varianza de datos. Por todo ello, se deberán de hacer comprobaciones de que esto no ocurra y no se podrá aceptar sin más un modelo porque este tenga un R^2 más cercano a la unidad que el resto.

3.1.3. Experiencia, *E*

Los algoritmos de *ML* se pueden clasificar a grandes rasgos en **supervisados** y **no supervisados** [22] según el tipo de experiencia que tienen durante el proceso de aprendizaje.

De ahora en adelante, en este proyecto se hará referencia a los *dataset*, que son los conjuntos de datos, procesados o no, que se le proporciona a los algoritmos para que realicen un aprendizaje sobre ellos. También se hará referencia a la existencia de los *features* dentro de los *dataset*, los cuales no son más que los datos medidos cuantitativamente de cualquier evento u objeto que se quiera estudiar. Por ejemplo, los *features* de una imagen son los valores de los píxeles de esa imagen.

- **Algoritmos no supervisados**: En este tipo de algoritmos, el ordenador es alimentado con un *dataset* con multitud de *features* y este es capaz de aprender propiedades de su estructura. El objetivo es que el algoritmo aprenda la plenitud de la distribución de probabilidades que generó el *dataset* de entrada. Esto puede ser o bien mediante estimación de la densidad probabilística o implícitamente a través de tareas de síntesis o eliminación de ruido. También existen otros algoritmos no supervisados que se encargan, por ejemplo, de hacer agrupamientos o *clustering*, lo cual consiste en subdividir el *dataset* en subgrupos que compartan características similares.
- **Algoritmos supervisados**: Estos tipos de algoritmos reciben un *dataset* similar a los algoritmos no supervisados, pero en este caso, cada ejemplo viene asociado además con una etiqueta categórica. Esto permite al ordenador clasificar *features* y, además, facilita ciertas tareas como el *clustering*.

El término **supervisado** hace referencia a que las etiquetas deben de ser proporcionadas por quien esté a cargo del aprendizaje para que el ordenador sepa qué aprender. Sin embargo, en los algoritmos no supervisados, estas etiquetas no les son proporcionadas al ordenador y es este quien por cuenta propia debe ser capaz de aprender la estructura del *dataset*.

3.2. Estadística en *ML*

Se han estado haciendo menciones a distribuciones de probabilidades, densidades probabilísticas y demás conceptos del campo de estudio de la estadística, a continuación se procederá a definir qué son y por qué son tan relevantes estos términos para el *ML*.

Todo algoritmo de *ML* aprende a partir de los *dataset*, por lo que cómo sean dichos *dataset* y cómo estén organizados jugará un papel muy importante a la hora del aprendizaje del ordenador.

En concreto, se procede a introducir la **distribución normal** o **distribución de Gauss** y a explicar por qué es tan importante para el *ML*.

Cualquier muestra aleatoria de *features* independientes entre sí tiene aproximadamente una distribución gaussiana. Esta es una distribución que puede ser definida con tan sólo dos parámetros, la **media** μ y la **varianza** σ^2. El parámetro de la media se define, por ejemplo, para una variable aleatoria $Y \in \mathbb{R}$ de la cual se conocen n observaciones $y_1, \ldots, y_n$, como [23]

$$\mu = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{3.7}$$

Para el cálculo de la varianza antes se debe definir qué es la desviación. La desviación se define [24] como la diferencia entre una observación y_i y la media μ, es decir, $y_i - \mu$. Es fácil ver que hay tantas desviaciones como observaciones de la variable aleatoria.

Volviendo al cálculo de la varianza, esta se puede definir como el promedio de los cuadrados de las desviaciones. Matemáticamente se define mediante

$$\sigma^2 = \frac{\sum_{i=1}^{n} (y_i - \mu)^2}{n} \tag{3.8}$$

A partir de la ecuación anterior, también es de interés introducir el concepto de **desviación estándar** σ, el cual es un indicativo de la dispersión de los datos y será de utilidad más adelante. Este simplemente se define como la raíz cuadrada de la varianza

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \mu)^2}{n}} \tag{3.9}$$

Una vez se tienen definidos los parámetros de la media y de la varianza se puede definir la función de densidad de probabilidad con respecto a estas variables. Esta función define cuál será la probabilidad de que cierto evento x contenido dentro de los *features* X, es decir, $x \in X$, ocurra. La expresión que define esta función es

$$f\left(x|\mu, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.10}$$

Esta función, gráficamente toma la forma de la Figura 3.2 y es comúnmente denominada como *campana de Gauss*.

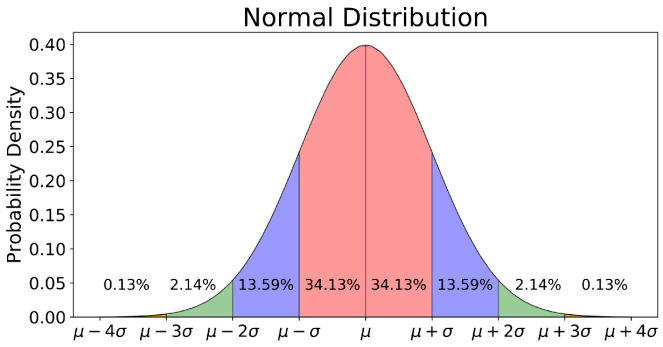

Figura 3.2: Distribución normal o distribución gaussiana. Fuente: [25].

Se puede observar en la figura anterior que si un evento x es igual a la media μ, este tendrá la probabilidad más alta de ocurrir. A medida que x se va alejando de la media, el evento va perdiendo probabilidad, es decir, el evento se va convirtiendo en más "raro".

Una de las características de la desviación estándar es que cuanto menor sea esta, mayor será por lo tanto el máximo de la función y más agrupada estará la función alrededor de la media. Por otro lado, la media lo que hará será desplazar toda la función sobre el eje de abscisas.

Una vez definida la distribución gaussiana, esta se puede demostrar que tiene una estrecha relación con cualquier tarea de *ML*.

El ejemplo más sencillo de explicar es la tarea de regresión. Se toma como ejemplo de partida una predicción por regresión lineal básica. Ahora, se puede considerar cada uno de los eventos reales como una distribución gaussiana con media en la línea de regresión tal y como se muestra en la Figura 3.3. De esta manera, si el evento real estuviera justo en la línea de regresión, este estaría en la media de la distribución normal y por lo tanto tendría la mayor probabilidad. Al contrario, cuanto más alejado esté el evento de la línea de regresión, más alejado estará de la media y su probabilidad será más baja.

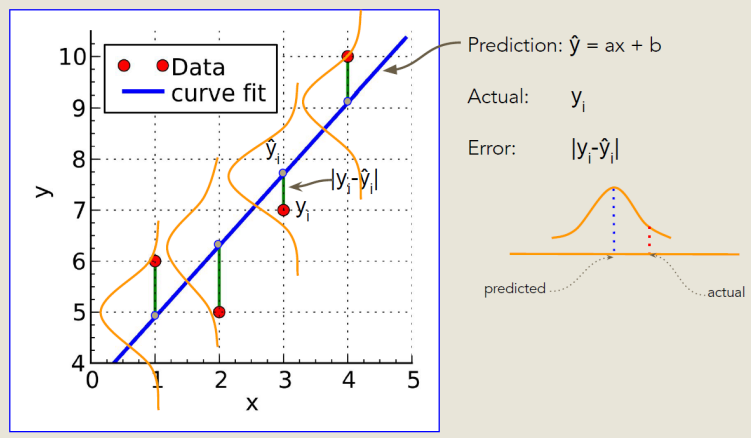

Figura 3.3: Ejemplo de regresión lineal. Fuente: [26].

Por lo tanto, con esta información se puede deducir que el objetivo de un algoritmo de una tarea de regresión será maximizar la probabilidad combinada de todos los eventos. Es decir, matemáticamente tiene que maximizar la función de probabilidad de los n eventos x_n de distribución gaussiana con la ecuación siguiente

$$f\left(x_1,\ldots,x_n|\mu,\sigma^2\right)=\prod_{i=1}^{n}f\left(x_i|\mu,\sigma^2\right)=\left(\frac{1}{2\pi\sigma^2}\right)^{n/2}\cdot\exp\left(-\frac{\sum_{i=1}^{n}\left(x_i-\mu\right)^2}{2\sigma^2}\right) \tag{3.11}$$

3.3. Redes Neuronales

Con todo lo desarrollado en las secciones anteriores, se converge en el concepto de *red neuronal*. Una red neuronal es un modelo de aprendizaje automático cuya estructura está inspirada en las redes neuronales del cerebro humano. Este tipo de modelos son utilizados

en el **deep learning**, algoritmos estructurados y jerarquizados capaces de modificar sus parámetros automáticamente y, en definitiva, *aprender* una tarea.

Para hablar de cómo se estructuran las redes neuronales primero se deben introducir los conceptos de **nodo** y **capa**. Manteniendo el símil con el cerebro humano, los nodos están estructurados en capas y, como las neuronas del cerebro, estos son capaces de transmitir información a otros nodos que estén en capas contiguas. Las capas están organizadas secuencialmente y siempre debe de haber un mínimo de tres capas: una capa de entrada o *input layer*, una o varias capas intermedias denominadas *hidden layers* y una capa de salida o *output layer*. Gráficamente, un ejemplo de una red neuronal simple está representada en la Figura 3.4.

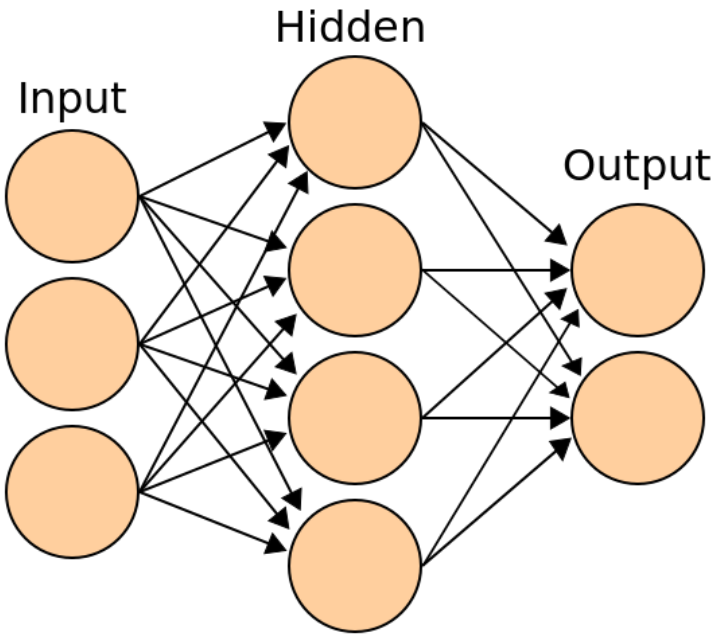

Figura 3.4: Ejemplo simple de red neuronal con tres entradas y dos salidas. Fuente: [27].

Según cómo estén estructurados estos nodos y capas, se tratará de un tipo de red neuronal u otra. En la Sección 3.3.1 se detallarán las más comunes.

Ahora bien, ¿cómo se pasa información de un nodo a otro? Para ello, se comienza explicando la capa de entrada. En esta capa, cada uno de los nodos que la componen se corresponderá con una entrada, por ejemplo en la figura 3.4 podrían ser tres mediciones de sensores de un

motor. Después, en este mismo ejemplo, la *hidden layer* que existe se trata de una capa totalmente conectada o capa densa o *fully connected layer*. En este tipo de capas, sus nodos, como su nombre indica, están todos conectados con todos los nodos de las capas contiguas. Finalmente, en el ejemplo de la Figura 3.4, se tiene la capa de salida con dos nodos, por lo tanto existirán dos salidas en esta red neuronal, que siguiendo el ejemplo anterior, podría ser el desgaste de dos piezas del motor.

En este tipo de redes neuronales, la información se pasa a través de los nodos con las conexiones que existen entre ellos. Suponiendo que los nodos de entrada contienen un valor cualquiera denominado $i_1, \ldots, i_n$ con n el número de nodos de entrada, estos se conectan con todos los nodos de la *hidden layer* contigua. Cada una de estas conexiones tiene asociada cierto factor de peso o *weight factor* $w_{11}, w_{12}, \ldots, w_{n1}, \ldots, w_{nm}$ con m el número de nodos de la *hidden layer*. Si se supone ahora que el valor de los nodos de la *hidden layer* se les denomina por $h_1, \ldots, h_m$, estos se calculan como

$$h_{1,\ldots,m} = \begin{bmatrix} i_1, \ldots, i_n \end{bmatrix} \cdot \begin{bmatrix} w_{11} \\ \vdots \\ w_{nm} \end{bmatrix} \tag{3.12}$$

El resto de valores de los nodos de las capas posteriores se calcularán de forma análoga. Estos factores de peso son los que mediante la experiencia, el algoritmo irá ajustando para producir las salidas deseadas y, por consiguiente, realizar el aprendizaje de la tarea en cuestión.

Profundizando un poco más en este aspecto, las redes neuronales disponen de unas funciones no lineales que se encargan de activar o desactivar ciertos nodos, de igual manera que hace el cerebro humano con las neuronas. Estas funciones no lineales se denominan **funciones de activación**. Así pues, la Ecuación 3.12 está incompleta y es necesario añadirle la función de activación genérica $g(z)$ de la forma

$$h_{1,\ldots,m} = g\left(\begin{bmatrix} i_1, \ldots, i_n \end{bmatrix} \cdot \begin{bmatrix} w_{11} \\ \vdots \\ w_{nm} \end{bmatrix} \right) \tag{3.13}$$

Existen múltiples funciones de activación distintas. A continuación se enumeran las más

usadas:

- **Sigmoid**: Convierte una función lineal en probabilidades acotadas entre 0 y 1

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.14}$$

- **Tanh**: Es la función trigonométrica tangente hiperbólica. Se calcula como

$$g(z) = \tanh z = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.15}$$

  Guarda una estrecha relación con la función *sigmoid*, tal que

$$\tanh z = 2g_{sigmoid}(2z) - 1 \tag{3.16}$$

- **ReLU**: Es la abreviación de *rectified linear unit*. Esta función de activación puede ser más eficientemente guardada y calculada que la función *sigmoid* y se calcula [28] como

$$g(z) = \begin{cases} 0 & \text{si } z < 0 \\ z & \text{si } z \geqslant 0 \end{cases} \tag{3.17}$$

3.3.1. Tipos de Redes Neuronales

Tal y como se introdujo en la sección anterior, según cómo se organicen las capas de las redes neuronales se tendrá un tipo de red distinta. A grandes rasgos, todas las redes neuronales se pueden dividir en tres grandes grupos: *MLP*, *RNN* y *CNN*.

- **MLP (Multi-Layer Perceptron)**: Se trata de una red con múltiples capas de perceptrones. Un perceptrón es la unidad más básica de red neuronal [29]. Se forma de solamente dos capas: la capa de entrada y la de salida, tal y como se muestra en la Figura 3.5 . Así pues, el perceptrón simplemente coge los datos de entrada, los multiplica por cierto factor, les aplica la función de activación y eso es la salida. Debido a su simpleza, el perceptrón sólo se puede usar para clasificaciones binarias.

  Por otro lado, las *MLP* son perceptrones que se forman por tres o más capas y múltiples nodos. Las estructuras de las redes *MLP* tienen la forma de la Figura 3.6, en la cual se

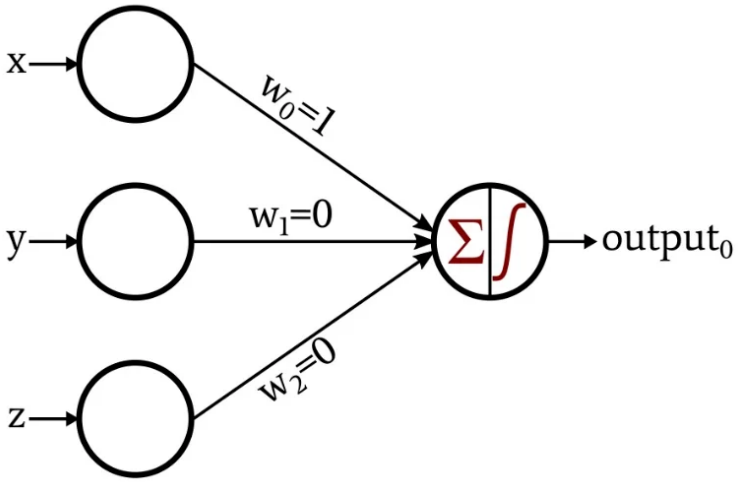

Figura 3.5: Ejemplo de una red neuronal perceptrón. Fuente: [30].

puede ver que todas las capas son densas. Se puede decir que es una red bidireccional, las entradas se envían hacia delante y las actualizaciones de los pesos se envían hacia atrás.

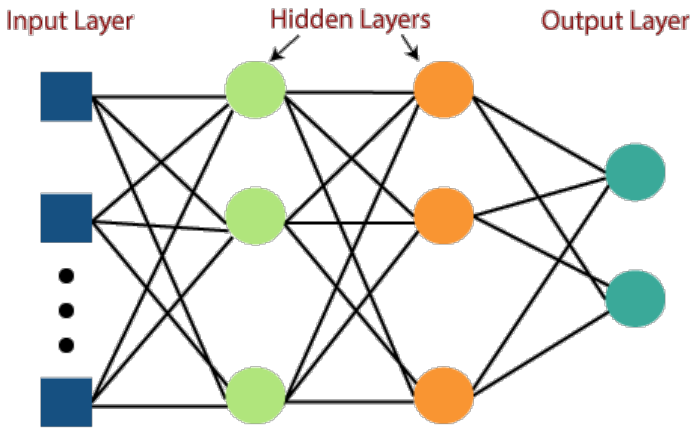

Figura 3.6: Ejemplo de una red neuronal *MLP*. Fuente: [31].

Este tipo de redes neuronales se pueden usar para multitud de tareas [32] tales como reconocimiento de voz, traducción automática, reconocimiento de imágenes, etc. Además, gracias a la posibilidad de crear redes muy complejas con un gran número de capas y nodos y, por ende, pesos, estas redes son muy útiles a la hora de crear modelos matemáticos mediante regresión para aproximar funciones. También son comúnmente utilizadas para tareas de clasificación, ya que estas son un caso particular de las tareas

de regresión en las que la salida es categórica.

- **RNN (Recurrent Neural Network)**: Las *RNN* poseen una estructura muy similar a las *MLP* con la única y gran diferencia de que las salidas de los nodos de las *hidden layers*, además de ser transferidas a las siguientes capas, son retroalimentadas en esos mismos nodos, gráficamente se puede ver en la Figura 3.7. Esto se hace para que las nodos actúen, en parte, como células de memoria y sean capaces de recordar información del instante de tiempo anterior. Por ello, las *RNN* son especialmente útiles para tareas dependientes del tiempo, semiperiódicas o no, predicción de texto o procesamiento de audio.

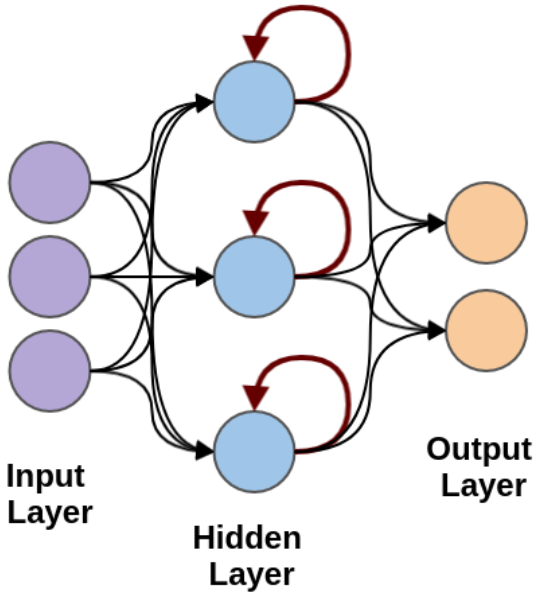

Figura 3.7: Ejemplo de una *RNN* de una sola *hidden layer*. Fuente: [33].

En especial, del tipo de redes *RNN*, destacan las *LSTM* o *Long Short-Term Memory*. Este tipo de redes son capaces de almacenar información durante periodos de tiempo más largos que el resto [34]. Debido a esta capacidad, son muy útiles en tareas en las que los datos tienen dependencias a largo plazo. A grandes rasgos, las redes *LSTM* tienen tres puertas: puerta de entrada, de salida y de "olvidar". La puerta de entrada se encarga de controlar qué datos son los que se almacenan en memoria, la puerta de salida se encarga de decidir qué información se transfiere a la siguiente capa y la puerta

de “olvidar” controla qué datos olvidar y cuándo hacerlo.

- **CNN (Convolutional Neural Network)**: Las *CNN* son redes neuronales que suelen tener como *input* imágenes y cuyas principales tareas son la clasificación y la regresión.

  Las *CNN* poseen una estructura más compleja que el resto de redes anteriormente mencionadas. A grandes rasgos, la estructura de una *CNN* se puede separar en dos partes bien diferenciadas (Figura 3.8): la parte convolucional y la parte de clasificación/regresión.

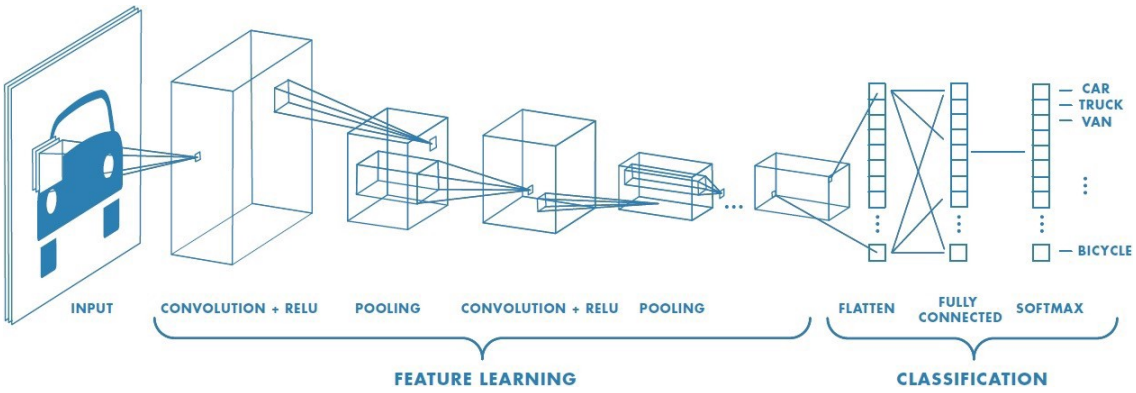

Figura 3.8: Ejemplo de una *CNN* de clasificación de imágenes. Fuente: [35].

La parte convolucional es la encargada de extraer las características propias de las imágenes y en el proceso, normalmente, reducir el tamaño de dichas imágenes para que la parte de clasificación/regresión no trabaje con un *dataset* demasiado grande. Esto lo consigue concatenando capas de filtros, funciones de activación y de *pooling* o submuestreo.

Una imagen no deja de ser una matriz tridimensional de Altura x Anchura x Canales píxeles. El número de canales son la cantidad de colores primarios de los que la imagen se compone. La mayoría de las imágenes son *RGB* lo que quiere decir que se componen de los 3 colores primarios rojo (*red*), verde (*green*) y azul (*blue*). Existen otros tipos de canales en los que una imagen puede existir: escala de grises, CMYK, HSV, etc. Un ejemplo de cómo es la estructura de una imagen *RGB* se puede observar en la Figura 3.9.

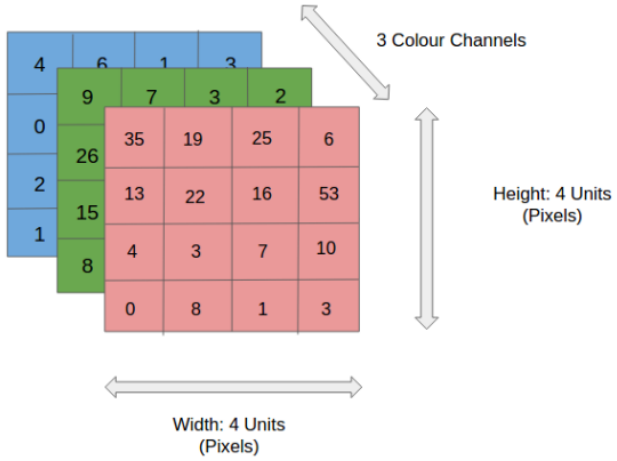

Figura 3.9: Estructura de una imagen 4x4x3 RGB. Fuente: [35].

- **Filtro:**

  Una vez definido el *input*, el primer paso para la convolución es el **filtro** (también denominado *Kernel*). El filtro es una matriz de pesos que se van actualizando a lo largo del entrenamiento y que tiene la misma profundidad que el *input*. El tamaño del filtro suele ser de 3x3x3, aunque este puede variar y esta variación determinará el tamaño de la salida del filtro. El filtro es entonces aplicado [35] a cada uno de los canales de la imagen de entrada comenzando desde la parte superior izquierda de la matriz, esto es, se calcula el producto escalar entre la imagen y el filtro y el resultado más un cierto *bias* será una de las salidas (Figura 3.10). Acto seguido, el filtro se desplaza cierta cantidad de **pasos** hacia la derecha y se vuelve a aplicar el producto escalar. Una vez se completa toda la anchura de la imagen, el filtro vuelve a la parte izquierda de la matriz y se desplaza hacia abajo la misma cantidad de pasos establecida. Este proceso se repite de forma análoga hasta haber cubierto la totalidad de la imagen. El resultado de esta operación se denomina mapa de características o *feature map*.

  El objetivo de la operación del filtro es extraer las principales características de las imágenes como los bordes. Las *CNN* no tienen por qué estar limitadas a una sola capa convolucional, de hecho rara vez tienen sólo una capa, puesto que la primera capa es la encargada de obtener las características de bajo nivel como los colores, orientación, etc.

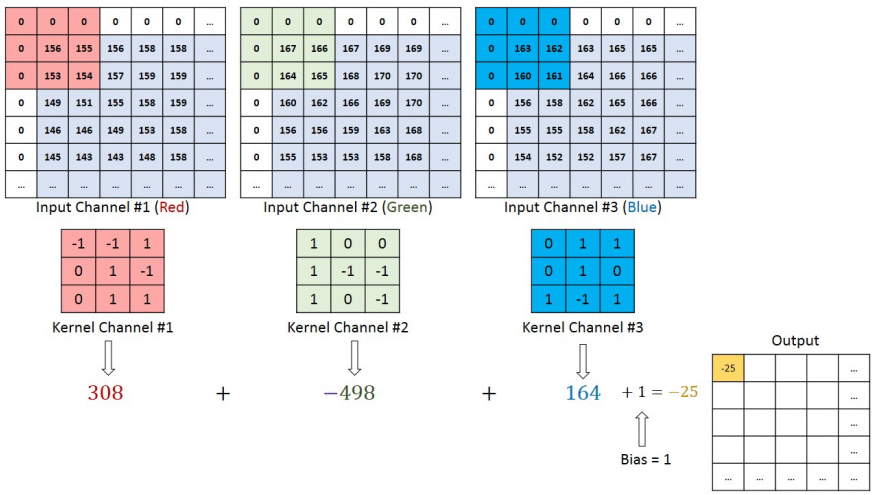

Figura 3.10: Ejemplo de la aplicación de un filtro. Fuente: [35].

Existen tres tipos de resultados del filtro: uno en el que el *feature map* es de un tamaño menor al del *input*, otro en el que el resultado aumenta y otro en el que ha mantenido su tamaño. Esto depende del tipo de relleno o *padding* utilizado [36]: *valid padding*, *full padding* y *same padding*, respectivamente.

- En el caso del *valid padding*, también conocido como *no padding*, la última convolución no se realiza si las dimensiones de la imagen y del filtro no coinciden.
- El *full padding* incrementa el tamaño de la salida mediante la adición de filas y columnas de ceros en el borde de la imagen.
- Finalmente, en el *same padding* se asegura que la salida del filtro tenga el mismo tamaño que la entrada, esto lo consigue añadiendo o no filas y/o columnas de ceros en el borde de la imagen.

- ***Pooling:*** En la capa de *pooling*, o capa de reducción de resolución, de nuevo se produce una disminución del número de parámetros del *input*. Esto es para reducir la potencia computacional necesaria para procesar los datos.

  De forma similar a la convolución explicada previamente, la operación *pooling* recorre un filtro por toda la superficie del *input* pero en vez de realizar un producto escalar mediante unos pesos, esta aplica una función de agregación a los valores del *input* que encierra el tamaño del *pooling* elegido. Existen dos tipos principales

de *pooling* (Figura 3.11):

- ***Max pooling:*** Conforme se mueve el filtro a través del *input* este selecciona el valor más alto de entre los que encierra el filtro. Este tipo de *pooling* además actúa como un supresor de ruido ya que descarta los valores menos relevantes. Por ello, este tipo de *pooling* suele ser el más utilizado.
- ***Average pooling:*** En este tipo de *pooling*, en vez de obtener el máximo del submuestreo, se obtiene la media de los valores de este.

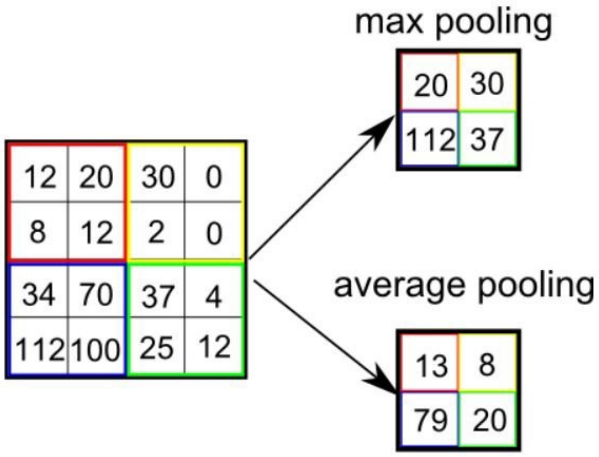

Figura 3.11: Ejemplos de *max pooling* y *average pooling*. Fuente: [35].

Así pues, la parte convolucional de una *CNN* no es más que la concatenación de las capas de filtros y *pooling* que se deseen. A mayor número de agrupaciones de capas se dice que la red es más profunda o *deep* y con ello la red es capaz de aprender características de mayor nivel de las imágenes. Sin embargo, se debe de tener especial cuidado porque también se aumenta el riesgo de sobreajustado de la red.

Por otro lado, la parte de regresión/clasificación de una *CNN* es igual al resto de redes explicadas anteriormente. Por ello, la principal diferencia de las redes *CNN* con el resto es la forma en la que se *preprocesan* las entradas para poder ser luego transferidas a cualquiera de los otros tipos de redes existentes.

3.3.2. Elección de Redes Neuronales

Tal y como se ha explicado en la sección anterior, existen multitud de tipos de redes neuronales distintas disponibles para la realización de una tarea. Además de las mencionadas anteriormente, también existen combinaciones, alteraciones y profundizaciones de todos los tipos de redes explicadas. Debido a ello y a las limitaciones de este proyecto, se deben de elegir un cierto número de tipos de redes a estudiar.

Para los objetivos propuestos del proyecto, se han descartado las redes tipo *RNN*. Estas redes *RNN* se descartan debido a que en el problema del proyecto no existe ningún tipo de dependencia temporal entre los datos, que es donde las redes *RNN* destacan, por lo que estas no serían la mejor opción.

Por otro lado, las redes *MLP* ofrecen una gran flexibilidad a la hora de establecer sus estructuras y, además, poseen una buena relación entre la sencillez de éstas y lo efectivas que son al desempeñar tareas de regresión.

En cuanto a las redes *CNN*, estas son ciertamente interesantes para el problema planteado del proyecto. Esto es así ya que, como se verá más adelante en el Capítulo 4, el *input* de las redes será la geometría de los asteroides, la cual se puede representar de forma matricial y esto permite hacer uso pleno del potencial de este tipo de redes, ya que en esencia no habría distinción con una 'imagen'.

Por todo lo anterior, los tipos de **redes elegidas** para la realización del proyecto son las redes ***MLP*** y ***CNN***.

Capítulo 4

Arquitectura general

En este capítulo se va a introducir y explicar de forma detallada la estructura del código desarrollado capaz de entrenar a una red neuronal con la tarea de conseguir los armónicos esféricos particulares de un asteroide a partir de la geometría de este.

Para comenzar, se parte como referencia del diagrama de flujo general representado en la Figura 4.1. Este diagrama muestra a *grosso modo* la estructura del código que se encarga de entrenar la red neuronal para la tarea deseada y de calcular los rendimientos de dicha red.

Como se puede observar en la Figura 4.1, el código se puede dividir en 4 secciones en las cuales se profundizarán más adelante.

Primero, a grandes rasgos el código comienza con la creación de N asteroides. Después, estos asteroides se dividen en *Train Asteroids* y *Test Asteroids*, es decir, en los asteroides que se van a usar para entrenar la red y en los que se van a utilizar para calcular las métricas que van a permitir conocer el rendimiento de la red, respectivamente. A continuación, los *Train Asteroids* se vuelven a dividir en *Input* y *Validation Asteroids*, ambos requeridos para el entrenamiento y que se explicarán en detalle más adelante. Después, una vez la red está entrenada, se le introduce como *input* los *Test Asteroids*, es decir, asteroides que no han pasado por la red durante el entrenamiento y que, por lo tanto, esta no conoce. Finalmente, los *Predicted Asteroids*, asteroides predichos por la red en el paso anterior, se comparan junto con los *Test Asteroids* mediante las métricas establecidas para calcular el rendimiento de la red.

Todo el código desarrollado a lo largo de este proyecto se ha realizado en **MATLAB**

R2022b versión académica. Además, una versión básica del código representado por la Figura 4.1, tanto para las redes MLP como CNN, se pueden consultar a través de mi repositorio público personal de GitHub[1].

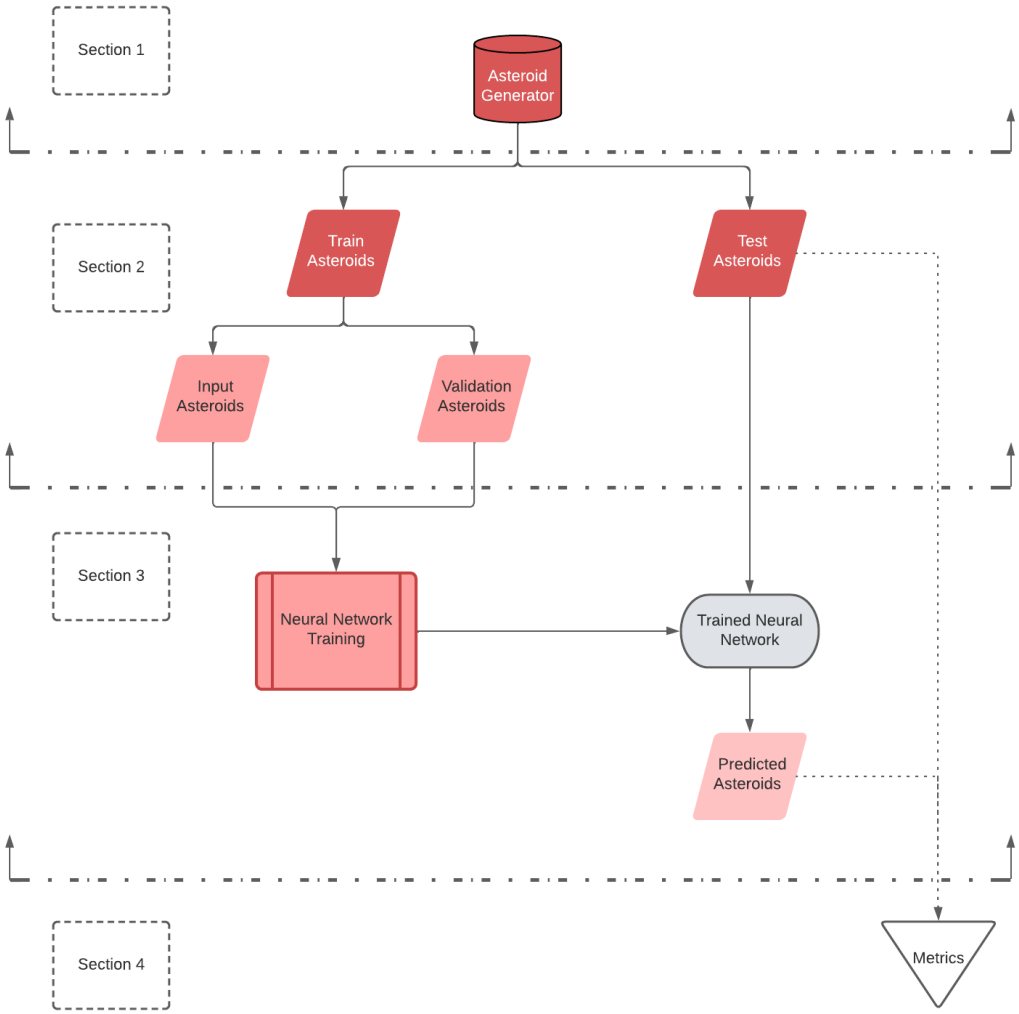

Figura 4.1: Diagrama del flujo general del código. Fuente: Elaboración propia.

4.1. Section 1

La primera sección se dedica exclusivamente a la generación de todos los asteroides que se van a utilizar en el código. Antes de explicar cómo se generan estos, se debe definir qué es lo que se considera un asteroide.

[1] `https://github.com/jjgonzreig/Spherical-harmonics-ML`

Un asteroide, de ahora en adelante y para el resto del proyecto, se le considera una variable matricial de 3 columnas y N filas. Las 3 columnas representan, respectivamente, ambos coeficientes de armónicos esféricos C_{nm}, S_{nm} y la componente radial $r(\lambda, \phi)$ en coordenadas esféricas (λ, ϕ, r) de la geometría del asteroide en cuestión, la cual se calcula mediante la Ecuación 2.36.

El número de N filas dependerá del máximo de los tamaños entre los armónicos esféricos y la componente radial $r(\lambda, \phi)$:

- **Armónicos esféricos:** El número de filas de los armónicos esféricos dependerá del armónico de mayor grado n y orden m que se quiera generar. Por otro lado, tal y como se puede ver en la Ecuación 2.36, los armónicos C_{nm}, S_{nm} van multiplicados por los polinomios de Legendre $P_{nm}(\sin\phi)$, definidos en la Ecuación 2.18, por lo que los armónicos sólo tendrán impacto cuando dichos polinomios de Legendre sean no nulos. Esto ocurre [12] para los grados $n \in \mathbb{N}$ y ordenes $m = 0, 1, \ldots, n$.

  Esta restricción es relevante para el conocimiento de las filas puesto que estableciendo el grado n y orden m del último armónico a generar, se puede obtener el número de elementos no nulos mediante el polinomio:

  $$N_{C,S} = \frac{1}{2}(n_{max} + 2)(n_{max} + 1) \tag{4.1}$$

  siendo n_{max} el grado del armónico de mayor grado y orden a generar, esto es, para un $n_{max} = 3$ se obtendrían 10 armónicos y los últimos coeficientes serían los C_{33}, S_{33}.

- **Componente radial $r(\lambda, \phi)$:** El número de filas de la componente radial $r(\lambda, \phi)$ dependerá del número de pasos escogidos para la longitud $\lambda \in [0, 2\pi]$ y latitud $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. Esto generará una malla de puntos alrededor de una esfera para los cuales mediante la Ecuación 2.36 se obtendrá la componente radial $r(\lambda, \phi)$ para cada uno de ellos. Por lo tanto, se puede establecer el número de filas mediante la sencilla expresión:

  $$N_r = grid(\lambda) \cdot grid(\phi) \tag{4.2}$$

  donde $grid(\lambda)$ y $grid(\phi)$ son los pasos de la longitud y la latitud, respectivamente. Por ejemplo, si se establece un paso de 10 elementos tanto para la longitud como para la

latitud, el número de filas de la componente radial será de 100.

Así pues, el número de N filas de un asteroide será el máximo entre $N_{C,S}$ y N_r. Si por ejemplo, el máximo de filas se corresponde con $N_{C,S}$, los elementos que correspondan con la diferencia entre N_r y $N_{C,S}$ de la columna de la componente radial se rellenarán con ceros (posteriormente en la Section 2 del código se eliminarán para no introducir error), tal y como se representa en la Tabla 4.1. De forma similar sucederá si el máximo de filas se corresponde con N_r (ver Tabla 4.2).

n	**m**	$C_{nm}[m]$	$S_{nm}[m]$	**r[m]**
0	0	7316.022	0	9582.554
1	0	-41.6948	0	8017.326
1	1	-183.991	72.22445	7088.885
2	0	286.5868	0	8029.27
2	1	-266.086	4.70745	8282.827
2	2	192.995	-252.541	7684.345
⋮	⋮	⋮	⋮	⋮
13	7	1.292318	-0.15447	7376.211
13	8	0.568926	-1.18949	7192.404
13	9	0.225755	0.744551	0
13	10	0.097036	0.323486	0
13	11	0.884846	-0.42193	0
13	12	0.38716	-0.36434	0
13	13	0.608808	0.758862	0

Tabla 4.1: Ejemplo de variable asteroide en el caso $N_{C,S} > N_r$

Se han introducido las columnas de los grados n y órdenes m en las Tablas 4.1 y 4.2 como ayuda visual, pero estas columnas no existen dentro de la variable asteroide tal y como se explicó al comienzo de la sección.

4.1.1. Generación aleatoria de armónicos esféricos

Una vez se han establecido las dimensiones de la variable del asteroide, se comienza el proceso de generación de todos los asteroides que se van a usar en el resto del código.

Para ello, se ha tomado como ejemplo de partida el famoso asteroide Eros 433, representado en la Figura 4.2. Se ha escogido el asteroide Eros 433 debido a que es un asteroide

n	**m**	$C_{nm}[m]$	$S_{nm}[m]$	**r[m]**
0	0	7316.022	0	11947.6
1	0	315.1271	0	8809.227
1	1	-76.2842	134.4207	6872.006
2	0	-94.475	0	6938.374
2	1	75.82977	-488.043	5826.044
2	2	116.2496	464.6119	7506.409
⋮	⋮	⋮	⋮	⋮
14	9	0	0	5826.044
14	10	0	0	7506.409
14	11	0	0	10391.55
14	12	0	0	7412.307
14	13	0	0	4101.818
14	14	0	0	7237.95
15	0	0	0	9201.572

Tabla 4.2: Ejemplo de variable asteroide en el caso $N_r > N_{C,S}$

ampliamente estudiado y que gracias a su geometría singular, comparándolo con el resto de asteroides [37], se crea un código más robusto frente a asteroides atípicos.

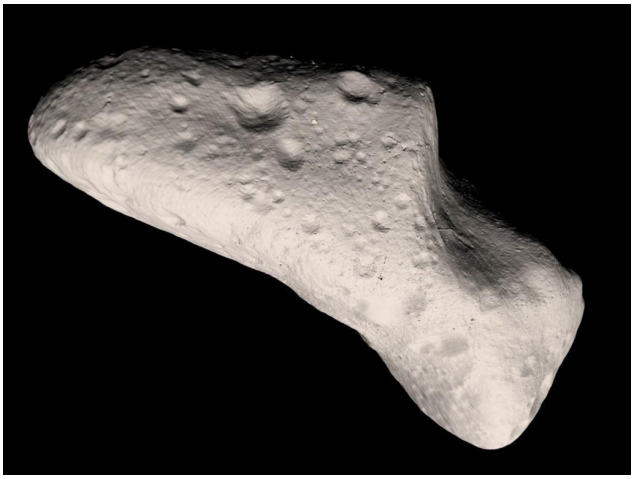

Figura 4.2: Fotografía del asteroide Eros 433. Fuente: [38].

Así pues, primero se han obtenido los coeficientes C_{nm} y S_{nm} del Eros 433 a partir de los datos recogidos por la misión espacial NEAR *(Near Earth Asteroid Rendezvous)* [39]. Después, tomando como referencia los coeficientes del Eros 433, se diferencian 2 tipos de asteroides generados:

- El primer tipo de asteroide aleatorio es el que difiere ligeramente del Eros 433. Esto se

ha conseguido mediante la variación de cada coeficiente del Eros 433 de forma aleatoria pero siempre acotando el máximo de dicha variación dentro de cierto porcentaje que puede modificar el usuario.

$$CS_{asteroide} = CS_{Eros433} \cdot (1 \pm random(porcentaje)) \tag{4.3}$$

Para ilustrar este concepto, en la Figura 4.3 se puede ver tanto el asteroide Eros 433, como uno generado de forma aleatoria similar a este con una cota de variación máxima de un 25 % . Ambos están representados con 13 armónicos.

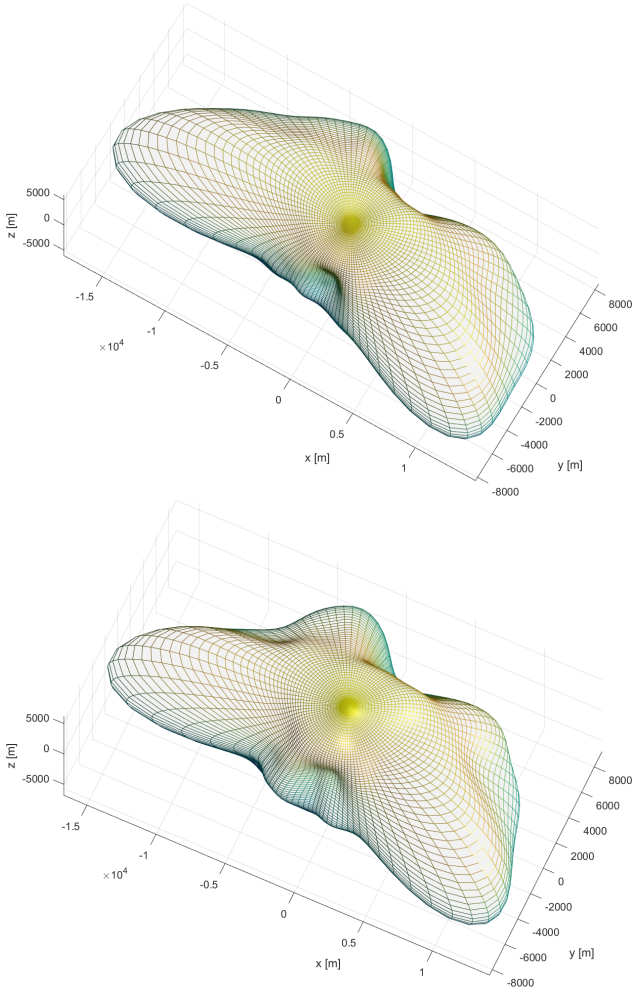

Figura 4.3: Eros 433 real (arriba) y asteroide similar a este (abajo)

- El segundo tipo de asteroide es el que se genera de forma casi completamente aleatoria. No es del todo completamente aleatorio debido a que el coeficiente C_{00}, comúnmente

denominado *radio de referencia*, se ha establecido el mismo que el del Eros 433 para todos los asteroides generados. Esta decisión se tomó para que todos los asteroides tuvieran el mismo orden de magnitud para obtener una mayor visibilidad a la hora de calcular las métricas. El resto de coeficientes se generan de forma aleatoria pero siempre siguiendo ciertos patrones y tendencias impuestas para que las geometrías resultantes tengan formas lo más parecidas posibles a asteroides esféricos. Además, estas restricciones evitan que se generen geometrías imposibles para un asteroide, tales como formas afiladas o geometrías que se pliegan sobre sí mismas. En la Figura 4.4 se puede ver un ejemplo de un asteroide generado de esta manera.

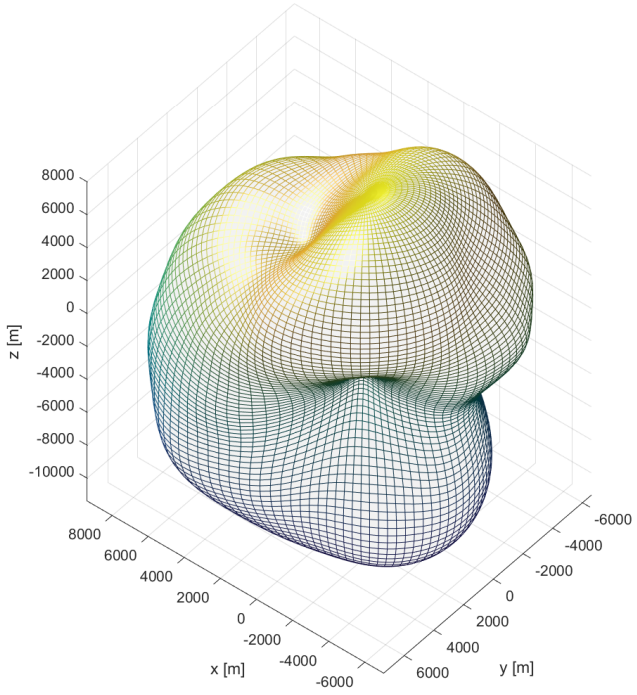

Figura 4.4: Ejemplo de asteroide aleatorio

En resumen, en la Section 1 se generan una combinación de ambos tipos de asteroides que sentarán la base de datos para las secciones venideras. Cabe destacar que la proporción de asteroides de un tipo y de otro puede ser modificada por el usuario y que ambos tipos de asteroides se barajan de forma aleatoria para evitar que la red neuronal aprenda dependencias no deseadas.

4.2. Section 2

Esta sección se encarga de recibir, procesar y preparar los datos provenientes de la Section 1 para su posterior introducción a la red neuronal en la Section 3.

4.2.1. Procesamiento

Lo primero que se realiza tras la generación de los asteroides es un filtrado y una separación de las columnas definidas por la variable asteroide, anteriormente definida en la Section 1. Esto es, por un lado, se eliminan los ceros introducidos de forma artificial previamente por cuestiones de tamaños de variables y, por otro lado, se separa en dos la variable asteroide. La separación consiste en dejar en una variable ambos coeficientes de armónicos esféricos C_{nm} y S_{nm}, es decir, las 2 primeras columnas de la variable asteroide y, en otra variable, la componente radial $r(\lambda, \phi)$.

Esta separación de variables se ha realizado con el objetivo de facilitar el manejo e identificación de las componentes que forman el asteroide de ahora en adelante.

El siguiente proceso al que se someten los datos es a la estandarización. Tal y como se introdujo en la Sección 3.2, las redes neuronales tienen una estrecha relación con las distribuciones normales y estas son capaces de aprender mejor la tarea objetivo. Además de esto, al estandarizar los datos se elimina el hecho de que haya varios órdenes de magnitud de diferencia entre los datos y, de esta manera, se previene la posibilidad de que la red neuronal priorice unos datos frente a otros.

Esta estandarización se realiza primero calculando la media y la desviación estándar mediante las Ecuaciones 3.7 y 3.9, respectivamente, de tanto los armónicos esféricos como la componente radial r de forma separada y para cada uno de los asteroides generados. Además, cabe destacar que en el caso de los armónicos esféricos esto se hace por separado para cada uno de los armónicos, es decir, se calcula la media y la desviación estándar por bloques. Por ejemplo, para los coeficientes de grado $n = \varepsilon$ y órdenes $m = 0, 1, \ldots, \varepsilon$ se obtendría una media y una desviación estándar encargadas de estandarizar dicho bloque de armónicos. Una vez hecho esto, los datos estandarizados se obtienen mediante la ecuación:

$$Data_{stand} = \frac{Data - \mu}{\sigma} \tag{4.4}$$

4.2.2. Preparación

Una vez se tienen los datos estandarizados, comienzan las divisiones del *dataset* de todos los asteroides generados tal y como se puede observar en la Figura 4.1. La primera gran división se trata de la diferenciación entre los *Train Asteroids* y los *Test Asteroids*: los primeros, se encargarán exclusivamente del entrenamiento de la red neuronal; y los segundos, del cálculo de las métricas. Esta partición normalmente se establece como un 80/20, es decir, un 80 % de los asteroides generados se destinarán al entrenamiento de la red neuronal y el 20 % restante se encargará del cálculo de las métricas.

A continuación, los *Train Asteroids* vuelven a dividirse entre *Input Asteroids* y *Validation Asteroids*. Ambos *datasets* son posteriormente alimentados a la red neuronal. El propósito de los *Input Asteroids* es el entrenamiento de la red neuronal, mientras que los *Validation Asteroids* actúan como forma de testar la red durante el entrenamiento con asteroides que están fuera de éste. El porcentaje destinado para cada uno de estos *datasets* también se ha establecido como en el caso anterior como un 80 % para los *Input Asteroids* y un 20 % para los *Validation Asteroids*. Más adelante aún así, en el capítulo de Resultados, se comprobará que se podría destinar un menor porcentaje de *Validation Asteroids*, ya que estos no tienen un gran impacto en el entrenamiento de la red.

4.3. Section 3

En esta sección primero se detallará cómo se estructuran las redes *MLP* y *CNN* usadas en este proyecto y después se explicará cómo se procesan los *Test Asteroids* y *Predicted Asteroids* para su posterior estudio de métricas.

4.3.1. Estructura de las redes *MLP*

Tanto la creación como el entrenamiento de las redes neuronales han sido realizados mediante el uso de la función integrada `trainNetwork` de *MATLAB*. Esta función consta de

4 argumentos de entrada:

- **Features:** Argumento de entrada de la red neuronal. En el caso del proyecto será la componente radial r, es decir, la tercera columna de la variable asteroide, del *dataset Input Asteroids*. Esta variable será referenciada dentro del código como `input_data`.

- **Responses:** Argumento de salida de la red neuronal. En este caso se trata de las 2 primeras columnas de la variable asteroide, es decir, de los armónicos esféricos C_{nm} y S_{nm}. En realidad, lo que se va a hacer es entrenar una red neuronal para cada uno de los 2 coeficientes, es decir, habrá una red neuronal cuyo argumento de salida sea exclusivamente los armónicos C_{nm} y otra red cuya salida sean los S_{nm}. Se ha tomado esta decisión de diseño debido al hecho de que las redes neuronales mejoran su precisión a mayor relación entre el número de entradas y salidas de estas [40]. Además de ello, esto soluciona una serie de problemas encontrados debido a discrepancias de dimensiones de variables que presenta la función integrada `trainNetwork` para el caso del proyecto sin que haya ningún tipo de detrimento por hacerlo. Dentro del código, la variable encargada del argumento de salida se denomina `target_data`.

- **Layers:** Este argumento se trata de una lista de las capas que van a conformar la red neuronal. Existen multitud de capas distintas [2] y formas de secuenciarlas en función de la necesidad de la red neuronal. Tal y como se introdujo en la Sección 3.3, las redes neuronales constan, a grandes rasgos, de la capa de entrada o *input layer*, las capas intermedias o *hidden layers* y la capa de salida u *ouput layer*:

  - ***Input layer:*** Esta capa debe de contener un número de nodos igual al número de *Features* de la red. Además, puesto que en el proyecto se va a trabajar con matrices numéricas, se va a usar como *input layer* la capa `sequenceInputLayer (inputSize)`, donde `inputSize` indica el número de nodos de esta.

  - ***Hidden layers:*** Este conjunto de capas es el que realmente define la estructura central de la red neuronal. Por ello, tal y como se pudo observar como ejemplo en la Figura 3.6, la red constará en este caso de una secuencia de 2

[2] `https://www.mathworks.com/help/deeplearning/ug/list-of-deep-learning-layers.html`

`fullyConnectedLayer` intercaladas con capas `sigmoidLayer`, las cuales limitan los datos en el intervalo $(0, 1)$ y mejora el aprendizaje de la red. Se ha establecido una secuencia de 2 pares de estas capas debido a que se ha comprobado (en el Capítulo 5 se demostrará) que son suficientes (con el debido número de nodos por capa `fullyConnectedLayer`) para que la red sea capaz de aprender el objetivo del proyecto con suficiente precisión para el *dataset* establecido de entrenamiento. Finalmente, además de esta secuencia de capas, se ha introducido inmediatamente después de la *input layer* una capa `flattenLayer` la cual tiene el objetivo de colapsar todas las dimensiones de la entrada de la red en una sola para que la red sea capaz de manejar los datos de forma interna con más facilidad.

- ***Output layer:*** Para la capa de salida de la red se debe utilizar, de manera análoga a la *input layer*, una `fullyConnectedLayer` cuyo número de nodos sea igual al número de `Responses`, además de una capa `regressionLayer` la cual computa el error medio cuadrático definido por la Ecuación 3.2 y permite la salida de la red.

Así pues, la lista de capas utilizada para la red neuronal es, de forma genérica:

```
layers = [ ...
    sequenceInputLayer(inputSize)
    flattenLayer()
    fullyConnectedLayer(numNodes)
    sigmoidLayer
    fullyConnectedLayer(numNodes)
    sigmoidLayer
    fullyConnectedLayer(ouputSize)
    regressionLayer
];
```

- **Options:** De la misma manera que el argumento *Layers* es el encargado de definir la estructura de la red, el argumento *Options* es el que dicta cómo va a entrenarse dicha red neuronal. Se trata de una variable interna de *MATLAB* denominada `trainingOptions (solverName, Name=Value)` cuyos argumentos de entrada son:

  - `solverName`: Se trata del optimizador que va a usar la red neuronal para entrenar. Se puede elegir entre 3 casos: `'sgdm'`, `'rmsprop'` y `'adam'`. Cada uno de ellos

tiene a su vez diferentes parámetros que se pueden modificar [3], para el caso de las redes *MLP* se ha elegido el optimizador `'adam'` puesto que era el que mejores resultados obtenía.

- `Name=Value`: Son una o más opciones adicionales que se pueden especificar mediante pares nombre-valor para ajustar el entrenamiento de la red neuronal. Las opciones que se han usado en este caso para las redes *MLP* son:
  - `'MaxEpochs'`: Es el número total de épocas realizadas por el optimizador durante el entrenamiento. Una época es el paso completo de la totalidad del *dataset* de entrenamiento por el optimizador.
  - `'ValidationData'`: Es la introducción de los *Validation Asteroids* al entrenamiento. Estos se deben de introducir en formato celda con los *Features* como primer elemento y los *Responses* como segundo.
  - `'ValidationFrequency'`: Frecuencia, especificada en épocas, a la que la red utiliza los *Validation Asteroids* para calcular la precisión y pérdidas de validación.
  - `'InitialLearningRate'`: Es el ritmo de aprendizaje inicial, especificado como un escalar positivo. Un valor muy pequeño puede generar entrenamientos que tarden mucho tiempo en completarse. Sin embargo, un valor demasiado alto puede originar un entrenamiento de la red subóptimo o incluso la divergencia de esta.
  - `'LearnRateSchedule'`: Este parámetro controla el comportamiento del ritmo de aprendizaje y existen 2 opciones: `'none'` y `'piecewise'`. En el caso de elegir `'none'`, el ritmo de aprendizaje será el mismo para todo el entrenamiento y será igual al establecido como valor del `'InitialLearningRate'`. En el caso de elegir `'piecewise'`, el ritmo de aprendizaje se va actualizando cada cierto número de épocas siendo multiplicado por un factor estipulado. En el caso del proyecto se ha elegido la opción `'piecewise'` puesto que ofrece un mayor control a la hora de realizar los entrenamientos de las redes.

[3] `https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html`

- `'LearnRateDropPeriod'`: Periodicidad de épocas en las que se va actualizando el ritmo de aprendizaje de la red neuronal, especificado como número entero positivo.
- `'LearnRateDropFactor'`: Factor por el que es multiplicado el ritmo de aprendizaje cada vez que se cumple el número de épocas especificadas por el `'LearnRateDropPeriod'`. Se trata de un escalar con rango $[0, 1]$.
- `'Verbose'`: Booleano para indicar si se desea que se muestre información del progreso del entrenamiento en la ventana de comandos de *MATLAB*.
- `'ExecutionEnvironment'`: Esta opción especifica el recurso de *hardware* que se desea usar para el entrenamiento de la red. Durante el desarrollo del proyecto se ha escogido la opción de `'cpu'` y los entrenamientos se han realizado usando un ordenador personal con un procesador Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz.Además, se ha contado con la ayuda de un clúster con un procesador Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz facilitado por la ETSIT para los análisis más pesados.
- `'OutputFcn'`: Esta opción permite establecer una o más funciones que serán ejecutadas justo antes del entrenamiento, después de cada iteración y al terminar el entrenamiento. En el caso del proyecto, se ha utilizado una función que cambia el eje vertical de las gráficas de los entrenamientos para que esté en escala logarítmica para así conseguir una mejor interpretación visual.
- `'Plots'`: Se puede escoger como opciones `'none'` y `'training-progress'` si se quiere que se muestre una gráfica durante el transcurso del entrenamiento o no, respectivamente. En esta gráfica se muestra información relevante como el *RMSE* y las pérdidas tanto del entrenamiento como de la validación, además de mostrar a tiempo real las épocas, el tiempo transcurrido, el ritmo de aprendizaje actual, etc.

Una vez explicados todos los argumentos de entrada de las opciones del entrenamiento de la red neuronal, este sería un ejemplo genérico de la variable `trainingOptions` de la red *MLP*:

```
optionsMLP = trainingOptions('adam', ...
    'MaxEpochs',epochs, ...
    'ValidationData',{inputValidation,targetValidation
      }, ...
    'ValidationFrequency', validationFreq, ...
    'InitialLearnRate',ILR, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',dropPeriod, ...
    'LearnRateDropFactor',dropFactor, ...
    'Verbose',0, ...
    'ExecutionEnvironment','cpu', ...
    'OutputFcn',@(x) makeLogVertAx(x, [true,true]), ...
    'Plots','training-progress');
```

Finalmente, con todos los argumentos definidos implicados en la creación y entrenamiento de la red neuronal *MLP*, simplemente habría que llamar a la función `trainNetwork` con la línea de código:

```
net = trainNetwork(input_data,target_data,layers,optionsMLP
   );
```

4.3.2. Estructura de las redes *CNN*

La estructura de las redes *CNN* es, en su gran mayoría, bastante similar a las redes *MLP*. A grandes rasgos, la estructura de la red *CNN* desarrollada en el proyecto es análoga a la red *MLP* con ciertas diferencias a la hora de introducir los *dataset* a la red y con las primeras capas de la variable `layers`. Así pues, estas redes también utilizan la función `trainNetwork` y, en este caso, sus 4 argumentos de entrada son:

- **Features:** Tal y como se desarrolló en la Sección 3.3.1, las redes *CNN* trabajan con imágenes como *input*, por lo que la variable `input_data` en este caso debe ser modificada para que tenga unas dimensiones de $grid(\lambda) \times grid(\phi) \times 1 \times N$, donde N es el número de asteroides en *Input Asteroids* y el 1 proviene de que las imágenes son monocromáticas.

- **Responses:** En el caso de la variable `target_data` no hay diferencias con el caso de

la red *MLP*. Se trata de los armónicos esféricos C_{nm} y S_{nm} separados, es decir, un coeficiente para cada una de las 2 redes neuronales *CNN* a entrenar.

- **Layers:** Para las capas de la *CNN*, tal y como se puede observar en la Figura 3.8, la parte de clasificación/regresión es la misma que en las redes *MLP* y son las capas iniciales las que son exclusivas de este tipo de redes. Así pues, las capas son:

  - ***Input layer:*** En el caso de las *CNN* se debe utilizar la capa de entrada `imageInputLayer(inputSize)`, donde en este caso `inputSize` debe ser el tamaño de una imagen, el cual fue establecido con aterioridad como $grid(\lambda) \times grid(\phi) \times 1$.

  - ***Hidden layers:*** Tal y como se introdujo en la Sección 3.3.1 y, gráficamente, en la Figura 3.8, la parte convolucional de las *CNN* son encadenaciones de conjuntos de una capa de filtros, una capa de activación y otra de *pooling*. En el caso del proyecto se ha optado por 2 encadenaciones de estos conjuntos.

    Para la capa de filtros se utiliza la capa `convolution2dLayer(filterSize,NumFilters,Name,Value)` donde: `filterSize`, es el tamaño del filtro, el cual se puede introducir como un vector $[altura, anchura]$ o como un número entero el cual produce un filtro cuadrado cuyos lados son iguales a dicho número; `NumFilters`, es el número de filtros; y `Name,Value`, pueden ser una o más opciones adicionales que se pueden especificar por pares. En el caso del proyecto, solo se ha establecido la opción adicional de `'Padding','same'`, ya que el tamaño de imagen no es lo suficientemente grande como para que sea de interés reducirlo mediante los filtros.

    A continuación, antes de la capa de activación, se ha introducido una capa de normalización denominada `batchNormalizationLayer`. Esta capa se incluye debido a que en las redes *CNN* aumenta la velocidad de los entrenamientos y reduce la sensibilidad frente a las inicializaciones de la red.

    Despúes, se incluye la capa de activación, que en este caso se ha utilizado una *ReLU*, explicada en la Sección 3.3, mediante la capa `reluLayer`. Cabe destacar que para el caso del proyecto no se va a utilizar a continuación una capa de *pooling*

por el mismo motivo que se ha elegido la opción de *same padding* en los filtros, por el hecho de que el tamaño de la imagen no es tan grande como para necesitarlo.

Finalmente, después de concluir la parte convolucional de la *CNN*, se forman las capas de la parte de regresión de manera análoga a las redes *MLP*.

- ***Output layer:*** En este caso no hay variación con las redes *MLP*, también se utiliza una capa `'fullyConnectedLayer'` con el número de nodos igual al de `Responses` seguida de una `'regressionLayer'`.

Con todo ello, las capas de una red neuronal *CNN* tienen una forma genérica tal que:

```
layers = [ ...
    imageInputLayer(inputSize)
    convolution2dLayer(filterSize,NumFilters,'Padding',
       'same')
    batchNormalizationLayer
    reluLayer
    convolution2dLayer(filterSize,NumFilters,'Padding',
       'same')
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(numNodes)
    sigmoidLayer
    fullyConnectedLayer(numNodes)
    sigmoidLayer
    fullyConnectedLayer(ouputSize)
    regressionLayer];
```

- **Options:** De igual manera a las redes *MLP*, las redes *CNN* también usan la función `trainingOptions(solverName, Name=Value)`. Por ello, y debido a que las opciones escogidas para modificar son las mismas que en las redes *MLP*, sólo se va a detallar la única opción que ha sufrido cambios, la cual es:

  - `'solverName'`: En el caso de la red *CNN*, se ha optado por elegir el optimizador `'sgdm'` en vez de `'adam'`, puesto que se ha detectado que funciona mejor para las redes convolucionales y para el caso del proyecto en concreto.

Así pues, las opciones de la red *CNN* quedaría con la forma genérica:

```
optionsCNN = trainingOptions('sgdm', ...
    'MaxEpochs',epochs, ...
    'MiniBatchSize', batchSize, ...
    'ValidationData',{inputValidation,targetValidation
      }, ...
    'ValidationFrequency', validationFreq, ...
    'InitialLearnRate',ILR, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',dropPeriod, ...
    'LearnRateDropFactor',dropFactor, ...
    'Verbose',0, ...
    'ExecutionEnvironment','cpu', ...
    'OutputFcn',@(x) makeLogVertAx(x, [true,true]), ...
    'Plots','training-progress');
```

Finalmente, al igual que con la red *MLP*, la creación y entrenamiento de la red *CNN* se inicia con la simple línea de código:

```
net = trainNetwork(input_data,target_data,layers,optionsCNN
   );
```

4.3.3. Predicción de asteroides

Una vez ya se tienen las redes neuronales entrenadas, el siguiente paso a realizar es, recordando la Figura 4.1, introducir los *Test Asteroids* en las redes para obtener los *Predicted Asteroids* y con ambos *dataset* obtener las métricas de las redes para poder conocer cómo de precisas son.

Para predecir con una red neuronal cualquiera, ya sea con la *MLP* o la *CNN*, se utiliza la función integrada de *MATLAB* `pred = predict(net, input_test)`. En dicha función, el parámetro `net` se trata de la red neuronal ya entrenada y el parámetro `input_test` es el equivalente de la variable `input_data` del entrenamiento de la red pero en este caso del *dataset Test Asteroids*. Esto es, son los `Features` de la red, por lo que en función de si se trata de la red *MLP* o *CNN* habrá que tener especial cuidado con que tengan las dimensiones correctas para cada tipo de red.

Por otro lado, la función `predict` devuelve los `Responses` de las redes y los almacena en este caso en la variable denominada `pred`. A continuación, es necesario recordar que en la Sección 4.2.1 se estandarizaron todos los *dataset*, por lo que la variable `pred` también son datos estandarizados. Por ello, para poder hacer uso de ellos: primero, se hace una copia de esos datos para el estudio de algunás métricas; y después, se procede a desestandarizarlos. Además, se sigue el mismo procedimiento para los *Test Asteroids*. Así pues, recordando que la estandarización de los armónicos esféricos se hizo por bloques de armónicos, para obtener la fórmula utilizada para desestandarizar basta con despejar de la Ecuación 4.4:

$$Data = Data_{stand} \cdot \sigma + \mu \tag{4.5}$$

Finalmente, recordando que los `Responses` de las redes neuronales son los armónicos esféricos C_{nm} y S_{nm}, para poder construir el *dataset Predicted Asteroids* es necesario volver a hacer uso de la Ecuación 2.36 para obtener la componente radial $r(\lambda, \phi)$, la cual se corresponde con la última columna de la variable asteroide.

4.4. Section 4

En esta sección lo que se va a realizar es el estudio del rendimiento de las redes neuronales entrenadas mediante el uso de ciertas métricas. Para obtener dichas métricas, se parte de los *dataset Test Asteroids* y *Predicted Asteroids*, tanto estandarizados como sin estandarizar, descritos en la sección anterior.

Las métricas a realizar se pueden dividir en 2 grandes categorías: métricas gráficas y métricas numéricas.

4.4.1. Métricas gráficas

Se le denomina métricas gráficas a las que permiten al usuario discernir de un vistazo cuál ha sido el rendimiento de la red neuronal a *grosso modo*. Dentro de esta categoría de métricas se distinguen 2 casos:

- **Armónicos reales *vs* Armónicos predichos**

  Esta métrica se trata de la representación gráfica de todos los armónicos esféricos reales *vs* los predichos, es decir, la comparativa entre las 2 primeras columnas de los *Test Asteroids* y los *Predicted Asteroids* sin estandarizar. Esta comparativa se hará en 2 gráficas yuxtapuestas, una para los coeficientes C_{nm} y otra para los S_{nm}.

  Cabe destacar que a pesar de que los *Test Asteroids* comprenden un 20 % del total de asteroides generados, estos siguen siendo del orden de miles. Por ello, para evitar generar demasiadas gráficas, el usuario puede establecer el número de gráficas de esta categoría que quiere mostrar. Además, estas gráficas se escogerán de entre todos los asteroides de test/predichos de manera aleatoria cada vez.

  En la Figura 4.5 se muestra un ejemplo de cómo serían este tipo de métricas gráficas.

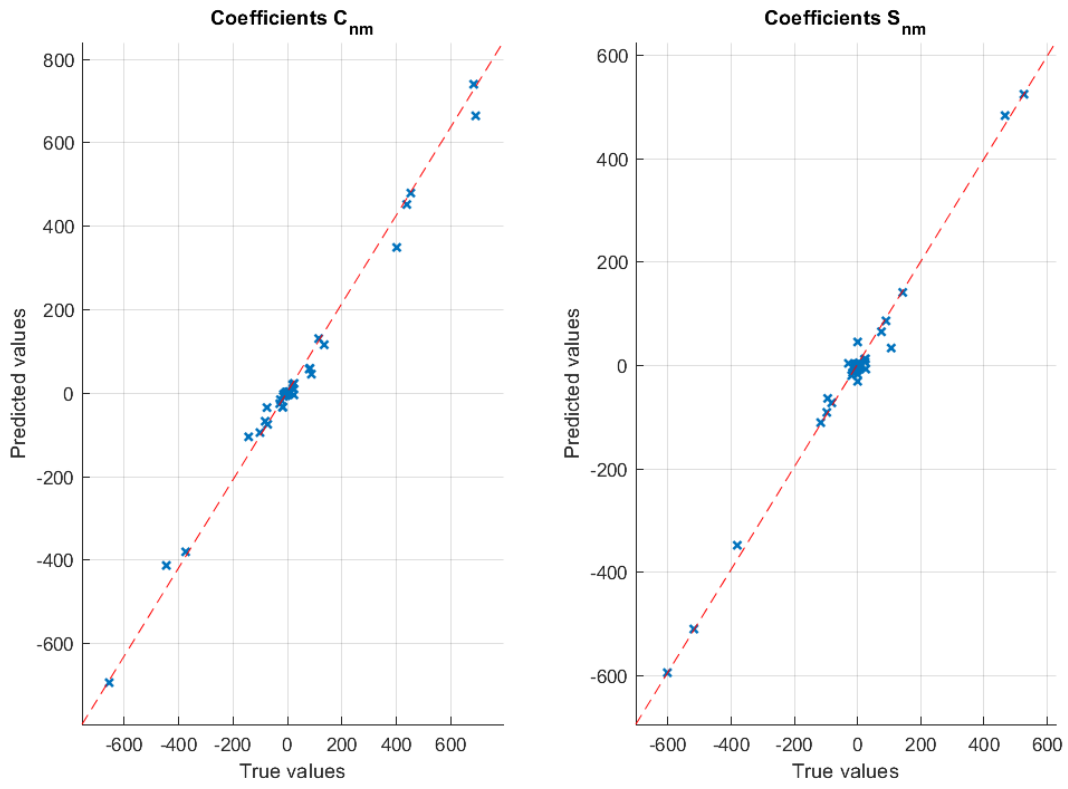

Figura 4.5: Comparativa de los armónicos esféricos predichos frente a los reales.

- **Geometría real *vs* Geometría predicha**

  Esta métrica consiste en la representación gráfica de los asteroides reales frente a los predichos. Al igual que en la otra métrica gráfica y, en consonancia con esta, se van comparar gráficamente únicamente los mismos asteroides escogidos de forma aleatoria para la comparativa de armónicos. Esto se ha establecido así para poder tener ambas gráficas sobre los mismos asteroides y así ser capaces de extraer más información útil de ellas.

  Además, el código se ha escrito de forma que el usuario es capaz de seleccionar el paso de la malla, es decir, los parámetros $grid(\lambda)$ y $grid(\phi)$ definidos en la Sección 4.1, utilizado para estas representaciones gráficas. Esto se ha establecido de forma independiente del paso de la malla usado en los asteroides de entrenamiento de las redes por si el usuario desea ver los asteroides con una mayor o menor definición de detalles.

  En la Figura 4.6 se puede ver la comparativa gráfica del asteroide real frente al predicho que se corresponden con los armónicos esféricos comparados en la Figura 4.5 con un mallado de 50 pasos.

Por otro lado, tal y como se explicó en la Sección 4.1.1, ya que se han utilizado asteroides similares al Eros 433 para los entrenamientos de las redes y que se dispone de los datos reales de este, se aprovecha y se calculan todas las métricas para el Eros 433 real, tanto las gráficas como las numéricas.

4.4.2. Métricas numéricas

Las métricas numéricas son las que, una vez revisadas las métricas gráficas como primer acercamiento, van a permitir conocer de manera cuantitativa los rendimientos de las redes neuronales entrenadas. Esto es, métricas gracias a las cuales se podrán establecer criterios de rendimiento a partir de los cuales se podrán comparar distintos entrenamientos entre sí y/o unas redes con otras.

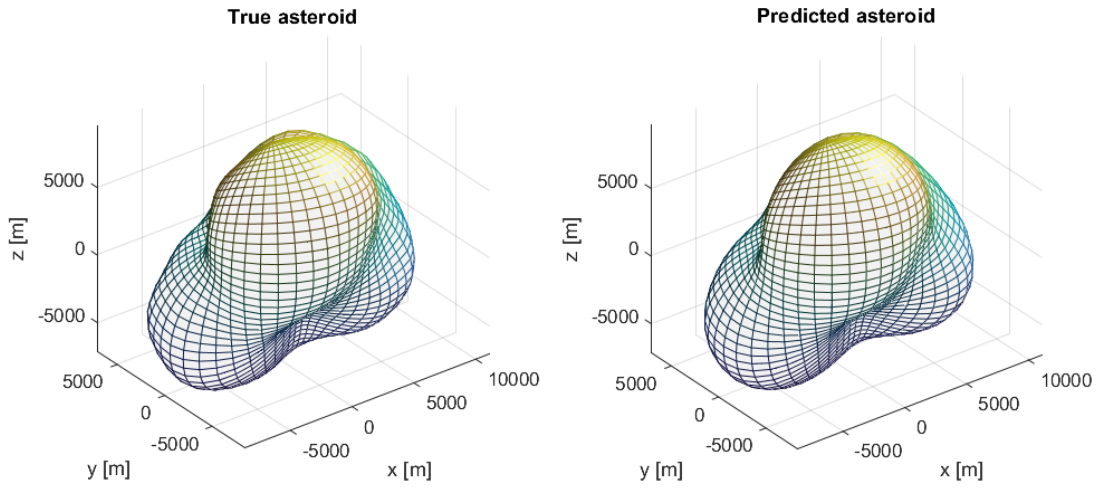

Figura 4.6: Comparativa gráfica de un asteroide predicho frente al real.

Así pues, las métricas numéricas establecidas para este proyecto son:

- **Error Absoluto Global**

  Con esta métrica se obtiene una media del error absoluto total cometido por las redes neuronales sobre los armónicos esféricos predichos frente a los reales. Para calcularla se ha tomado como referencia la métrica *MAE* explicada en la Sección 3.1.2.

  Primero se parte de la diferencia en valor absoluto de los coeficientes sin estandarizar. Después, se hace un normalizado mín.-máx para limitar en el rango $[0, 1]$ estas diferencias. A continuación, se hace un sumatorio sobre toda la dimensión de los coeficientes para obtener el error absoluto total de un asteroide. Finalmente, se hace una media sobre todos los asteroides testados. De forma algebraica se puede expresar como:

$$Diff = |SH_{real} - SH_{pred}|$$
$$AbsError = \frac{1}{n_{asteroids}} \sum_{n=1}^{n_{asteroids}} \left(\sum_{i=1}^{N_{C,S}} \frac{Diff_i - \text{mín}\,(Diff)}{\text{máx}\,(Diff) - \text{mín}\,(Diff)} \right)_n \tag{4.6}$$

donde $N_{C,S}$ es la longitud de los armónicos, explicada en la Sección 4.1. Cabe recordar que como se ha entrenado una red para cada coeficiente de los armónicos, SH será en un caso los coeficientes C_{nm} y en otro los S_{nm}. Por ello, se tendrán 2 errores absolutos globales, uno para cada red entrenada.

- **Error Absoluto por Armónicos**

  Esta métrica es la equivalente a la anterior pero, en este caso, dividida por armónicos. Es decir, se tendrá un error absoluto para cada uno de los $k \in [0, n_{max}]$ armónicos generados. Algebraicamente se puede expresar como:

$$AbsErrorSH|_{k^{th}harmonic} = \frac{1}{n_{asteroids}} \sum_{n=1}^{n_{asteroids}} \left(\sum_{i=1}^{N_k} |SH_{real,i} - SH_{pred,i}| \right)_n \tag{4.7}$$

  donde en este caso SH ya son los coeficientes normalizados y N_k es una longitud variable dependiendo del k-ésimo armónico que se esté calculando, en concreto:

$$N_k = k + 1 \tag{4.8}$$

  Así pues, en la Figura 4.7 se muestra un ejemplo de cómo sería la representación gráfica de esta métrica tanto para el conjunto de los *Test Asteroids* como del Eros 433 real por separado.

- **Precisión Global**

  Esta métrica se trata del coeficiente de determinación o R^2 introducido en la Sección 3.1.2 aplicado sobre los armónicos esféricos. En este caso se utilizan los armónicos dimensionados. La manera de obtener esta métrica es mediante la siguiente ecuación:

$$R^2 = \frac{1}{n_{asteroids}} \sum_{n=1}^{n_{asteroids}} \left(1 - \frac{\sum_{i=1}^{N_{C,S}} (SH_{real,i} - SH_{pred,i})^2}{\sum_{i=1}^{N_{C,S}} (SH_{real,i} - mean(SH_{real}))^2} \right)_n \tag{4.9}$$

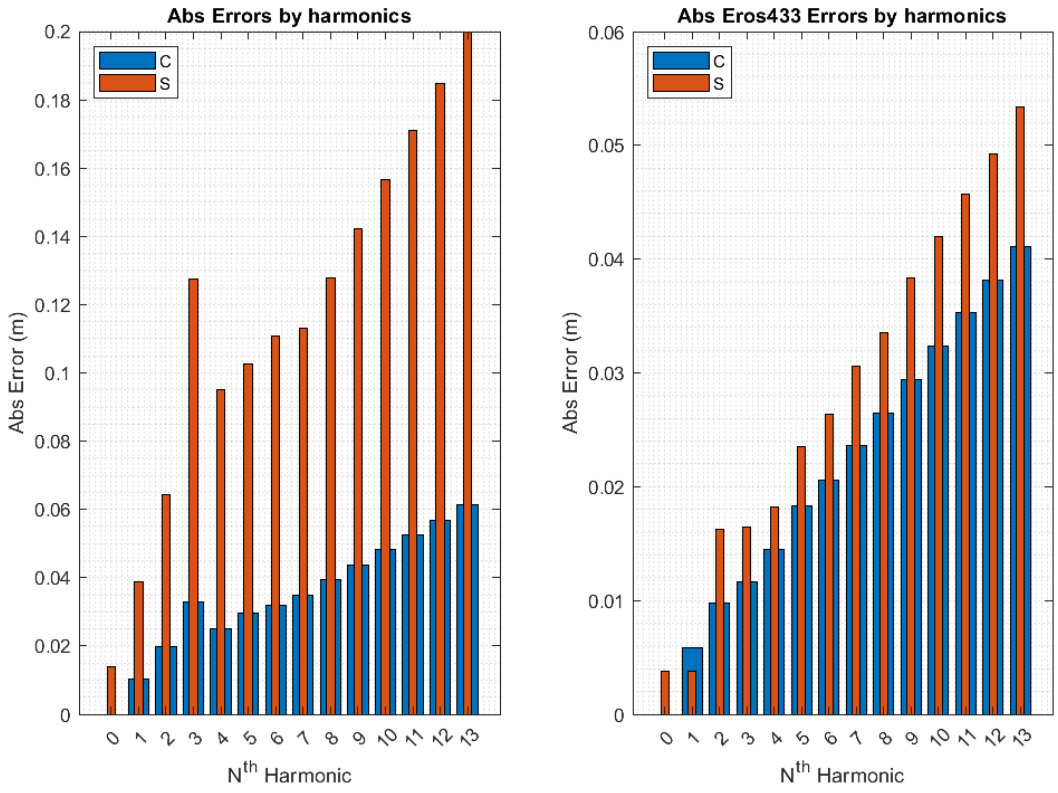

Figura 4.7: Ejemplo de la métrica Error Absoluto por Armónicos.

Cabe recordar que el coeficiente de determinación se encuentra en el rango $(\infty, 1]$, pudiendo adoptar valores negativos, los cuales indicarían que la red tiene un precisión peor que predecir simplemente la media. Por otro lado, un $R^2 = 0$ indica que la red predice exactamente lo mismo que predecir la media. Finalmente, un $R^2 = 1$ significaría que la red predice sin errores, por lo que el objetivo es intentar que las redes del proyecto se acerquen lo máximo posible a la unidad. Además, al igual que con el resto de métricas, se obtendrá un R^2 para la red que predice los C_{nm} y otro para la red que predice los S_{nm}.

- **Error Porcentual Global**

Para esta métrica se han utilizado los armónicos esféricos normalizados, a los cuales también se les ha aplicado un adimensionalizado respecto a unos valores de referencia. Esto se ha hecho debido a que la fórmula contiene una división sobre los armónicos reales y, si se hubieran utilizado los coeficientes normalizados cuyo rango es de $[0, 1]$, podrían

haber existido indeterminaciones. Así pues, los armónicos esféricos ya normalizados, tanto reales como predichos, se han adimensionalizado entre el rango $[-1, 1]$ mediante:

$$SH = \frac{SH - SH_{min}}{SH_{max} - SH_{min}} \tag{4.10}$$

siendo $SH_{max} = 1$ y $SH_{min} = -1$.

Finalmente, el error porcentual global se obtiene como:

$$Error\,\% = \frac{1}{n_{asteroids}} \sum_{n=1}^{n_{asteroids}} \left(\frac{100}{N_{C,S}} \sum_{i=1}^{N_{C,S}} \left| \frac{SH_{real,i} - SH_{pred,i}}{SH_{real,i}} \right| \right) \tag{4.11}$$

- **Error Porcentual por Armónicos**

Al igual que se hizo con las métricas de Error Absoluto Global y Error Absoluto por Armónicos, se sigue la misma línea con el Error Porcentual Global y el Error Porcentual por Armónicos. Estas métricas divididas por armónicos están hechas con la idea de buscar posibles patrones en los resultados y poder discernir si a partir de cierto armónico las métricas mejoran o empeoran. Con todo ello, la fórmula utilizada es análoga a la Ecuación 4.11 y tiene la forma:

$$Error\,\%|_{k^{th}harmonic} = \frac{1}{n_{asteroids}} \sum_{n=1}^{n_{asteroids}} \left(\frac{100}{N_k} \sum_{i=1}^{N_k} \left| \frac{SH_{real,i} - SH_{pred,i}}{SH_{real,i}} \right| \right) \tag{4.12}$$

donde N_k vuelve a ser la longitud variable dependiente del armónico que se esté calculando, definida en la Ecuación 4.8.

Finalmente, un ejemplo gráfico de este tipo de métrica se puede observar en la Figura 4.8.

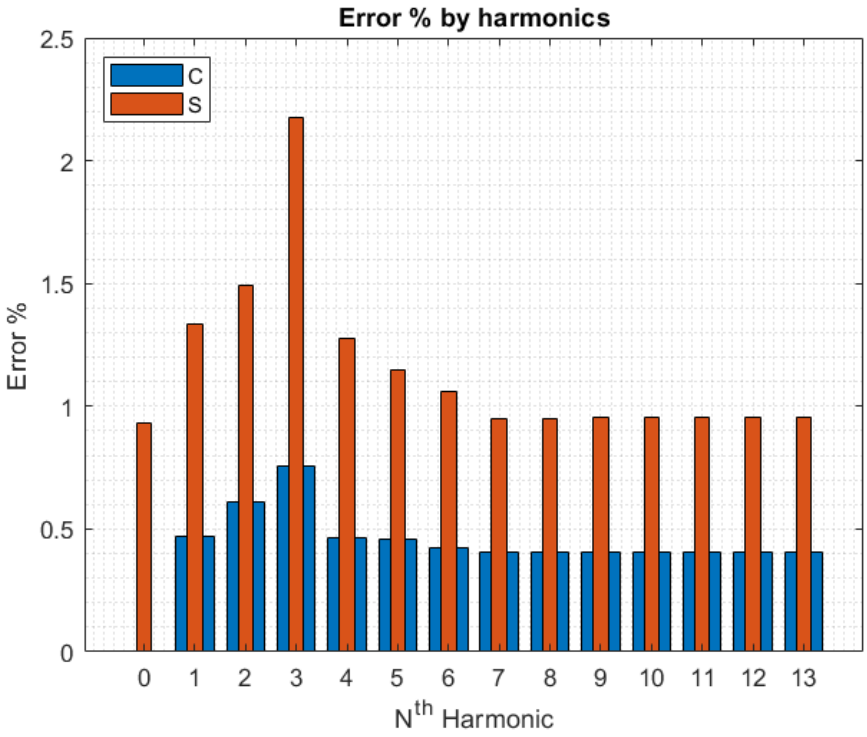

Figura 4.8: Ejemplo de la métrica Error porcentual por Armónicos.

Capítulo 5

Resultados

En este capítulo, partiendo de las estructuras generales de las redes *MLP* y *CNN* desarrolladas en el Capítulo 4, se va a hacer un estudio de ambas redes con el objetivo de encontrar cuál es la que mejor resultados obtiene para el caso del proyecto.

Primero, debido a que la red *CNN* es, en cierta medida, una ampliación de la red *MLP* tal y como se vio en la Sección 4.3.2, se comienza realizando un estudio paramétrico más exhaustivo sobre las redes *MLP*. Con ello, se obtendrán conclusiones sobre qué parámetros tienen una repercusión más notable sobre las redes y cómo es esta influencia. A continuación, con toda la información recabada hasta el momento, se procederá a hacer un análisis paramétrico sobre las redes *CNN*, en concreto, un análisis sobre los parámetros que son exclusivos de este tipo de redes. Finalmente, una vez se conozcan las mejores combinatorias de parámetros para cada tipo de red, se entrenarán ambas redes con el mismo *dataset* independiente y se compararán sus métricas para encontrar la que obtuvo mejores resultados.

5.1. Análisis redes *MLP*

Para los análisis realizados sobre las redes *MLP* se distinguen 3 grandes grupos: análisis unidimensionales ($1D$), bidimensionales ($2D$) y análisis global (ND). Donde el número de dimensiones viene dado por la cantidad de parámetros de los entrenamientos que se van a ir modificando cada vez, siendo N el número total de parámetros con relevancia en los entrenamientos para modificar. Los análisis se han hecho en ese orden para primero encontrar y distinguir cuáles de todos los parámetros disponibles para el entrenamiento son los que

realmente tienen cierta importancia en los resultados y cuáles no. De esta manera, a la hora de ir ascendiendo a análisis con más dimensiones, por un lado se descartarán parámetros sin impacto y, por otro, permitirá cercar más los valores de los parámetros en la vecindad en la que estos proporcionan resultados decentes a las redes.

Para estos 3 grandes grupos de análisis se ha utilizado el mismo *dataset* de asteroides generados para que todas las redes trabajen con los mismos datos y así se puedan comparar entre sí. Por ello, tal y como se vio en la Sección 4.1, se debe establecer:

- El número de asteroides totales a generar $n_{asteroids}$.
- El tamaño de malla empleado para representar la geometría de los asteroides, es decir, la división de pasos de la longitud y latitud denominadas $grid(\lambda)$ y $grid(\phi)$ respectivamente. Dicha malla definirá la longitud N_r.
- El armónico de mayor orden de los asteroides n_{max}, el cual define la longitud $N_{C,S}$ mediante la Ecuación 4.1.

Así pues, para establecer un *dataset* común a todos los análisis que disponga del suficiente número de asteroides y definición de estos, se realiza un preanálisis sobre los 3 parámetros que definen la variable asteroide.

5.1.1. Preanálisis

Para la realización del preanálisis se han fijado unos parámetros de las opciones de entrenamiento de las redes que mediante la experiencia se conoce que proporcionan resultados aceptables. Estos parámetros quedan recogidos en la Tabla 5.1.

Neuronas por capa	**Epochs**	**V.Freq**	**ILR**	**Drop Period**	**Drop Factor**
100	8000	100	0.01	150	0.95

Tabla 5.1: Parámetros fijos de entrenamiento del preanálisis.

En cuanto a los parámetros de estudio del preanálisis, se ha optado por escoger los valores representados en la Tabla 5.2, formando un total de 8 ensayos. Cabe destacar que, en el caso

del tamaño de la malla, se establecerán el mismo número de pasos para la latitud y para la longitud puesto que se quiere obtener así una malla regular que tenga la misma definición en toda su superficie.

$n_{asteroids}$	Grid(λ) & Grid(ϕ)	n_{max}
1000	10	13
5000		
10000	20	
20000		

Tabla 5.2: Parámetros de entrenamiento del preanálisis.

Se ha utilizado un $n_{max} = 13$ debido a que se ha considerado que es un armónico lo suficientemente elevado como para poder captar los rasgos generales de los asteroides y un mínimo de su detalle superficial sin comprometer la carga computacional y temporal que conllevaría utilizar un armónico de mayor orden para este tipo de análisis. Además, a diferencia de la Figura 4.4 que utiliza un generador de asteroides más "esféricos", es decir, con menos protuberancias, se utiliza un generador de asteroides que los crea tal y como se muestra en la Figura 5.1.

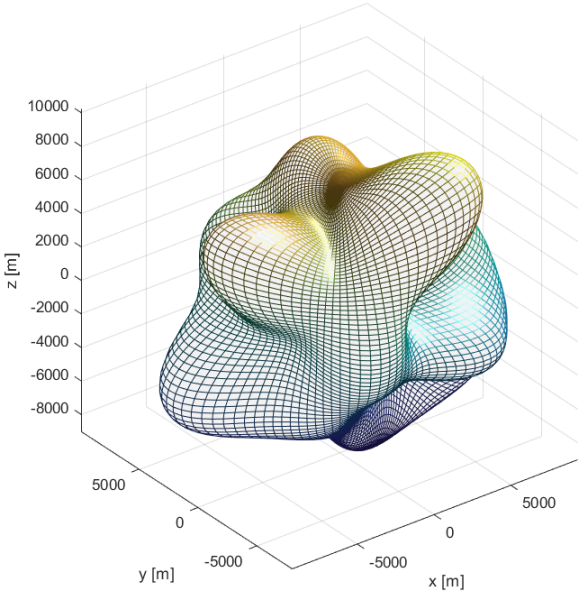

Figura 5.1: Asteroide aleatorio con $n_{max} = 13$.

Esto se ha hecho así para sobredimensionar en cierta medida las redes, puesto que a pesar de que existen infinidad de formas de asteroides [1], la mayoría poseen una forma más similar a la representada en la Figura 4.4.

Así pues, se realizan los 8 ensayos que surgen del establecimiento de los parámetros fijos de entrenamiento de la Tabla 5.1 y los parámetros de estudio de la Tabla 5.2 y se obtienen las métricas representadas en las Figuras A.1 y A.2

A partir de las métricas globales de la Figura A.1 se puede observar que, a medida que aumentan el número de asteroides utilizados, mejoran las métricas y, que esto ocurre para ambos tamaños de malla utilizados. Esta tendencia también se observa en la Figura A.2. Además, se puede extraer de todas las métricas que, para los ensayos con el mayor número de asteroides (20000), los resultados son mejores para el caso con mayor tamaño de malla. Sin embargo, también se puede observar que estos resultados no son mejores por un gran margen respecto a los resultados con la malla más pequeña. Por otro lado, los análisis con un tamaño de malla mayor conllevan un incremento en el tiempo de los ensayos considerable. Por ello, teniendo en cuenta que se van a realizar multitud de ensayos, se opta por escoger el menor tamaño de malla.

$n_{asteroids}$	Grid(λ) & Grid(ϕ)	n_{max}
20000	10	13

Tabla 5.3: Parámetros del dataset común para los análisis.

Por todo lo anterior, se establece el *dataset* común para el resto de análisis sucesorios a aquel con las condiciones de la Tabla 5.3.

5.1.2. Análisis unidimensional (1D)

Una vez se dispone ya de un *dataset* común con la suficiente definición y robustez para los análisis que se van a realizar, se comienzan los susodichos análisis por el unidimensional. Esto lo que quiere decir es que se van a establecer fijos todos los parámetros menos uno, el cual se variará y se estudiará su impacto en las métricas de las redes *MLP*.

Así pues, los parámetros sujetos a estudio serán los mismos que los que aparecen en la Tabla 5.1 a excepción de las épocas, las cuales se han prefijado en **8000 *epochs***. Esto se ha hecho así para que todas los análisis dispongan del mismo tiempo de entrenamiento y este no sea demasiado elevado, lo cual podría producir un sobreajustado (ver Sección 3.1.2). En resumen, se va a realizar un análisis unidimensional sobre los parámetros: *Validation Frequency*, *ILR*, *Drop Period*, *Drop Factor* y número de neuronas por capa.

5.1.2.1. *Validation Frequency*

Se comienzan los análisis por el parámetro *Validation Frequency*, ya que, tal y como se introdujo en la Sección 4.2.2, se tiene la hipótesis de que no va a tener un gran impacto en los entrenamientos. Si esto fuera así, se podría descartar como parámetro para análisis de mayores dimensiones.

Neuronas por capa	**ILR**	**Drop Period**	**Drop Factor**
100	0.01	150	0.95

(a) Hiperparámetros fijos.

V.Freq
50
100
250
500
750
1000
1500
2000
2500
3000

(b) Hiperparámetros variables.

Tabla 5.4: Hiperparámetros del análisis de la frecuencia de validación.

Los hiperparámetros utilizados en este análisis han sido los recogidos en la Tabla 5.4. Dicha combinación de parámetros ha generado redes neuronales con unas métricas globales y por harmónicos que se muestran, respectivamente, en las Figuras B.1 y B.2. Efectivamente, la hipótesis que se tenía de que el *Validation Frequency* no tendría un gran impacto en las redes

se corrobora con las métricas mencionadas. En las cuales se puede ver que la modificación de este parámetro no varía las métricas lo suficiente como para considerarlo relevante para el resto de análisis de mayor dimensionalidad. Por esto mismo, en la Sección 4.2.2 se dijo que la partición de los *Train Asteroids* en *Input Asteroids* y *Validation Asteroids* podría haber destinado menos asteroides para los de validación, puesto que se va a establecer un *Validation Frequency* relativamente alto y por lo tanto no se van a necesitar tantos asteroides para ello.

5.1.2.2. *Initial Learning Rate (ILR)*

Continuando con los análisis unidimensionales, se sigue con el parámetro encargado de establecer el ritmo de aprendizaje inicial de la red neuronal. Este parámetro junto con el *Drop Period* y el *Drop Factor*, serán los encargados de controlar el ritmo de aprendizaje de la red durante todo el entrenamiento.

Neuronas por capa	**V.Freq**	**Drop Period**	**Drop Factor**
100	2000	150	0.95

(a) Hiperparámetros fijos.

ILR
1
0.75
0.5
0.25
0.1
0.075
0.05
0.01
0.0075
0.005
0.001

(b) Hiperparámetros variables.

Tabla 5.5: Hiperparámetros del análisis del ritmo de aprendizaje inicial.

Los hiperparámetros utilizados en este análisis son los reflejados en la Tabla 5.5. Estos generan unas redes cuyas métricas están representadas en las Figuras B.3 y B.4. En dichas figuras, es fácil ver que existe un umbral para el valor $ILR = 0{,}1$ a partir del cual todas las

métricas mejoran sobremanera, por lo que para futuros análisis se escogerán valores iguales o menores a este para así obtener resultados más precisos.

5.1.2.3. *Drop Period*

Se sigue con el análisis del parámetro *Drop Period*, el cual se encarga de establecer el periodo de épocas a partir del cual el ritmo de aprendizaje es actualizado. Esto es, partiendo del *ILR*, cada cierto número de épocas determinadas por el *Drop Period*, el ritmo de aprendizaje es multiplicado por el parámetro *Drop Factor*. Así pues, los hiperparámetros establecidos para este análisis han sido los que aparecen en la Tabla 5.6.

Neuronas por capa	**V.Freq**	**ILR**	**Drop Factor**
100	2000	0.05	0.95

(a) Hiperparámetros fijos.

Drop Period
25
50
100
250
500
1000
1500
2500
3500
5000

(b) Hiperparámetros variables.

Tabla 5.6: Hiperparámetros del análisis del *Drop Period*.

De manera análoga al resto de análisis, los resultados de estos ensayos se ven reflejados en las métricas de las Figuras B.5 y B.6. En este caso, se puede ver que el valor de *Drop Period* = 25 es el único para el cual se obtienen unos resultados peores que el resto. A partir de dicho valor las métricas se estabilizan y no sufren una variación notable. Por ello, para futuros análisis se establecerán unos valores de *Drop Period* $\geqslant$ 50.

5.1.2.4. *Drop Factor*

Terminando con el conjunto de parámetros que modifican el ritmo de aprendizaje de las redes neuronales se tiene el *Drop Factor*, el cual establece el factor por el que es multiplicado el ritmo de aprendizaje cada número de épocas dictadas por el *Drop Period*. Para este ensayo se han utilizado los hiperparámetros representados en la Tabla 5.7.

Neuronas por capa	**V.Freq**	**ILR**	**Drop Period**
100	2000	0.05	250

(a) Hiperparámetros fijos.

Drop Factor
0.99
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55

(b) Hiperparámetros variables.

Tabla 5.7: Hiperparámetros del análisis del *Drop Factor*.

Al igual que en el resto de ensayos, las métricas resultantes se pueden ver en las Figuras B.7 y B.8. En este caso, se puede observar que, para la mayoría de métricas, a partir de un *Drop Factor* = 0,75 comienza un aumento de los errores y una bajada del *accuracy*. Por ello, para futuros análisis, con tal de ser conservadores se usarán valores de *Drop Factor* desde 0,99 hasta 0,8.

5.1.2.5. Número de neuronas por capa

Como último análisis unidimensional, se procede a hacer el estudio paramétrico del número de neuronas por capa de la red neuronal. Tal y como se explicó en la Sección 4.3.1, las redes del proyecto cuentan con 2 capas `fullyConnectedLayer`, las cuales hasta ahora han

contado con 100 neuronas cada una. Sin embargo, no tienen por qué tener el mismo número de neuronas cada una, por lo que se va a realizar a continuación el análisis con el objetivo de encontrar cuáles son las mejores combinaciones de neuronas para este caso. En concreto, se van a utilizar para este ensayo los parámetros recogidos en la Tabla 5.8.

V.Freq	**ILR**	**Drop Period**	**Drop Factor**
2000	0.05	250	0.99

(a) Hiperparámetros fijos.

Neuronas 1ªCapa	**Neuronas 2ªCapa**
25	25
50	50
100	100
150	150
200	200

(b) Hiperparámetros variables.

Tabla 5.8: Hiperparámetros del análisis del número de neuronas por capa.

Los resultados de estos análisis se pueden observar en las métricas de las Figuras B.9 y B.10. De dichas métricas se puede ver que, por lo general, los resultados empeoran bastante en cuanto la primera capa tiene más de 100 neuronas. Para los casos en los que la primera capa tiene 100 neuronas o menos se tienen unos resultados buenos de manera generalizada y, de forma específica, se obtienen unos especialmente buenos para los casos en los que el número de neuronas de la primera capa es mayor que el número de neuronas de la segunda.

5.1.3. Análisis bidimensional (2D)

Una vez realizados los análisis unidimensionales sobre las redes *MLP*, se continúa con los análisis bidimensionales. Para dichos ensayos, se trabaja con el mismo *dataset* común utilizado para los análisis $1D$ desarrollado en la Sección 5.1.1. Además, también se ha utilizado el mismo número de épocas para todos los entrenamientos, 8000, al igual que en los análisis $1D$ para garantizar las mismas condiciones.

Así pues, en los análisis bidimensionales se han tenido en cuenta las conclusiones obtenidas en los análisis unidimensionales. Esto quiere decir que tanto los parámetros fijos como los

variables que se van a usar van a estar más acotados dentro del rango de valores que producen buenos resultados. De esta manera, se podrán descartar entrenamientos que se conoce que producen malas métricas y a la vez se intenta ser más preciso con los entrenamientos que sí producen buenos resultados.

Según los resultados del análisis unidimensional del *Validation Frequency* explicados en la sección anterior, este parámetro ha sido descartado de los análisis bidimensionales, lo cual ocasiona que solamente se vayan a analizar los siguientes pares de parámetros: *ILR - Drop Factor*, *ILR - Drop Period*, *Drop Factor - Drop Period*, Neuronas - *ILR*, Neuronas - *Drop Factor* y Neuronas - *Drop Period*.

5.1.3.1. *ILR - Drop Factor*

Se comienza con el análisis bidimensional de los parámetros del ritmo de aprendizaje inicial *(ILR)* y el *Drop Factor*. De manera análoga a los análisis unidimensionales, se recogen los hiperparámetros fijos y variables de los entrenamientos en tablas, en este caso, los parámetros escogidos son los que aparecen en la Tabla 5.9.

Neuronas por capa	**Drop Period**	**V.Freq**
100	250	2000

(a) Hiperparámetros fijos.

ILR	**DF**
0.075	0.99
0.05	0.95
0.01	0.9
0.0075	0.85
0.005	0.8

(b) Hiperparámetros variables.

Tabla 5.9: Hiperparámetros del análisis *ILR - Drop Factor*.

De igual manera, los resultados de este análisis se pueden consultar en las métricas representadas en las Figuras C.1 y C.2. De las cuales se puede observar que, por lo general, se han obtenido unos resultados homogéneos para los valores de los hiperparámetros escogidos. Esto indica que dichos valores están dentro de unos límites para los cuales se puede asegurar con cierta seguridad que van a producir redes neuronales con buenos resultados. Además, se destaca que los peores resultados ocurren para un $ILR = 0{,}075$ y que, por el contrario, los mejores ocurren para la combinación de $ILR = 0{,}05$ y $DF = 0{,}99$.

5.1.3.2. *ILR - Drop Period*

Continuando con los análisis bidimensionales, se ensayan ahora el *ILR* y el *Drop Period*. Para este caso, se escogen los valores recogidos en la Tabla 5.10 para los hiperparámetros de entrenamiento.

Neuronas por capa	Drop Factor	V.Freq
100	0.99	2000

(a) Hiperparámetros fijos.

ILR	DP
0.075	250
0.05	500
0.01	1000
0.0075	1500
0.005	2000

(b) Hiperparámetros variables.

Tabla 5.10: Hiperparámetros del análisis *ILR - Drop Period*.

Las métricas de este ensayo se pueden encontrar en las Figuras C.3 y C.4. En dichas métricas se puede observar un fenómeno similar al ocurrido en el anterior ensayo, es decir, en este análisis también se observa una homogeneidad de los resultados. Dicha homogeneidad sólo es perturbada brevemente por una combinación de valores que proporcionan peores resultados que la media y otra combinación que da mejores resultados. El par de valores que obtienen las mejores métricas son para un $ILR = 0{,}05$ y un $DP = 250$. Además, se puede observar que a partir de un $ILR \leqslant 0{,}01$ los resultados convergen aproximadamente y no existe gran variación de estos.

5.1.3.3. *Drop Factor - Drop Period*

Es el turno del análisis de los parámetros *Drop Factor* y *Drop Period*, Los valores escogidos para estos parámetros son los representados en la Tabla 5.11.

Las métricas de este ensayo son las de las Figuras C.5 y C.6. En este caso, es fácil ver que para la mayoría de los pares de valores escogidos se obtienen unos buenos resultados y, a su vez, existen ciertos pares de valores que tienen unas métricas mucho peores que el resto. Estos pares de valores son los que cumplen la condición $DF < 0{,}99$ y $DP = 50$. Esto tiene sentido, puesto que con un *Drop Period* tan bajo, si se tiene también un *Drop Factor* que no se aproxima a la unidad, se ocasiona que el ritmo de aprendizaje descienda muy rápido. Esto

Neuronas por capa	ILR	V.Freq
100	0.05	2000

(a) Hiperparámetros fijos.

DF	DP
0.99	50
0.95	100
0.9	250
0.85	500
0.8	1000

(b) Hiperparámetros variables.

Tabla 5.11: Hiperparámetros del análisis *Drop Factor - Drop Period*.

genera que la red neuronal, para un mismo número de épocas, no tenga tiempo suficiente para aprender el objetivo y, por lo tanto, produzca resultados propios de redes que sólo han aprendido durante las primeras épocas, es decir, redes subentrenadas.

5.1.3.4. Número de neuronas por capa - *ILR*

Se comienza ahora con los análisis del número de neuronas por capa más un parámetro adicional. Como primer parámetro adicional se tiene el ritmo de aprendizaje inicial. Cabe destacar que, puesto que se va a realizar el análisis del número de neuronas por capa y se tienen 2 capas, para evitar que este ensayo sea tridimensional, se han cogido las 5 mejores combinaciones de neuronas por capa obtenidas del análisis unidimensional y se han establecido esos pares para este ensayo. Esto es, se van a analizar 5 pares de número de neuronas por capa con otros 5 valores de *ILR*. Así pues, los valores escogidos para los hiperparámetros de este ensayo se pueden observar en la Tabla 5.12.

Los resultados de este análisis se pueden ver en las Figuras C.7 y C.8. En ellas se puede observar que todas las combinatorias en las que se tiene un $ILR = 0{,}1$ son las que peores resultados presentan. Fuera de esas combinatorias de valores, los resultados son similares entre sí aproximadamente. Por otro lado, los hiperparámetros que mejores resultados presentan son para $Neuronas = 100 - 25$ e $ILR = 0{,}025$.

5.1.3.5. Número de neuronas por capa - *DF*

Continuando con los análisis del número de neuronas por capa más otro parámetro, es el turno del *Drop Factor*. De forma análoga al ensayo anterior, el número de neuronas por capa

Drop Factor	Drop Period	V.Freq
0.8	1000	2000

(a) Hiperparámetros fijos.

Neuronas 1ªCapa - Neuronas 2ªCapa	ILR
50 - 25	0.1
100 - 25	0.05
50 - 50	0.025
100 - 50	0.01
100 - 75	0.005

(b) Hiperparámetros variables.

Tabla 5.12: Hiperparámetros del análisis del número de neuronas por capa - *ILR*.

se organizará por pares para que así el análisis siga siendo bidimensional. Con todo ello, los valores de los hiperparámetros utilizados son los representados en la Tabla 5.13.

Drop Period	ILR	V.Freq
1000	0.025	2000

(a) Hiperparámetros fijos.

Neuronas 1ªCapa - Neuronas 2ªCapa	DF
50 - 25	0.99
100 - 25	0.95
50 - 50	0.9
100 - 50	0.85
100 - 75	0.8

(b) Hiperparámetros variables.

Tabla 5.13: Hiperparámetros del análisis del número de neuronas por capa - *Drop Factor*.

Las métricas correspondientes a estos análisis se pueden encontrar en las Figuras C.9 y C.10. En ellas se puede observar que, de manera generalizada, se obtiene unos resultados muy parecidos entre sí. El único caso en el que se pueden ver peores resultados es para el de $Neuronas = 50 - 50$. Por otro lado, las mejores métricas se observa que ocurren para el caso de $Neuronas = 100 - 25$ y $DF = 0{,}99$.

5.1.3.6. Número de neuronas por capa - *DP*

Finalmente, como último análisis bidimensional, se tiene el ensayo del número de neuronas por capa y el parámetro *Drop Period*. En el caso del parámetro del número de neuronas por capa, este se ha organizado de la misma manera que en los dos últimos análisis. En este ensayo se han utilizado los valores para los hiperparámetros mostrados en la Tabla 5.14.

ILR	**Drop Factor**	**V.Freq**
0.1	0.95	2000

(a) Hiperparámetros fijos.

Neuronas 1ªCapa - Neuronas 2ªCapa	**DP**
50 - 25	50
100 - 25	100
50 - 50	500
100 - 50	1000
100 - 75	2500

(b) Hiperparámetros variables.

Tabla 5.14: Hiperparámetros del análisis del número de neuronas por capa - *Drop Period*.

Esta combinatoria de hiperparámetros produce unas redes neuronales cuyas métricas se ven representadas en las Figuras C.11 y C.12. En ellas se puede observar un fenómeno similar al ocurrido en el ensayo anterior: salvo para los casos de $Neuronas = 100 - 25$ y $DP = 50$ y $Neuronas = 100 - 50$ y $DP = 100$, las peores métricas son las correspondientes a combinaciones en las que el número de neuronas de la primera capa es de 100. Por otro lado, los mejores resultados son los que ocurren para los casos de $Neuronas = 50 - 50$ y $DP = 500$ y $Neuronas = 50 - 25$ y $DP = 100$.

5.1.4. Análisis global (ND)

Como último gran grupo de análisis a realizar sobre las redes *MLP*, se tiene a continuación un análisis global de todos los hiperparámetros relevantes sobre este tipo de redes neuronales. Para ello, se ha utilizado el mismo *dataset* común explicado en la Sección 5.1.1, al igual que en los análisis unidimensionales y bidimensionales.

Los hiperparámetros escogidos para este análisis global son los siguientes: Número de neuronas por capa, *Drop Period*, *Drop Factor* e *Initial Learning Rate*. En concreto, para el hiperparámetro del número de neuronas por capa, al haber 2 capas en este caso y para reducir en 1 la dimensionalidad de este análisis, se procede a agrupar el número de neuronas por capa en pares al igual que se hizo en los análisis bidimensionales anteriores. Por ello, para este análisis se cuenta con un total de 4 hiperparámetros. Además, se han establecido 3 valores para cada uno de los hiperparámetros, por lo que se han realizado un total de **81 ensayos**. Estos ensayos son fruto de la combinatoria de los valores de los hiperparámetros mostrados en la Tabla 5.15.

Neuronas	**DP**	**DF**	**ILR**	**V.Freq**
50 - 25	50	0.99	0.1	2000
100 - 25	100	0.95	0.075	
50 - 50	250	0.9	0.05	

Tabla 5.15: Hiperparámetros del análisis global.

Debido al gran número de ensayos realizados y con el objetivo de mostrar las métricas lo más limpiamente posible, se han enumerado y recogido en la Tabla D.1 cada uno de los ensayos con su combinación de hiperparámetros particular. Por otro lado, las métricas se pueden consultar en las Figuras D, D.2, D.3, D.4 y D.5. En ellas se puede observar que los resultados obtenidos son, por lo general, similares entre sí, salvo por ciertos ensayos que han tenido unos resultados especialmente malos. Estos casos son principalmente los que tienen un $ILR = 0{,}1$. Además, a pesar de que para los ensayos 39 y 41 se obtienen localmente unos excelentes resultados, se puede ver que a grandes rasgos se obtienen peores métricas para los casos en los que se tienen un número de $Neuronas = 100 - 25$.

5.1.5. Conclusiones análisis redes *MLP*

Después de todos los análisis realizados se puede concluir que las redes *MLP* son capaces de obtener resultados con bastante precisión ($< 4\,\%$ de error en armónicos) para una amplia variedad de combinatoria de hiperparámetros.

En cuanto a los propios hiperparámetros y su impacto en las redes neuronales, se demuestra que el ritmo de aprendizaje en las primeras épocas de los entrenamientos tiene una gran

importancia en la precisión final de las redes. Por lo tanto, el conjunto de hiperparámetros que regulan este ritmo de aprendizaje, que son el *ILR*, *Drop Factor* y *Drop Period* deben estar acotados cada uno de ellos dentro de un rango de valores que garanticen que no se tiene un ritmo de aprendizaje demasiado elevado para las primeras épocas de los entrenamientos (ver Tabla 5.16). De lo contrario, la red aprendería demasiado rápido en las primeras épocas y ya no habría forma de evitar que la red fuera imprecisa. Cabe destacar que el extremo opuesto de establecer un ritmo de aprendizaje demasiado bajo para evitar esto último tampoco es deseable puesto que originaría unos tiempos de entrenamiento demasiado elevados. Esto es debido al hecho de que al aprender más lentamente, si se entrena para el mismo ritmo de épocas que en otros casos, es posible que no diera tiempo a que la red llegase al estado de entrenamiento óptimo y estuviera subentrenada, por lo que se tendría que aumentar el número de épocas de entrenamiento y, por lo tanto, el tiempo de computación.

Hiperparámetro	**Rango de valores aproximado**
ILR	$[0{,}01, 0{,}075]$
Drop Factor	$[0{,}9, 0{,}99]$
Drop Period	$[50, 250]$

Tabla 5.16: Rango de valores aproximado para los hiperparámetros encargados del ritmo de aprendizaje.

En el caso del parámetro del número de neuronas por capa, para los tamaños de los datos de entrada y salida que se han utilizado para estos análisis, se puede concluir que no es necesario establecer un número de neuronas por capa demasiado alto. En concreto, de los mejores resultados que se han obtenido han sido para los casos en los que se tiene 50 neuronas en la primera capa y entre 25 y 50 en la segunda. Por ello, para este caso se puede concluir que el número de neuronas de la primera capa debe ser mayor o igual al número de neuronas de la segunda capa.

5.2. Análisis redes *CNN*

Una vez realizados los análisis paramétricos más exhaustivos de las redes neuronales *MLP*, es el turno de los análisis sobre las redes *CNN*, en concreto, sobre los parámetros que

son exclusivos de este tipo de redes. Estos parámetros son, tal y como se vio en la Sección 4.3.2, el tamaño de los filtros y el número de estos de las capas convolucionales, es decir, los parámetros *Filter Size* y *Num Filters* respectivamente. Estos análisis serán unidimensionales y ambos se realizarán bajo la misma estructura regresiva de la red *CNN*. Esta parte regresiva de las *CNN* se ha establecido con los valores de los parámetros mostrados en la Tabla 5.17. Se han escogido estos valores en concreto porque, tal y como se concluyó en la Sección 5.1.5, se conoce que proporcionan unos buenos resultados para las redes *MLP*.

Neuronas 1ªCapa	**Neuronas 2ªCapa**	**DP**	**DF**	**ILR**	**V.Freq**
50	25	150	0.95	0.01	5000

Tabla 5.17: Hiperparámetros de la parte regresiva de las *CNN*.

Además, cabe destacar que se ha utilizado un *dataset* distinto para estos análisis. Se ha usado un *dataset* con el mismo número de asteroides (20000) y con el mismo orden de armónico máximo ($n_{max} = 13$) que en el *dataset* utilizado en los análisis de las *MLP*, pero se ha usado el generador de asteroides que los crea más esféricos, tal y como el asteroide de la Figura 4.4. Esto se ha hecho así debido al hecho de que, tal y como se explicó en la Sección 5.1.1, la mayoría de los asteroides tienden a tener este tipo de formas con menos protuberancias. Además, cabe destacar que lo que se va a realizar a continuación es un análisis paramétrico totalmente independiente a los realizados a las redes *MLP*. Posteriormente, para realizar la comparativa entre ambos tipos de redes neuronales se utilizará un mismo *dataset* común.

5.2.1. *Num Filters*

El primer análisis parámetrico de las redes *CNN* a realizar es del hiperparámetro *Num Filters*. Este parámetro, tal y como se introdujo en la Sección 4.3.2, se encarga de establecer el número de filtros que se utilizan en la capa convolucional 2D. Cabe destacar que, debido al hecho de que se tienen 2 capas convolucionales en las redes *CNN* del proyecto, se podrían establecer 2 números de filtros distintos. Sin embargo, para no aumentar demasiado la complejidad del proyecto, se va a utilizar el mismo valor de *Num Filters* para ambas capas

convolucionales 2D.

Así pues, los valores de los hiperparámetros utilizados en este análisis son los mostrados en la Tabla 5.18. En ella se puede observar que el número de épocas utilizadas es bastante menor a las usadas en los análisis de las redes *MLP*. Además de esto, el *Drop Period* utilizado también es menor al usado en las redes *MLP*. Esto es debido al hecho de que las iteraciones realizadas por cada época en esta red *CNN* es mayor a las de las redes *MLP* del proyecto. Es por ello que se establece un número menor de épocas y de *Drop Period*, porque con estos valores se llega aproximadamente en el final del entrenamiento a un ritmo de aprendizaje y tiempo de computación similares a los obtenidos en las redes *MLP*.

Ensayo	Num Filters	Filter Size	ILR	DP	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	DF	V.Freq	Epochs
1	1	3	0.01	25	50	25	0.95	5000	750
2	2	3	0.01	25	50	25	0.95	5000	750
3	5	3	0.01	25	50	25	0.95	5000	750
4	10	3	0.01	25	50	25	0.95	5000	750
5	20	3	0.01	25	50	25	0.95	5000	750
6	30	3	0.01	25	50	25	0.95	5000	750
7	40	3	0.01	25	50	25	0.95	5000	750
8	50	3	0.01	25	50	25	0.95	5000	750

Tabla 5.18: Hiperparámetros del análisis *Num Filters*.

Una vez realizados los ensayos, las métricas resultantes de ellos se pueden consultar en las Figuras E.1 y E.2. En ellas se puede observar que, a grandes rasgos, los mejores resultados son los obtenidos para un número de filtros de 1 y de 20. A pesar de que no tengan buenas métricas de error absoluto global, sí que son buenas las métricas por harmónicos, las porcentuales y el *accuracy*. De forma más precisa, el ensayo de 1 sólo filtro es el que presenta mejores resultados. Esto puede ser debido a que el tamaño del *input* no es lo suficientemente grande para necesitar de más filtros. Por ello al aumentar el número de filtros no mejoran los resultados, porque se introduce ruido. Así pues, para futuros análisis se utilizará *Num Filters* = 1 como hiperparámetro.

5.2.2. *Filter Size*

Como segundo y último parámetro a estudiar de las redes *CNN* se tiene el *Filter Size*. Este parámetro se trata del tamaño del filtro utilizado en la capa convolucional, tal y como

se explicón en la Sección 4.3.2. En este caso ocurre lo mismo que en el análisis anterior del parámetro *Num Filters*, y es que, debido al hecho de que se tienen 2 capas convolucionales se pueden establecer 2 tamaños de filtros distintos, uno para cada una de ellas. Sin embargo, se vuelve a proceder de la misma manera: se utilizará el mismo tamaño de filtro para ambas capas con el objetivo de reducir la complejidad de los análisis.

Ensayo	Filter Size	Num Filters	ILR	DP	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	DF	V.Freq	Epochs
1	1	1	0.01	25	50	25	0.95	5000	750
2	2	1	0.01	25	50	25	0.95	5000	750
3	3	1	0.01	25	50	25	0.95	5000	750
4	4	1	0.01	25	50	25	0.95	5000	750
5	5	1	0.01	25	50	25	0.95	5000	750
6	6	1	0.01	25	50	25	0.95	5000	750
7	7	1	0.01	25	50	25	0.95	5000	750
8	8	1	0.01	25	50	25	0.95	5000	750
9	9	1	0.01	25	50	25	0.95	5000	750
10	10	1	0.01	25	50	25	0.95	5000	750

Tabla 5.19: Hiperparámetros del análisis *Filter Size*.

En la Tabla 5.19 se pueden observar los valores escogidos para los hiperparámetros del análisis. De nuevo, se han usado unos valores de *Drop Period* y *Epochs* iguales a los del análisis de *Num Filters* por el mismo motivo: conseguir un ritmo de aprendizaje y tiempo de computación similar a los obtenidos en los análisis de las redes *MLP*. Además, cabe destacar que los valores de *Filter Size* puestos, al ser un sólo número entero, indican que los tamaños de los filtros son cuadrados de lado igual a dichos valores.

Las métricas de los ensayos realizados se pueden consultar en las Figuras E.3 y E.4. En ellas se puede observar que ha ocurrido un fenómeno similar al que ocurrió en el ensayo de *Num Filters* y es que los mejores resultados ocurren para el valor de la unidad del parámetro a ensayar. En este caso esto puede ser debido al hecho de que el objetivo de la red no requiere de la extracción de características complejas del *input* y, por lo tanto, la red no necesitar de filtros más grandes para procesar la información. Además, esto también podría deberse al hecho de que el tamaño del *input* a la red (10x10) no es lo suficientemente grande como para necesitar de filtros más grandes. Estos filtros entonces pueden introducir ruido o información innecesaria y por ello presentan peores resultados.

5.2.3. Conclusiones análisis redes *CNN*

Después de los análisis realizados a los parámetros exclusivos seleccionados de las redes *CNN* se puede determinar que, a grandes rasgos, este tipo de estructuras pueden generar redes cuyas métricas presentan una alta precisión ($< 2\,\%$ de error absoluto global).

En cuanto a los parámetros estudiados en concreto, tal y como se ha ido adelantando en sus respectivos apartados, los resultados que presentaban mejores métricas han sido los cuales minimizaban el impacto de la convolución en la red. Esto es, los mejores resultados ocurrían para una unidad de filtro por capa convolucional y para un tamaño de filtro de 1x1. Una de las causas de que esto haya sido así puede que haya sido que el tamaño de entrada de las redes es demasiado pequeño como para necesitar de filtros más grandes para así poder extraer más características complejas de las 'imágenes'. Se recuerda que las dimensiones de entrada de las redes viene determinado por el tamaño de la malla utilizada para definir los asteroides. Es por ello que, si se tuviera un tamaño de malla mayor y, por lo tanto, unas dimensiones de entrada a la red mayores, no se podría asegurar que los valores de los parámetros *Num Filters* y *Filter Size* que presentaban mejores resultados para los análisis realizados vayan a ser también los mejores para estos casos.

5.3. Comparativa redes *MLP vs CNN*

La parte final de la exposición de los resultados de este proyecto consta de una última comparativa entre las redes con una estructura *MLP* y las redes *CNN*. Esta comparativa se va a realizar con el objetivo de descubrir los rendimientos y límites de ambos tipos de redes bajo un mismo *dataset* común. Este nuevo *dataset* común constará del mismo número de asteroides (20000) que los *dataset* utilizados para los análisis independientes. Sin embargo, estos asteroides tendrán una mayor definición de malla superficial y un mayor grado y orden de los armónicos empleados. Se recuerda que los parámetros que definen estas características son tanto $grid(\lambda)$ y $grid(\phi)$ como n_{max}, los cuales fueron introducidos en la Sección 4.1. En la Tabla 5.20 se pueden observar los valores escogidos para dichos parámetros que conforman el *dataset*.

Num Asteroids	n_{max}	$grid(\lambda)$	$grid(\phi)$
20000	30	30	30

Tabla 5.20: Características del *dataset* de la comparativa entre redes *MLP* y *CNN*.

Se han establecido estos valores para la generación del *dataset* debido a que se ha considerado que producen unos asteroides lo suficientemente detallados sin llegar a comprometer demasiado los tiempos de computación necesarios tanto para el manejo de los datos como para los entrenamientos de las redes.

Además, se concreta que se ha utilizado el mismo generador de asteroides usado en la creación del *dataset* que se empleó en el análisis de las redes *CNN*. Esto se ha hecho así debido al hecho de que con esta variación del generador se obtienen unos asteroides más realistas para unos grados y órdenes de armónicos más altos. En la Figura 5.2 se puede ver un ejemplo del tipo de asteroides que van a formar parte de este *dataset* y su variación según el n_{max} empleado.

Una vez se tiene generado el *dataset* con el que van a ser entrenadas y comparadas las redes *MLP* y *CNN*, es el turno de detallar los entrenamientos.

Primero, tal y como se vio en las conclusiones del análisis de las redes *CNN* en la Sección 5.2.3, las deducciones que se hicieron de los parámetros exclusivos de estas redes estaban condicionadas por el pequeño tamaño de entrada a las redes usado. Por ello, no se pueden utilizar los mismos valores de dichos parámetros que producían buenos resultados con el nuevo *dataset*.

En cuanto a las redes *MLP*, a pesar de que *a priori* se podría pensar en utilizar los mismos hiperparámetros que en los ensayos anteriores producían buenos resultados, ya que estas redes no son tan dependientes del tamaño de entrada como las redes *CNN*, sigue existiendo cierta no linealidad en los entrenamientos y por lo tanto tampoco se deben usar los mismos hiperparámetros con este *dataset*.

Por ello, se ha optado por hacer un breve análisis exploratorio de ambas redes neuronales con el objetivo de encontrar unas combinatorias de hiperparámetros que produzcan buenos resultados con el nuevo *dataset*.

Se ha decidido hacer esto debido a que el objetivo de esta comparativa es poder equiparar

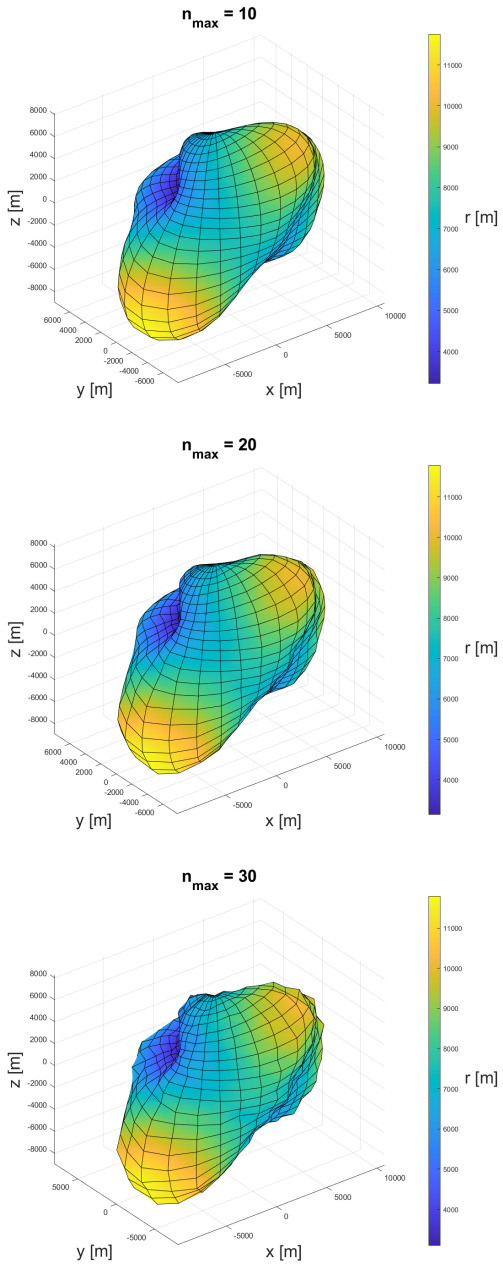

Figura 5.2: Comparativa del mismo asteroide con distinto n_{max}. Fuente: Elaboración propia.

unas redes que estén entrenadas de una forma lo más similar posible. Con esto se entiende que ambas redes lleguen a aproximadamente el mismo porcentaje de capacidad de entrenamiento de su óptimo.

Así pues, los hiperparámetros escogidos para los análisis exploratorios se pueden consultar en la Tabla F.1. Se debe aclarar que las columnas $N.F$ y $F.S$ se corresponden con los

hiperparámetros *Num Filters* y *Filter Size*, respectivamente. Además, se ha utilizado un hiperparámetro que hasta el momento no se había hecho uso de él, el denominado *Mini-Batch Size*, el cual se corresponde a la opción de entrenamiento `'MiniBatchSize'` y tiene un valor por defecto de 128. Este hiperparámetro establece el tamaño del *mini-batch* utilizado en cada iteración del entrenamiento. Un *mini-batch* no es más que una subdivisión de los datos de entrenamiento que es usada para evaluar el gradiente de la función de pérdida y actualizar los factores de peso de las redes en cada iteración del entrenamiento.

Las métricas utilizadas para evaluar los rendimientos de las redes neuronales son las mismas que se han usado en el resto de análisis del proyecto y estas se pueden observar en las Figuras F.1 y F.2.

En estas, es fácil ver que en la mayoría de los ensayos realizados se obtienen unas métricas con unos errores más elevados que los que se han visto anteriormente en el proyecto. Esto es debido a que se ha aumentado la definición y complejidad de los asteroides utilizados en el nuevo *dataset*, lo cual tiene como consecuencia un aumento de los tamaños tanto de entrada como de salida de las redes neuronales. Esto, además del impacto directo sobre las redes *CNN* debido a los filtros que usan, origina un aumento de datos a tratar por parte de las redes, lo que provoca un aumento en los tiempos de entrenamiento de estas que, si no se aumentan las épocas de entrenamiento, produce peores resultados. En la Tabla F.1 se puede ver que se han aumentado ligeramente las épocas para intentar paliar esto. Sin embargo, debido al mayor tamaño de las entradas y salidas de las redes, estas épocas conllevan más tiempo de computación. Es por esto por lo que se ha tomado la decisión de no aumentar demasiado las épocas para evitar un coste temporal muy elevado, en detrimento de unos resultados ligeramente más precisos.

A pesar de todo lo anterior, se ha conseguido una combinación de hiperparámetros para cada una de las redes neuronales que produce unas métricas que se acercan a las vistas en anteriores ensayos del proyecto. Estos han sido los ensayos 'CNN 11' y 'MLP 4'. Entre ellos, en sus métricas se puede ver que la red convolucional es capaz de reducir el error en los coeficientes S_{nm} con respecto a la red *MLP*, sin embargo, aumenta el error de los coeficientes C_{nm}. Además, en la Tabla F.1, se puede observar que el ensayo 'CNN 11' usa 3

filtros convolucionales de un tamaño de 5x5. Esto confirma la hipótesis de la conclusión que se hizo en la Sección 5.2.3 de que para que las redes *CNN* puedan ser aprovechadas, debe haber cierto tamaño mínimo de entrada a la red para que los filtros sean relevantes. Por otro lado, en las Figuras 5.3 y 5.4 se puede observar una comparativa de un asteroide aleatorio y la predicción de este por las redes *CNN* y *MLP*, respectivamente.

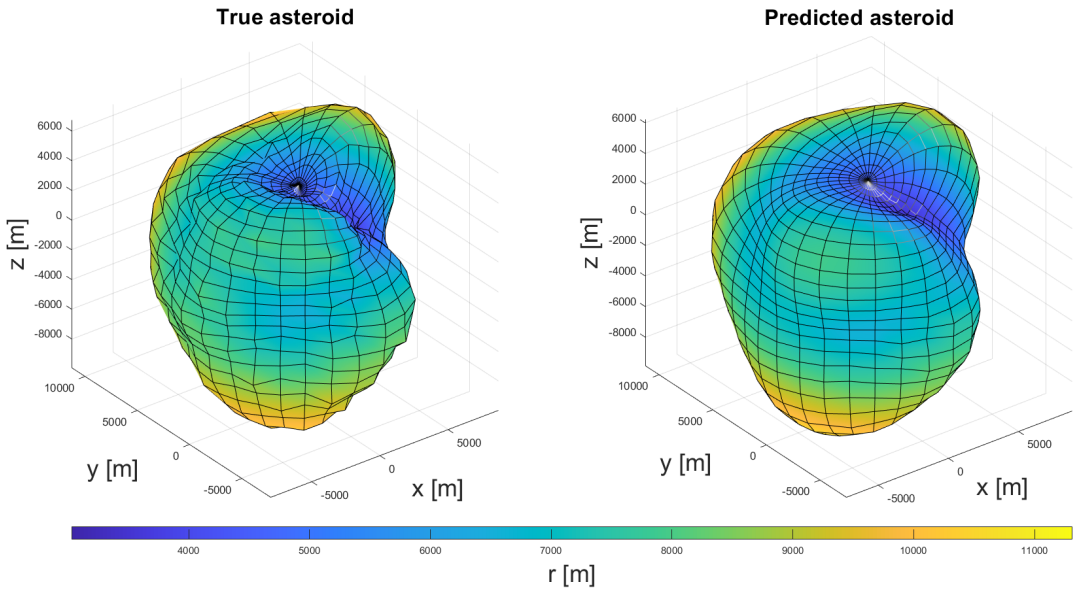

Figura 5.3: Asteroide real frente al predicho por la red del ensayo 'CNN 11'.

En estas gráficas se puede ver que, a pesar de haber conseguido unos ensayos con unas métricas del mismo orden de magnitud, aunque mayores que las de los ensayos anteriores en el proyecto, ambas redes son capaces de aproximar la geometría general con relativa precisión pero no consiguen representar las pequeñas irregularidades de la superficie de una manera fiel. Dicho de otra manera, predicen los asteroides con una superficie más lisa y regular que la real.

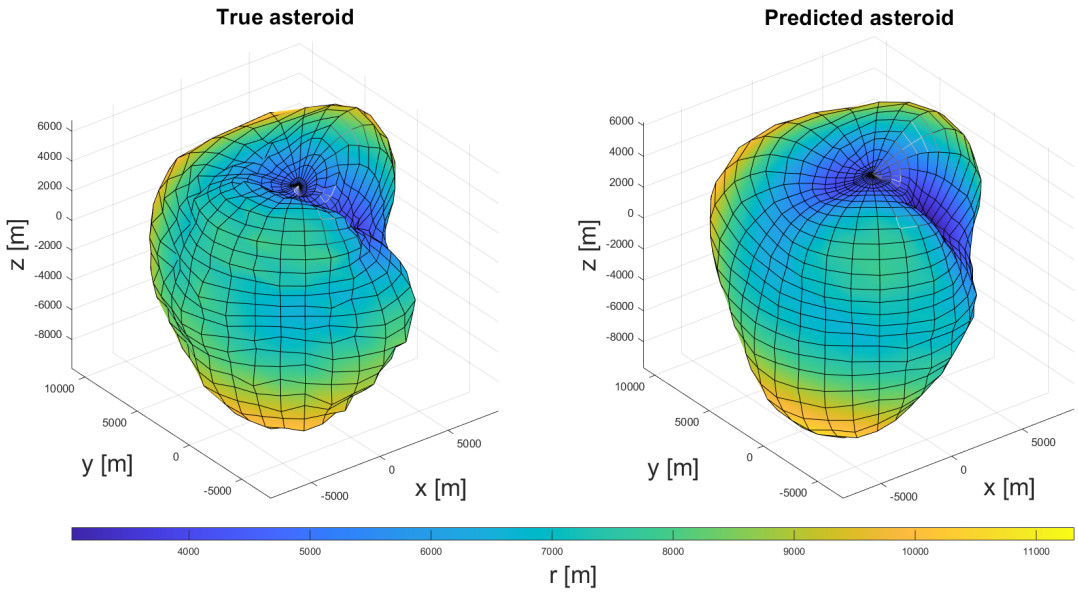

Figura 5.4: Asteroide real frente al predicho por la red del ensayo ‘MLP 4’.

Capítulo 6

Conclusiones y futuros trabajos

Finalmente, en este último capítulo se van a recoger las conclusiones que han sido extraídas del proyecto realizado y, después, se marcarán unas pautas de posibles caminos de mejora de futuros trabajos en los que profundizar sobre la materia vista.

6.1. Consecución de objetivos

A lo largo de este Trabajo de Fin de Grado se ha tratado de averiguar la viabilidad de crear y entrenar a unas redes neuronales con el objetivo de expandir en armónicos esféricos la geometría de un asteroide. Para ello, primero se hizo una introducción al *Machine Learning* y se escogieron las estructuras de redes neuronales que se iban a crear según sus características y la compatibilidad con la funcionalidad de éstas. Las estructuras escogidas fueron las redes *Multi-Layer Perceptron (MLP)* y *Convolutional Neural Network (CNN)*. Después, se generaron multitud de asteroides artificiales, tomando como referencia el asteroide Eros 433, con los que entrenar a las redes neuronales. Estos asteroides constaban de la componente radial de su geometría en coordenadas esféricas y de los coeficientes de los armónicos esféricos. A continuación, se entrenaron las redes neuronales escogidas con dichos asteroides y se evaluaron los rendimientos de éstas con las métricas establecidas. Seguidamente, se hicieron varios estudios hiperparamétricos de las redes neuronales con los objetivos de extraer posibles relaciones entre dichos hiperparámetros y de encontrar combinatorias de estos que produjeran buenos resultados. Finalmente, se ha realizado una última comparativa entre las redes *MLP* y *CNN* con un *dataset* de asteroides con mayor tamaño de malla y mayor número armónicos

esféricos. Además de la propia comparativa para ver qué tipo de red proporcionaba mejores resultados bajo un mismo *dataset*, esto se hizo para ver la respuesta que podían llegar a tener las redes frente a un mayor tamaño de entrada y salida, es decir, la respuesta a un problema más complejo.

Respecto a los **objetivos específicos** propuestos, el proyecto ha concluido con la consecución de todos ellos. Como primer objetivo específico se propuso la selección de los tipos de estructura de red neuronal que mejor fueran a desempeñar la función propuesta para ellas. Esto es, escoger los tipos de redes neuronales a desarrollar teniendo en cuenta que se desea obtener una red neuronal capaz de tomar como entrada la componente radial en coordenadas esféricas de la geometría de un asteroide y devolver los coeficientes fruto de la expansión en armónicos esféricos de dicha geometría. Para ello, en la Sección 3.3.1 se hizo un listado de los tipos de redes neuronales más comunes y además se definieron la funcionalidad y uso de estas. Finalmente, en la Sección 3.3.2, se concluyó que los mejores tipos de redes neuronales a desarrollar en este proyecto serían las *Multi-Layer Perceptron (MLP)* y *Convolutional Neural Network (CNN).*

El siguiente objetivo específico que se propuso fue la creación de una base de datos o *dataset* que permitiera el entrenamiento de las redes neuronales construidas para conseguir cumplir el propósito de estas. Además de esto, también se debía generar un código capaz de evaluar estos entrenamientos para poder cuantificar y representar la precisión de las redes.

Así pues, con respecto a los *dataset* desarrollados, se distinguen un total de 3 a lo largo de todo el proyecto. El primer *dataset* creado constaba de 20000 asteroides con un grado y orden máximo de representación de hasta el armónico esférico 13, es decir, constaba de un $n_{max} = 13$. Por otro lado, la malla que definía la geometría de estos asteroides tenía 10 divisiones sobre la longitud y latitud, lo cual estaba representado por las variables $grid(\lambda) = grid(\phi) = 10$. Además, este *dataset* fue creado con un generador de asteroides que los produce con unas formas con más protuberancias y exageradas que la mayoría de asteroides reales. Un ejemplo de este tipo de asteroides se puede ver en la Figura 5.1. Este *dataset* se usó en los análisis hiperparamétricos de las redes *MLP* en la Sección 5.1.

El segundo *dataset* creado posee las mismas características que el primero, es decir, consta

de 20000 asteroides con un $n_{max} = 13$ y $grid(\lambda) = grid(\phi) = 10$. Sin embargo, este nuevo *dataset* posee unos asteroides producidos por un generador distinto al utilizado en el primer *dataset*. Este nuevo generador es más estable para grados y órdenes mayores de armónicos y, además, produce unos asteroides más 'realistas', es decir, con formas más esféricas y sin tantas protuberancias. Un ejemplo de los tipos de asteroides que componen este *dataset* se puede ver en la Figura 4.4, así como en la Figura 5.2 se puede observar cómo varía la producción de este generador en función del n_{max}. Este *dataset* fue el que se utilizó para el análisis hiperparamétrico de las redes *CNN* en la Sección 5.2.

El tercer *dataset* creado utiliza el mismo generador de asteroides que en el anterior caso. Además, también se han creado el mismo número de asteroides que en los otros 2 *dataset*, es decir, 20000 asteroides. Sin embargo, estos asteroides poseen un mayor nivel de definición de superficie y de rugosidad de esta. Esto es, se ha incrementado el grado y orden máximo de armónicos esféricos a $n_{max} = 30$ y también se ha aumentado el tamaño de la malla a $grid(\lambda) = grid(\phi) = 30$. Este tipo de asteroides se puede observar en la Figura 5.3, concretamente en el asteroide de la izquierda, ya que representa el real. Este *dataset* fue el utilizado en la comparativa final entre las redes *MLP* y *CNN* en la Sección 5.3.

El tercer objetivo específico impuesto fue la creación de estudios hiperparamétricos de las redes neuronales seleccionadas con la finalidad de buscar el impacto de estos hiperparámetros en los rendimientos de las redes. El primer estudio hiperparamétrico fue sobre las redes *MLP* y comenzó en la Sección 5.1. Aquí, de primera mano se hizo un preanálisis (Sección 5.1.1) para la selección de los parámetros que iban a definir el primer *dataset* generado. Para ello, se fijaron todos los hiperparámetros encargados del entrenamiento de las redes (Tabla 5.1) y se escogieron, en la Tabla 5.2, una combinatoria de los parámetros encargados de la generación de los asteroides. De este modo, se estableció que el primer *dataset* generado, el que iba a ser usado en los análisis de las redes *MLP*, iba a contar con 20000 asteroides con un $n_{max} = 13$ y $grid(\lambda) = grid(\phi) = 10$.

Una vez realizado esto, se comenzó con los propios análisis hiperparamétricos de las redes *MLP*. Primero, en la Sección 5.1.2, se realizó un estudio hiperparamétrico unidimensional de las redes *MLP*. Esto es, siendo N el número total de hiperparámetros, se fijaron $N - 1$

hiperparámetros y se varió el restante. De esta manera, se podía estudiar su impacto en los rendimientos de las redes. En el caso de las redes *MLP*, se contaba con $N = 5$ hiperparámetros: *Validation Frequency, Initial Learning Rate, Drop Period, Drop Factor* y número de neuronas por capa. De este análisis se concluyó que el hiperparámetro *Validation Frequency* no tenía un impacto lo suficientemente relevante en los rendimientos y fue eliminado de los siguientes estudios. Por otro lado, también se observó cómo los parámetros *Initial Learning Rate, Drop Period* y *Drop Factor*, todos encargados del control del ritmo de aprendizaje de la red, están muy relacionados entre sí y tienen un gran impacto en las métricas de las redes. Por último, también se descubrió que los mejores resultados ocurrían para los casos en los que el número de neuronas de la primera capa oculta de las redes era mayor o igual al número de neuronas de la segunda capa. Después, en la Sección 5.1.3, se continuó con un análisis bidimensional. En estos ensayos se fijaban $N - 2$ hiperparámetros y se iban variando los 2 parámetros restantes a pares. De esta manera, se intentó descubrir dependencias y relaciones entre pares de hiperparámetros. Además de esto, puesto que previamente se hizo el análisis unidimensional, se consiguió cercar más el rango de valores de los hiperparámetros para los cuales se obtenían buenas métricas. Finalmente, en la Sección 5.1.4, se hizo un análisis de los N hiperparámetros existentes donde cada uno de ellos contaba con 3 valores distintos. Esto produjo un total de 81 ensayos. En este análisis no se consiguió obtener unas conclusiones nuevas y distintas a las ya conocidas. Simplemente, se volvieron a reiterar las relaciones anteriormente explicadas y se consiguió terminar de definir el abanico de valores para los cuales se obtienen buenos rendimientos de las redes *MLP* para el *dataset* utilizado.

Una vez terminado el análisis hiperparamétrico de las redes *MLP*, se siguió con las redes *CNN*. Para el caso de las redes convolucionales se hizo uso del segundo *dataset* creado, el cual fue definido anteriormente. En este caso, sólo se hizo un análisis de los hiperparámetros exclusivos de las redes neuronales convolucionales puesto que, tal y como se explicó en la Sección 3.3.1, las capas de las redes *CNN* se pueden separar en una parte convolucional y en otra regresiva. Por ello y debido a que la parte regresiva se estructuró de la misma manera que las redes *MLP*, sólo se hizo un análisis de los parámetros *Num Filters* y *Filter Size*. De ambos análisis hiperparamétricos se concluyó que los mejores rendimientos de las redes neuronales ocurrían para los casos en los que se tenía 1 filtro y cuyo tamaño fuese

de 1x1. Estas conclusiones estaban muy condicionadas por el pequeño tamaño de entrada a las redes *CNN* usado, el cual era de 10x10. Esto es así ya que como se vio más adelante en la comparativa entre las redes *MLP* y *CNN*, al utilizar un *dataset* con mayor tamaño de entrada, los mejores resultados ocurrían para otros valores distintos de los parámetros de los filtros convolucionales.

Finalmente, el último objetivo específico del proyecto se desarrolló en la Sección 5.3. Allí se hizo una comparativa entre las redes *MLP* y *CNN* bajo un mismo *dataset* común. Este *dataset* fue el último que se creó y, tal y como se explicó con anterioridad, constaba de unos asteroides más complejos que los otros 2 *dataset* utilizados hasta el momento. Esto se hizo con el objetivo de ver cómo respondían las redes neuronales a mayores tamaños de entrada y salida. El problema que se tuvo con esta comparativa fue el hecho de que al tratarse de dos estructuras de red distintas y con distintos hiperparámetros, hubo cierta complejidad en alcanzar un grado similar de entrenamiento para ambas redes para que la comparación no careciera de sentido. Por ello, se hizo un análisis exploratorio hasta obtener unos rendimientos lo suficientemente buenos y parecidos entre sí. Finalmente, a pesar de que se obtuvieron unas redes comparables, estas sólo fueron capaces de predecir unos armónicos esféricos que reproducían la geometría general de los asteroides. Es decir, no conseguían representar fielmente las pequeñas irregularidades de la superficie. Esto fue debido a varios factores. Por un lado, la mayor definición y complejidad de los asteroides se traslada a unos mayores tamaños de entrada y salida de las redes. Por otro lado, esto mismo genera unos entrenamientos con mayor duración debido al aumento de datos que la red debe manejar. Debido a esto y a las limitaciones técnicas y temporales de este proyecto, se decidió que los resultados obtenidos eran lo suficientemente buenos. Sin embargo, existe un gran potencial de mejora para futuros proyectos en los que aumentar la precisión de las redes para este tipo de asteroides más complejos.

En cuanto al **objetivo general** del proyecto, este se estableció como el estudio de la viabilidad del uso de distintas redes neuronales para el desempeño de la tarea de convertir la geometría de asteroides en su expansión en armónicos esféricos. Esto es, proporcionar como entrada a las redes la componente radial de la geometría en coordenadas esféricas y que la red

devuelva como salida los coeficientes C_{nm} y S_{nm} fruto de la expansión en armónicos esféricos de dicha geometría. Esto, tal y como se ha ido viendo a lo largo del proyecto, en especial en el Capítulo 5, se podría afirmar que se ha conseguido, puesto que se han creado multitud de redes neuronales distintas que son capaces de desempeñar la tarea propuesta. Sin embargo, la pregunta que se debe hacer a continuación es, ¿cómo de viable ha sido? Si, por ejemplo, se toma como referencia los ensayos 'CNN 11' y 'MLP 4' de la comparativa final de la Sección 5.3, cuyas métricas se pueden observar en las Figuras F.1 y F.2, se consiguió obtener un error global porcentual en torno al 1 %, además de un *accuracy* de más del 0,95 y un error porcentual por armónicos siempre menor al 3 %. Bien es cierto que no son lo suficientemente precisas como para embarcarlas en una misión espacial actualmente. Sin embargo, se debe tener en cuenta las limitaciones existentes por el hecho de que este proyecto se trata de un trabajo de fin de grado. De hecho, a pesar de estas limitaciones, se podría decir que son métricas realmente buenas y con un gran potencial de mejora. Por ello, se concluye que ciertamente existe viabilidad en el uso de redes neuronales para la expansión de geometrías de asteroides en armónicos esféricos.

6.2. Trabajos futuros

En esta sección se van a resaltar algunas posibles direcciones de investigación y desarrollo que podrían ampliar los conceptos vistos en este proyecto y, además, se va a describir posibles aplicaciones prácticas del trabajo realizado.

- **Mejora de los asteroides de entrenamiento**

  Tal y como se introdujo en la Sección 4.3.1, las redes neuronales mejoran su precisión cuanto mayor es la relación entre el número de entradas y salida de las redes [40]. Por ello, dado que el tamaño de entrada de las redes neuronales creadas era definido por la Ecuación 4.2, se debería aumentar el número de pasos establecidos para la longitud y latitud que define la malla de puntos que mapea la geometría de los asteroides. Además de esto, se debe tener en cuenta que si también se desean asteroides con una mayor definición superficial, es decir, con un n_{max} superior, esto aumentará el tamaño de salida

de las redes. Por esto, se debería aumentar en consecuencia el tamaño de entrada si se quiere mantener la relación entre ambas tal y como se ha mencionado.

Por otro lado, también se podría aumentar el número de asteroides de los *dataset* generados. De hecho, esto sería necesario si se decide aumentar las otras propiedades mencionadas, puesto que las redes necesitarían de más datos de entrenamiento para poder aprender la tarea deseada.

La contrapartida de aumentar cualquiera de estas características de los asteroides de entrenamiento será, en primera instancia, un gran aumento de los tiempos de entrenamiento de las redes neuronales. Por otro lado, tal y como se vio con las redes *CNN*, estas redes son muy dependientes de los tamaños de entrada a las redes debido a los filtros convolucionales, por lo que se deberá prestar especial atención a estas redes.

- **Uso de distintos hiperparámetros**

  En los análisis hiperparamétricos realizados en el Capítulo 5, sólo se variaron los parámetros definidos en la arquitectura del código en la Sección 4.3. Estos hiperparámetros son sólo algunos de los existentes dentro de la función `trainingOptions` [3] de *MATLAB*. Por ello, se podría estudiar el impacto que tienen los hiperparámetros que no se han modificado y se han dejado por defecto. Además de esto, también se podrían utilizar unos optimizadores distintos a los usados y analizar si obtienen mejores resultados o no y bajo qué conjuntos de hiperparámetros ocurre.

- **Uso de distintas estructuras de redes neuronales**

  En la Sección 3.3.2 se eligió los tipos de redes neuronales que se iban a construir y en la Sección 4.3 se definieron las estructuras de dichas redes. Estas estructuras de redes profundas se pueden modificar de multitud de maneras distintas haciendo uso de las distintas capas disponibles [2], haciendo estas redes neuronales más o menos profundas y con las *hidden layers* que se desee.

  Además, también existe la posibilidad de entrenar redes neuronales sin hacer uso de la función integrada de `trainNetwork` de *MATLAB*. Esto se puede lograr mediante la

creación de bucles de entrenamiento personalizados [1]. De este modo se puede personalizar con mayor libertad la forma en la que se entrenan las redes neuronales. Por ello, un enfoque distinto a este proyecto que se podría desarrollar es la comparativa entre los resultados obtenidos con un bucle de entrenamiento personalizado y los obtenidos mediante las funciones integradas de *MATLAB*.

- **Uso de imágenes como entrada a las redes neuronales**

  En este proyecto se ha utilizado como entrada a las redes neuronales la componente radial en coordenadas esféricas de la geometría de los asteroides. En concreto, para el caso de las redes *CNN*, un posible trabajo futuro podría ser entrenar a las redes neuronales para que tengan como entrada imágenes de asteroides. Presumiblemente, se deberían de utilizar varias imágenes por asteroide para poder ver la totalidad de este. De esta manera, un satélite podría orbitar un asteroide o hacer una maniobra de *flyby* y obtener mediante una red neuronal los armónicos esféricos de dicho asteroide. Sin embargo, realizar este trabajo se presupone complejo ya que presenta varios conflictos que se deberían atajar. Primero se deberían de escalar todas las imágenes de los entrenamientos para que ocupasen el mismo tamaño de píxeles, además de que fueran tomadas todas desde los mismos ángulos. Después, otro inconveniente sería la rotación de los asteroides, ya que, según ésta, los coeficientes armónicos varían y la rotación no se puede conocer sólo con imágenes.

- **Uso de los armónicos obtenidos para el cálculo de trayectorias**

  Una posible ampliación de este trabajo sería realizar simulaciones de trayectorias de un satélite ficticio frente a diversas maniobras orbitales alrededor de un asteroide. Tal y como se vio en la Ecuación 2.25, mediante los armónicos esféricos y la componente radial de la geometría se puede definir el potencial gravitatorio del asteroide. Con este potencial gravitatorio y conociendo la cinemática del asteroide, se podrían simular trayectorias alrededor de éste y estudiar cómo de precisas han sido por el hecho de haber usado armónicos esféricos obtenidos mediante redes neuronales para el cálculo

[1] `https://www.mathworks.com/help/deeplearning/ug/train-network-using-custom-training-loop.html`

del potencial gravitatorio.

Apéndices

Apéndice A

Métricas del preanálisis

Ensayo	Grid (λ)	Grid (ϕ)	Nº Asteroides	Nº 1ªCapa	Nº 2ªCapa	DP	DF	ILR	V.Freq	Epochs
1	10	10	1000	100	100	150	0.95	0.01	100	8000
2	10	10	5000	100	100	150	0.95	0.01	100	8000
3	10	10	10000	100	100	150	0.95	0.01	100	8000
4	10	10	20000	100	100	150	0.95	0.01	100	8000
5	20	20	1000	100	100	150	0.95	0.01	100	8000
6	20	20	5000	100	100	150	0.95	0.01	100	8000
7	20	20	10000	100	100	150	0.95	0.01	100	8000
8	20	20	20000	100	100	150	0.95	0.01	100	8000

Tabla A.1: Ensayos del preanálisis.

Volver al informe

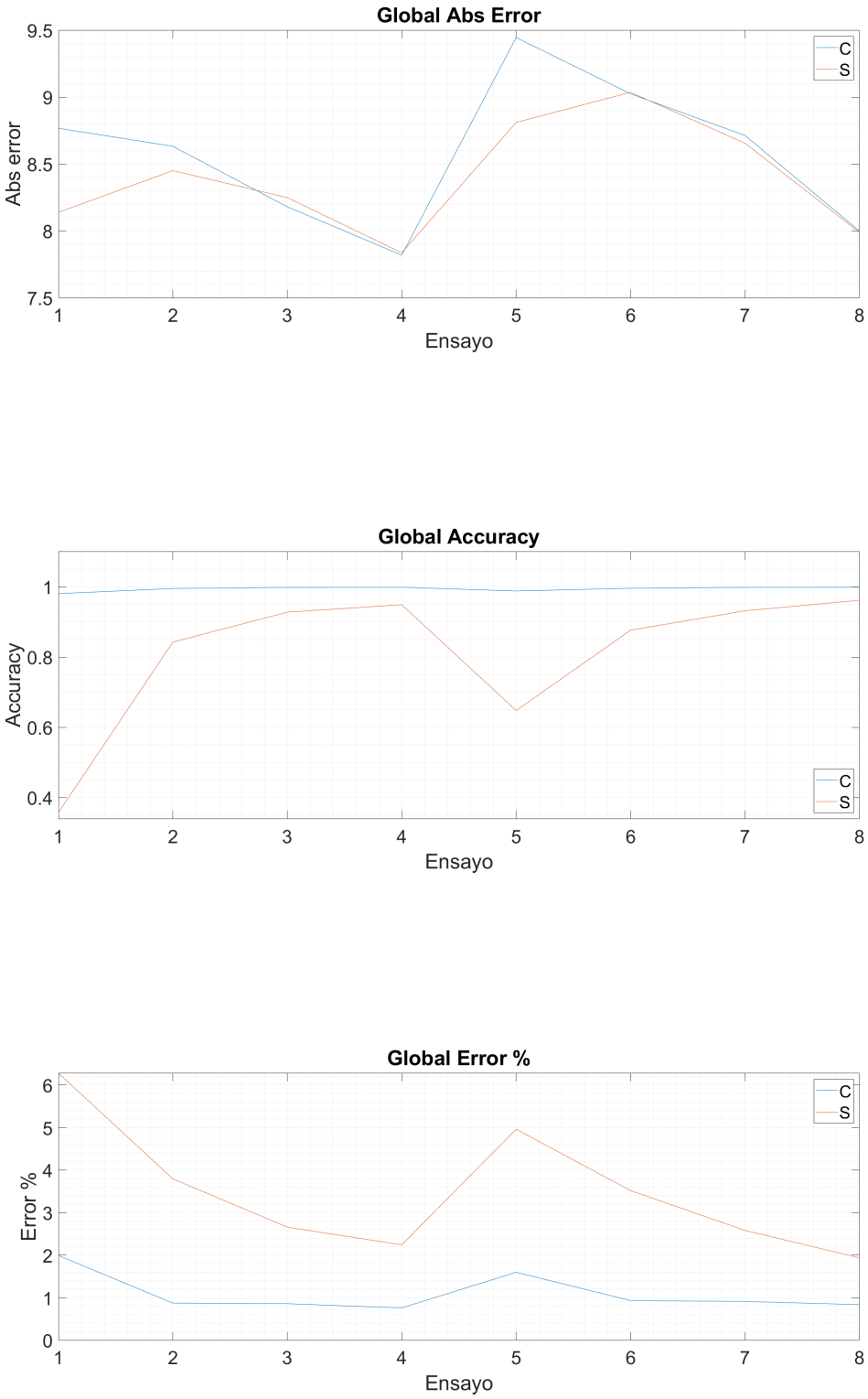

Figura A.1: Métricas numéricas globales del preanálisis.

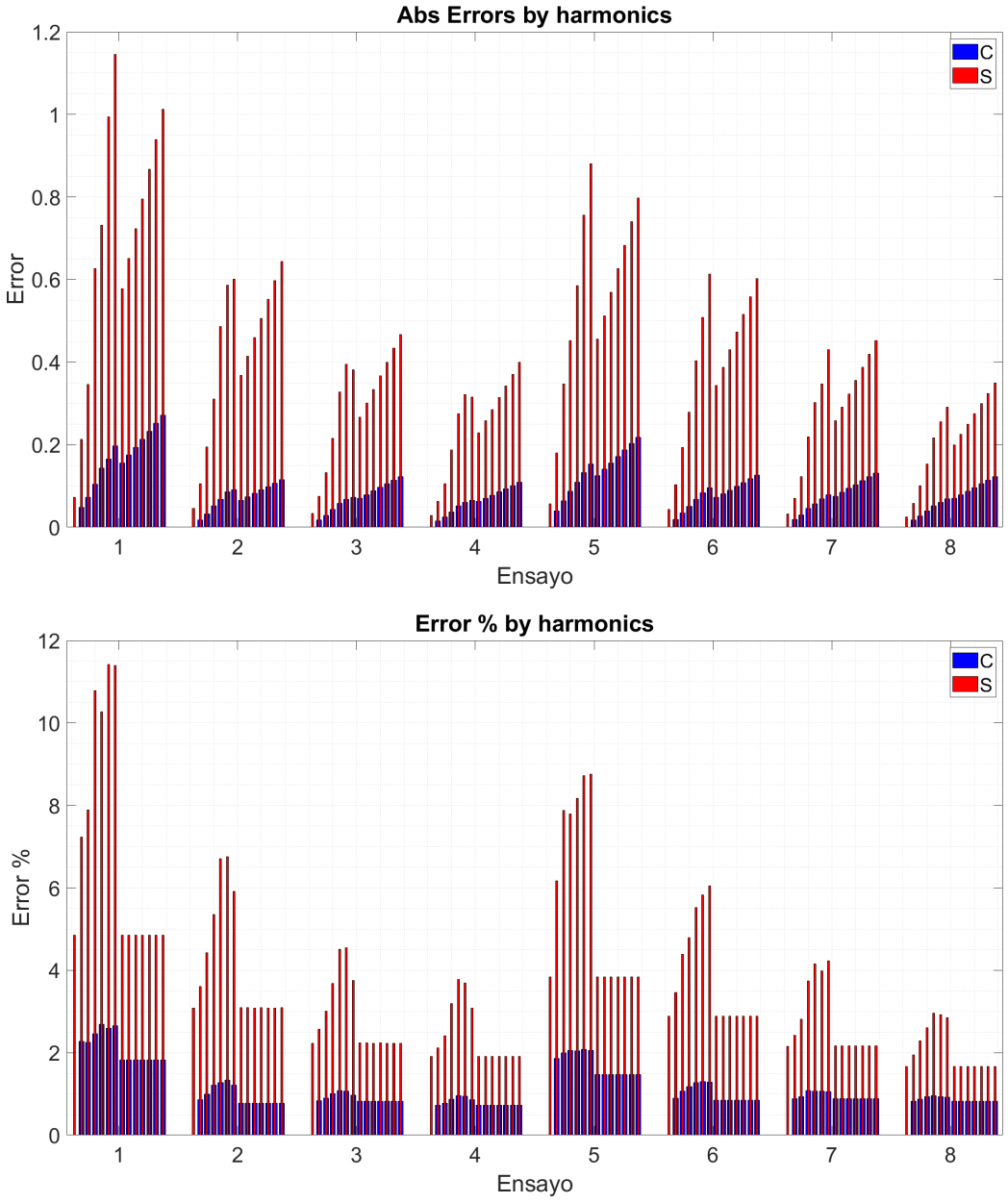

Figura A.2: Métricas numéricas por harmónicos del preanálisis.

Apéndice B

Métricas Análisis 1D redes MLP

B.1. Métricas 1D Validation Frequency

Volver al informe

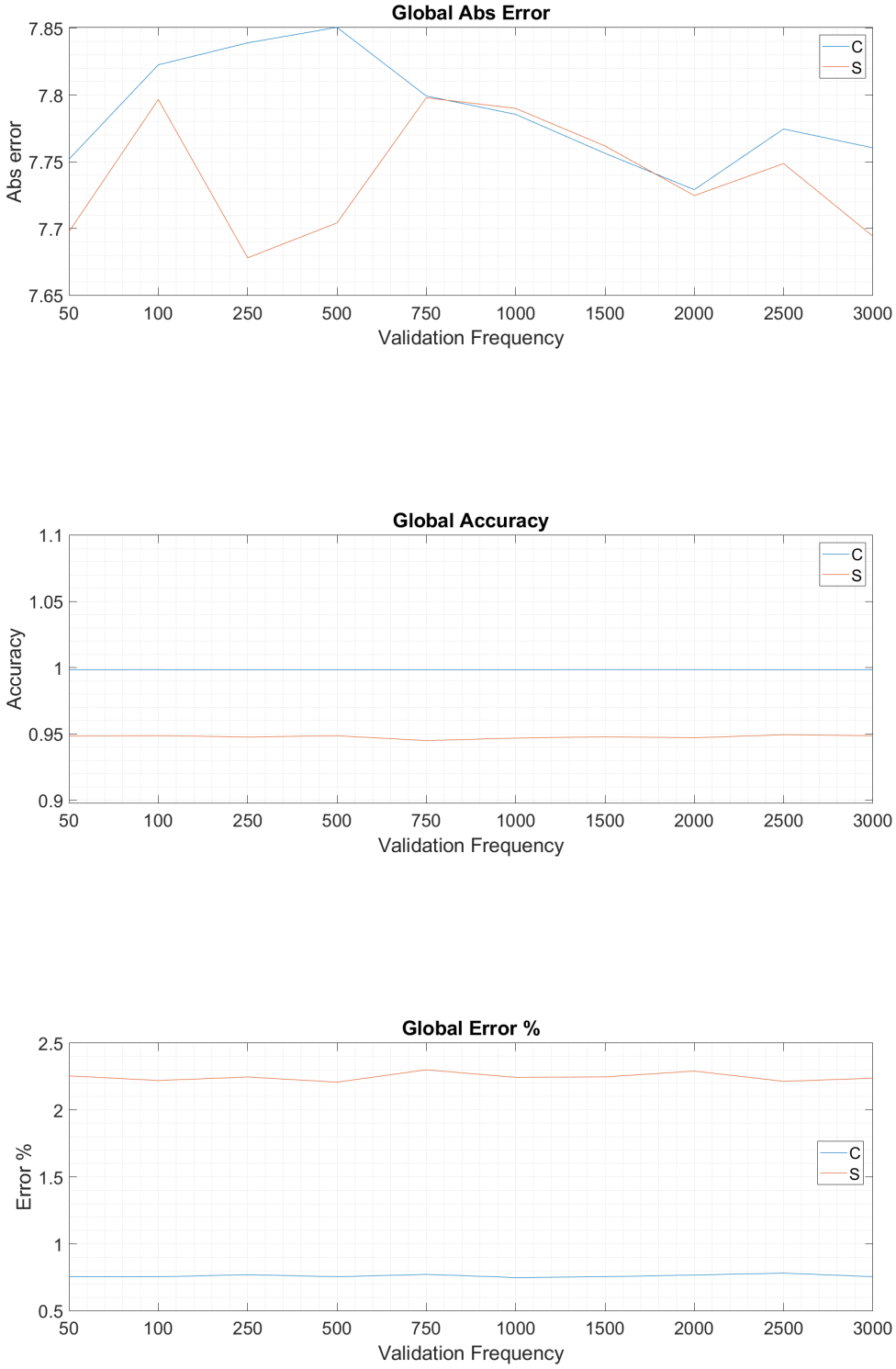

Figura B.1: Métricas numéricas globales del análisis 1D *Validation Frequency*.

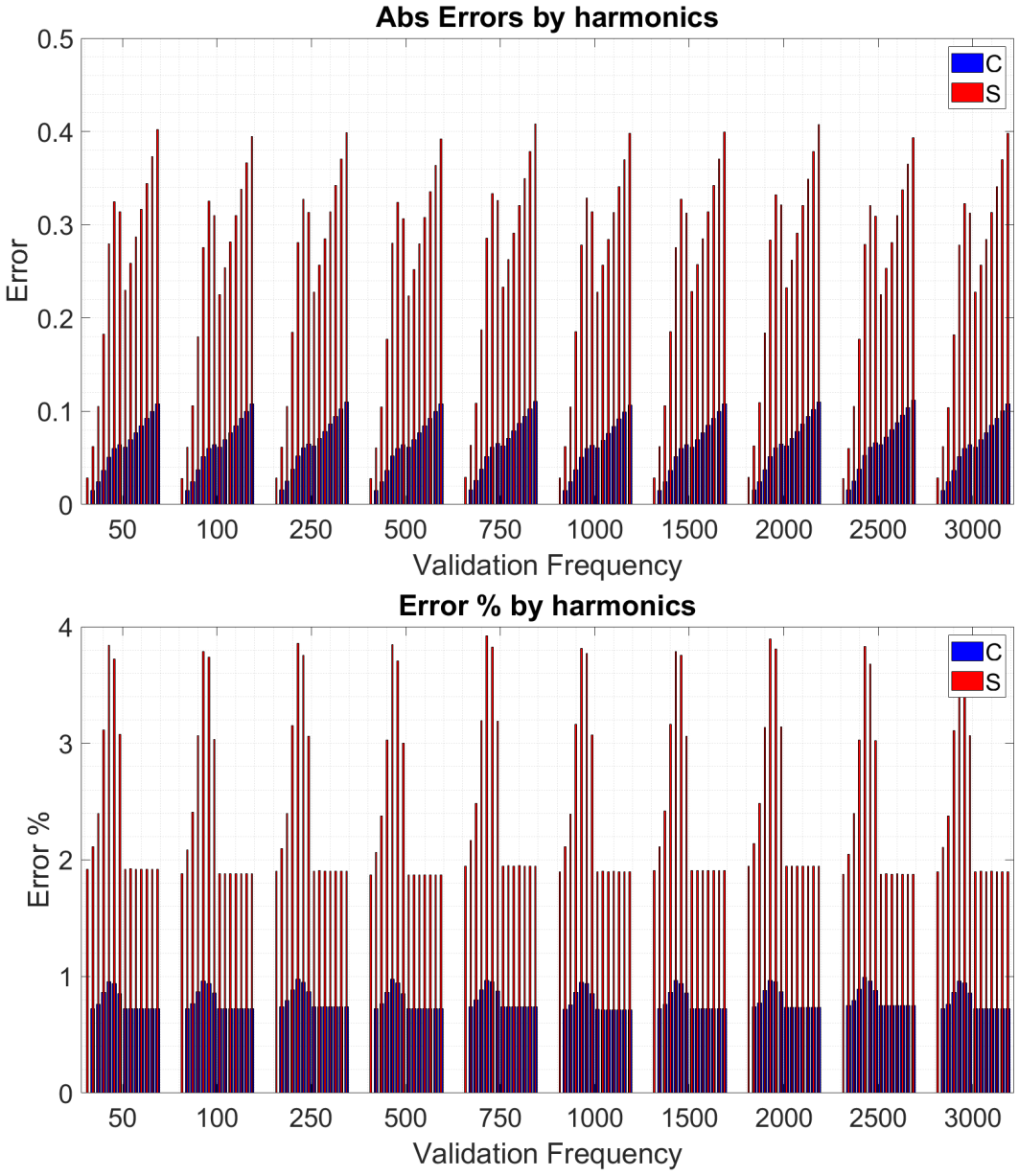

Figura B.2: Métricas numéricas por harmónicos del análisis 1D *Validation Frequency*.

B.2. Métricas 1D Initial Learning Rate

Volver al informe

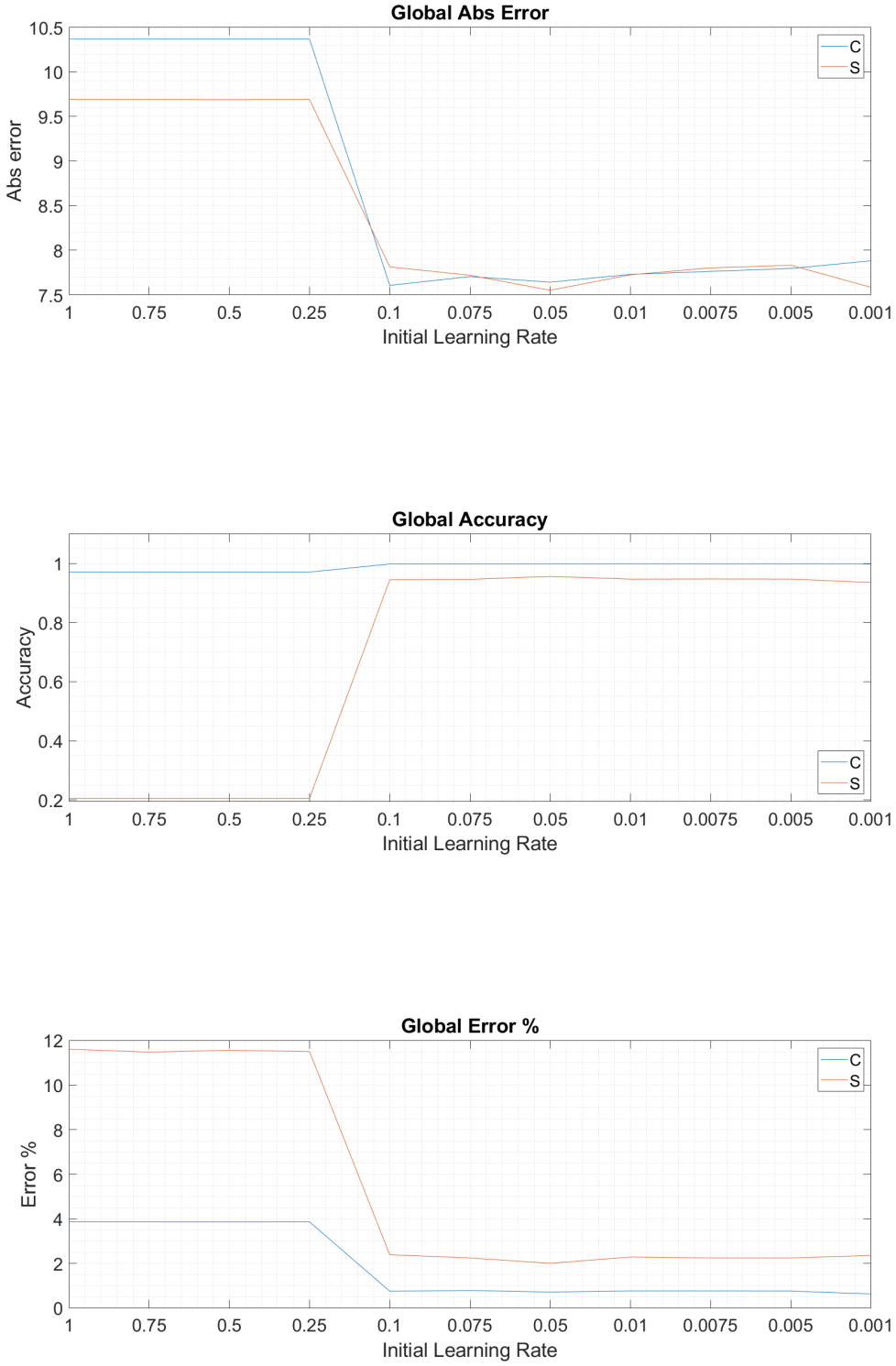

Figura B.3: Métricas numéricas globales del análisis 1D *Initial Learning Rate*.

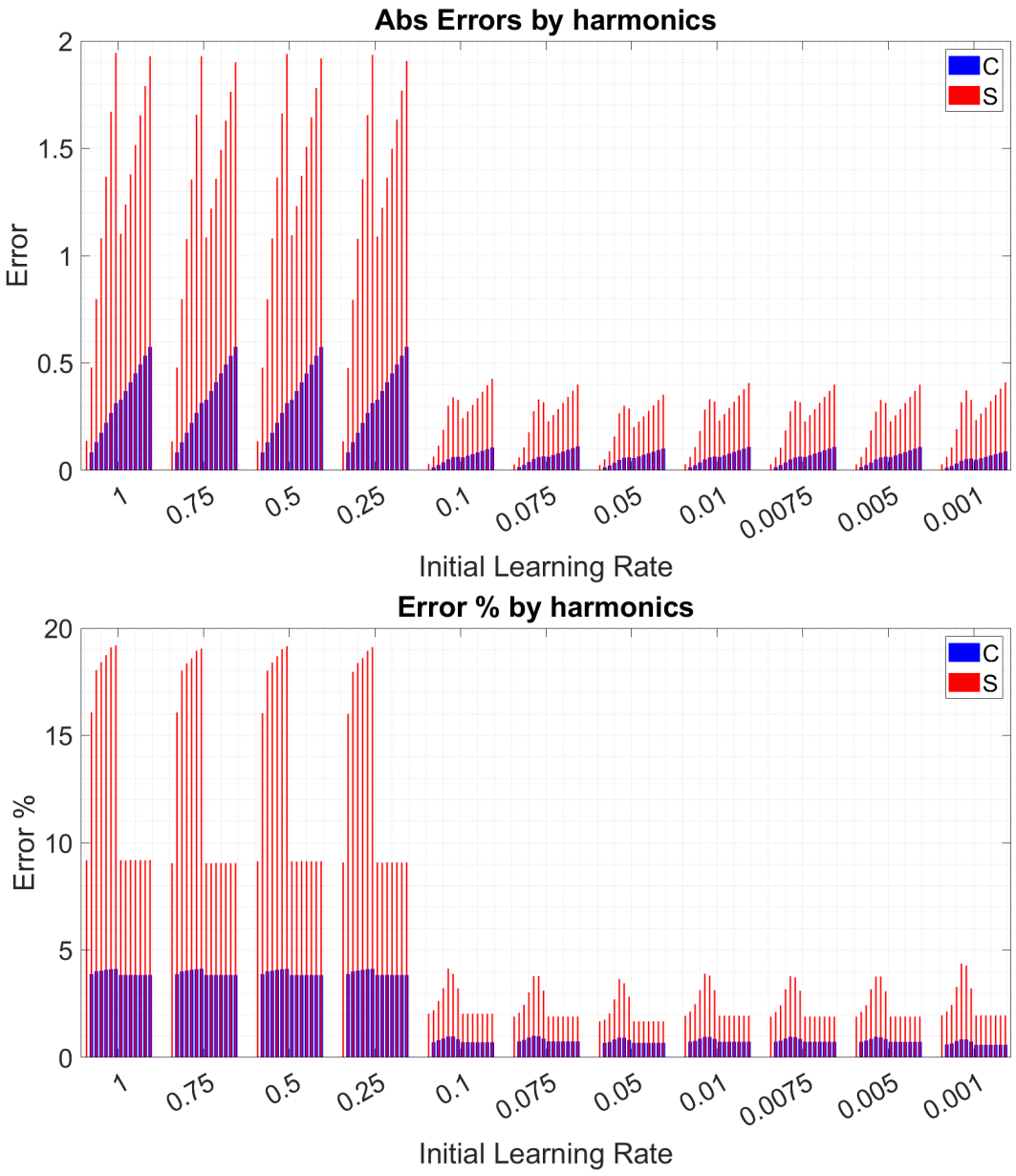

Figura B.4: Métricas numéricas por harmónicos del análisis 1D *Initial Learning Rate*.

B.3. Métricas 1D Drop Period

Volver al informe

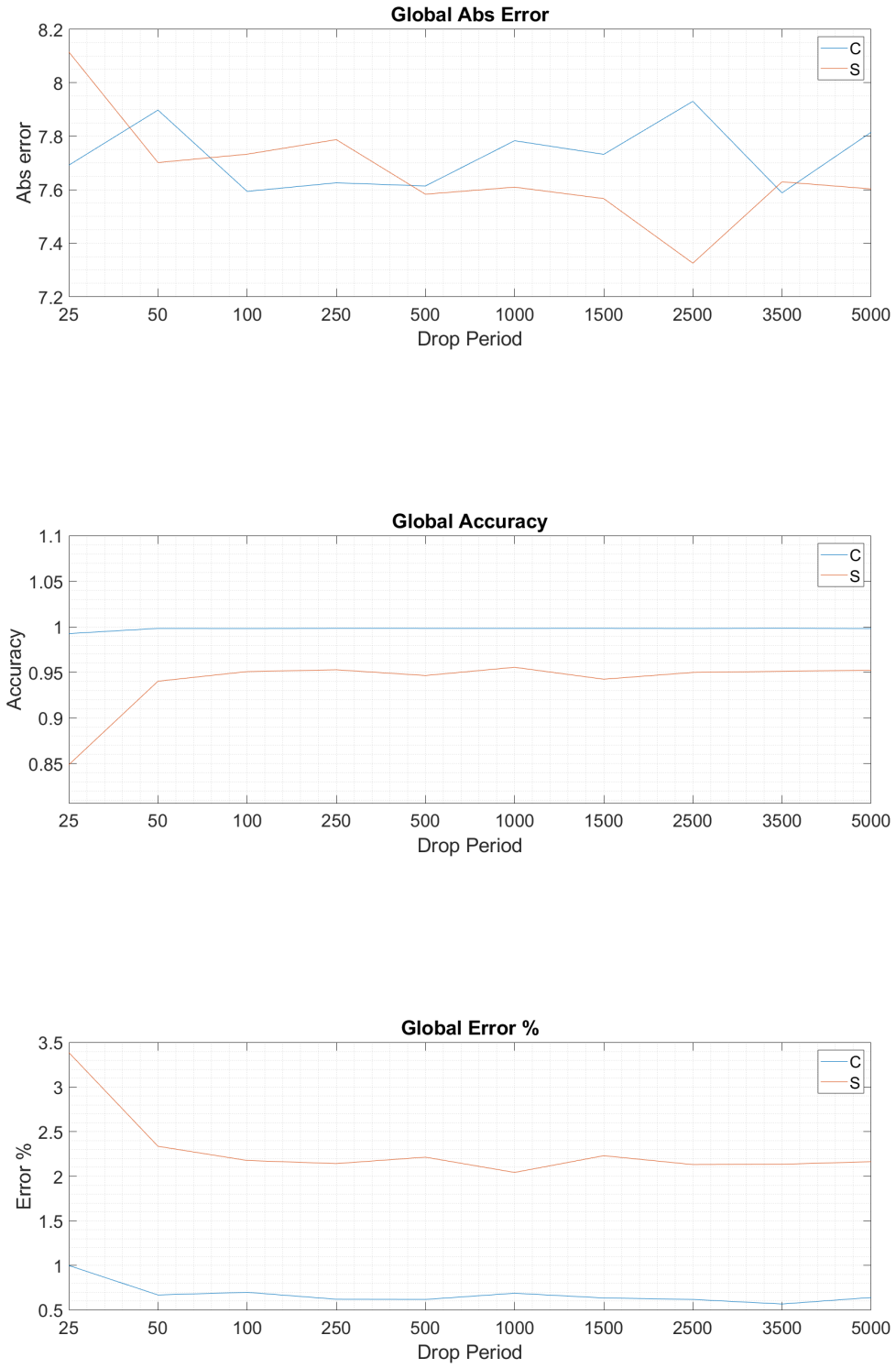

Figura B.5: Métricas numéricas globales del análisis 1D *Drop Period*.

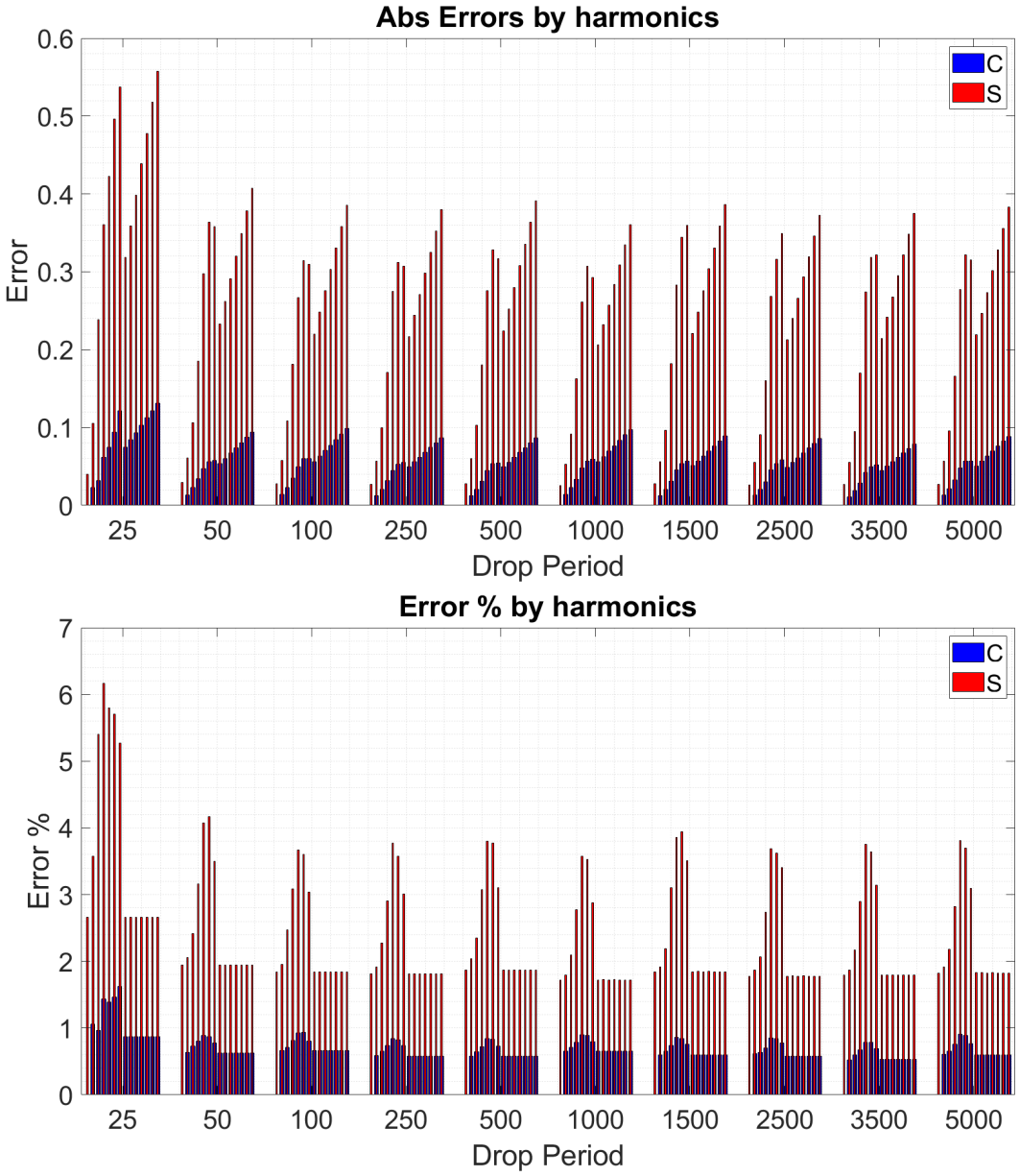

Figura B.6: Métricas numéricas por harmónicos del análisis 1D *Drop Period*.

B.4. Métricas 1D Drop Factor

Volver al informe

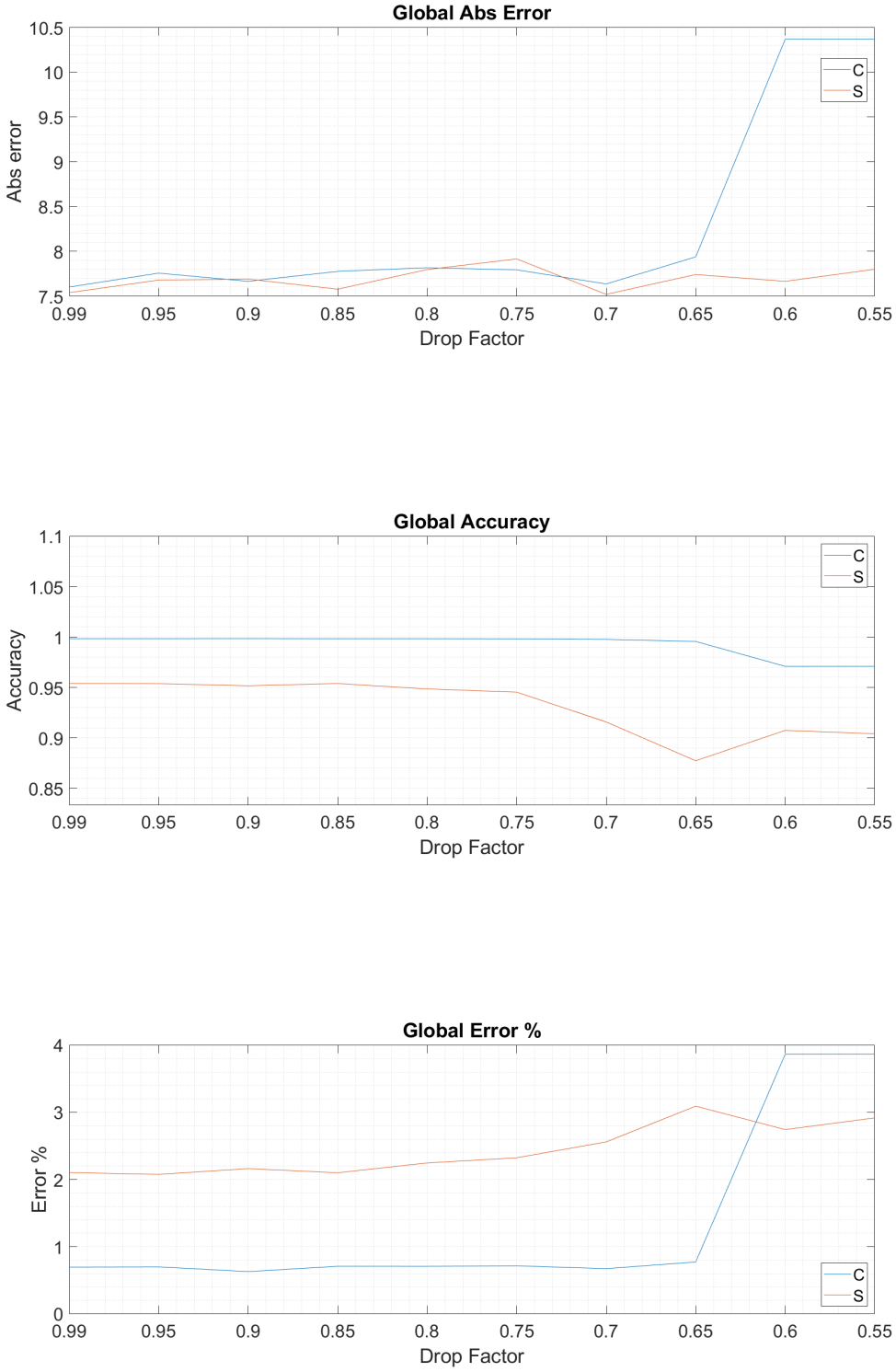

Figura B.7: Métricas numéricas globales del análisis 1D *Drop Factor*.

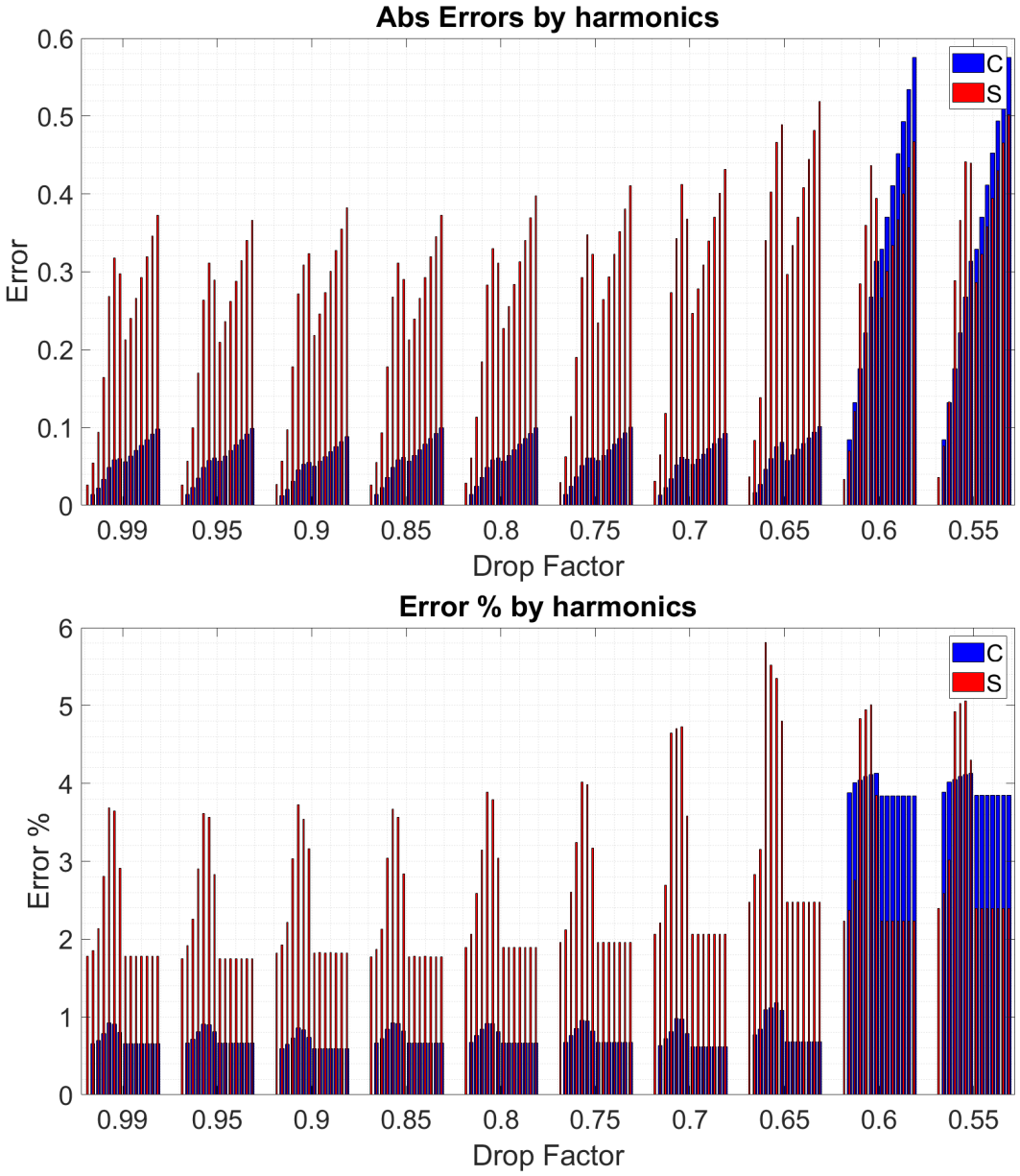

Figura B.8: Métricas numéricas por harmónicos del análisis 1D *Drop Factor*.

B.5. Métricas 1D Número de neuronas por capa

Ensayo	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	D.Period	D.Factor	ILR	V.Freq	Epochs
1	25	25	250	0.99	0.05	2000	8000
2	25	50	250	0.99	0.05	2000	8000
3	25	100	250	0.99	0.05	2000	8000
4	25	150	250	0.99	0.05	2000	8000
5	25	200	250	0.99	0.05	2000	8000
6	50	25	250	0.99	0.05	2000	8000
7	50	50	250	0.99	0.05	2000	8000
8	50	100	250	0.99	0.05	2000	8000
9	50	150	250	0.99	0.05	2000	8000
10	50	200	250	0.99	0.05	2000	8000
11	100	25	250	0.99	0.05	2000	8000
12	100	50	250	0.99	0.05	2000	8000
13	100	100	250	0.99	0.05	2000	8000
14	100	150	250	0.99	0.05	2000	8000
15	100	200	250	0.99	0.05	2000	8000
16	150	25	250	0.99	0.05	2000	8000
17	150	50	250	0.99	0.05	2000	8000
18	150	100	250	0.99	0.05	2000	8000
19	150	150	250	0.99	0.05	2000	8000
20	150	200	250	0.99	0.05	2000	8000
21	200	25	250	0.99	0.05	2000	8000
22	200	50	250	0.99	0.05	2000	8000
23	200	100	250	0.99	0.05	2000	8000
24	200	150	250	0.99	0.05	2000	8000
25	200	200	250	0.99	0.05	2000	8000

Tabla B.1: Ensayos del análisis del número de neuronas por capa.

Volver al informe

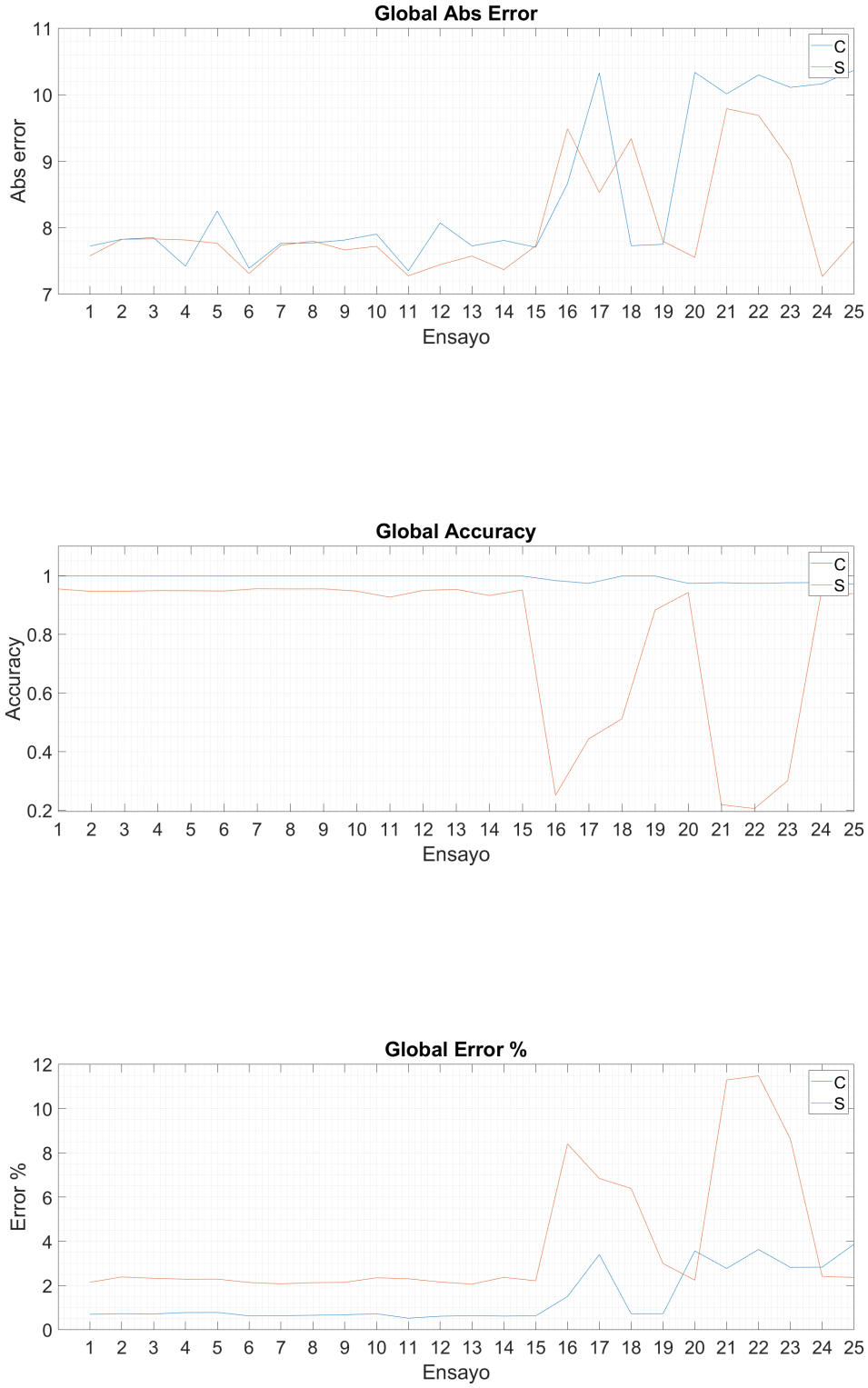

Figura B.9: Métricas numéricas globales del análisis 1D del número de neuronas por capa.

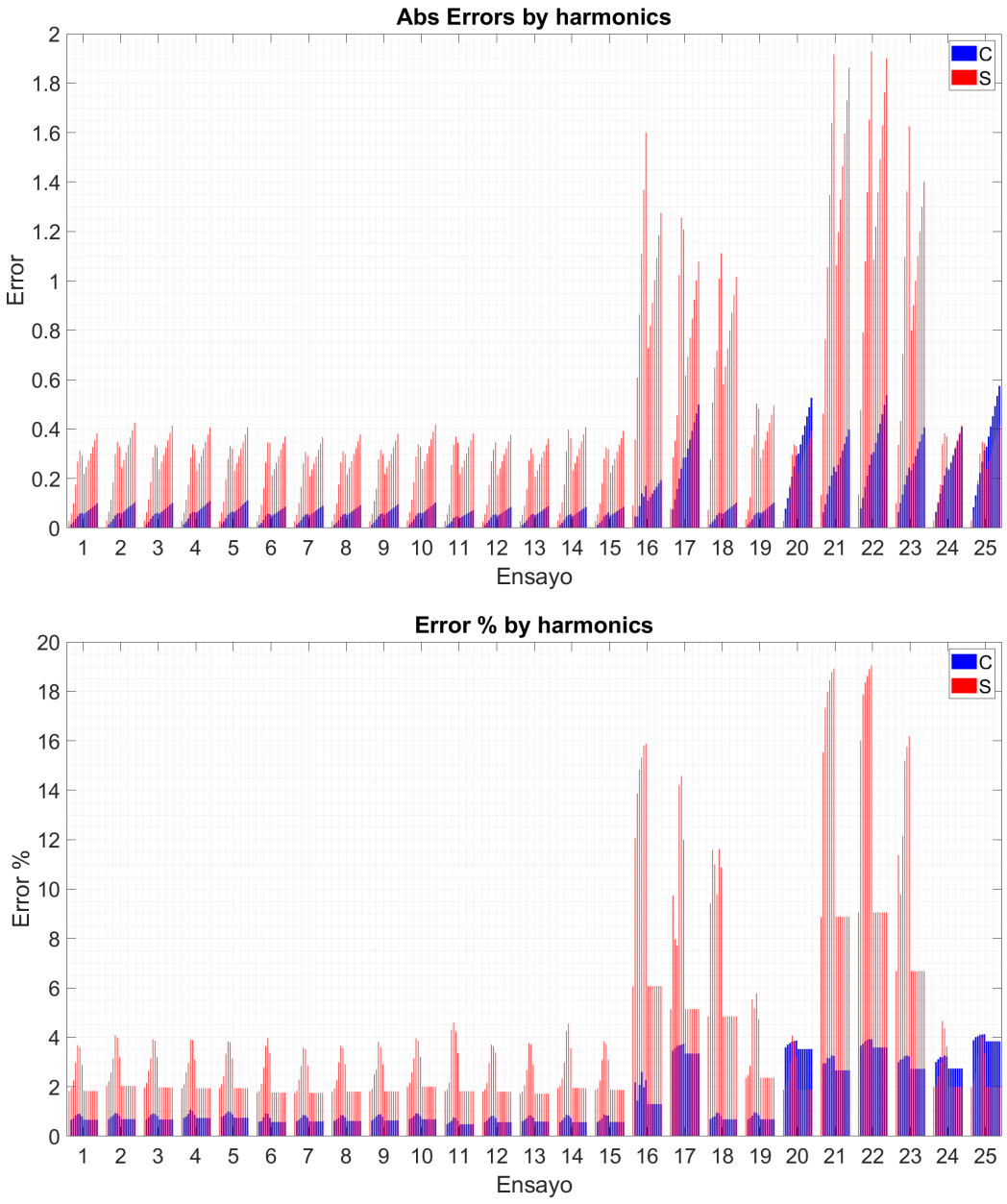

Figura B.10: Métricas numéricas por harmónicos del análisis 1D del número de neuronas por capa.

Apéndice C

Métricas Análisis 2D redes MLP

C.1. Métricas 2D ILR - DF

Ensayo	ILR	DF	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	DP	V.Freq	Epochs
1	0.075	0.99	100	100	250	2000	8000
2	0.075	0.95	100	100	250	2000	8000
3	0.075	0.9	100	100	250	2000	8000
4	0.075	0.85	100	100	250	2000	8000
5	0.075	0.8	100	100	250	2000	8000
6	0.05	0.99	100	100	250	2000	8000
7	0.05	0.95	100	100	250	2000	8000
8	0.05	0.9	100	100	250	2000	8000
9	0.05	0.85	100	100	250	2000	8000
10	0.05	0.8	100	100	250	2000	8000
11	0.01	0.99	100	100	250	2000	8000
12	0.01	0.95	100	100	250	2000	8000
13	0.01	0.9	100	100	250	2000	8000
14	0.01	0.85	100	100	250	2000	8000
15	0.01	0.8	100	100	250	2000	8000
16	0.0075	0.99	100	100	250	2000	8000
17	0.0075	0.95	100	100	250	2000	8000
18	0.0075	0.9	100	100	250	2000	8000
19	0.0075	0.85	100	100	250	2000	8000
20	0.0075	0.8	100	100	250	2000	8000
21	0.005	0.99	100	100	250	2000	8000
22	0.005	0.95	100	100	250	2000	8000
23	0.005	0.9	100	100	250	2000	8000
24	0.005	0.85	100	100	250	2000	8000
25	0.005	0.8	100	100	250	2000	8000

Tabla C.1: Ensayos del análisis 2D *ILR - DF*.

Volver al informe

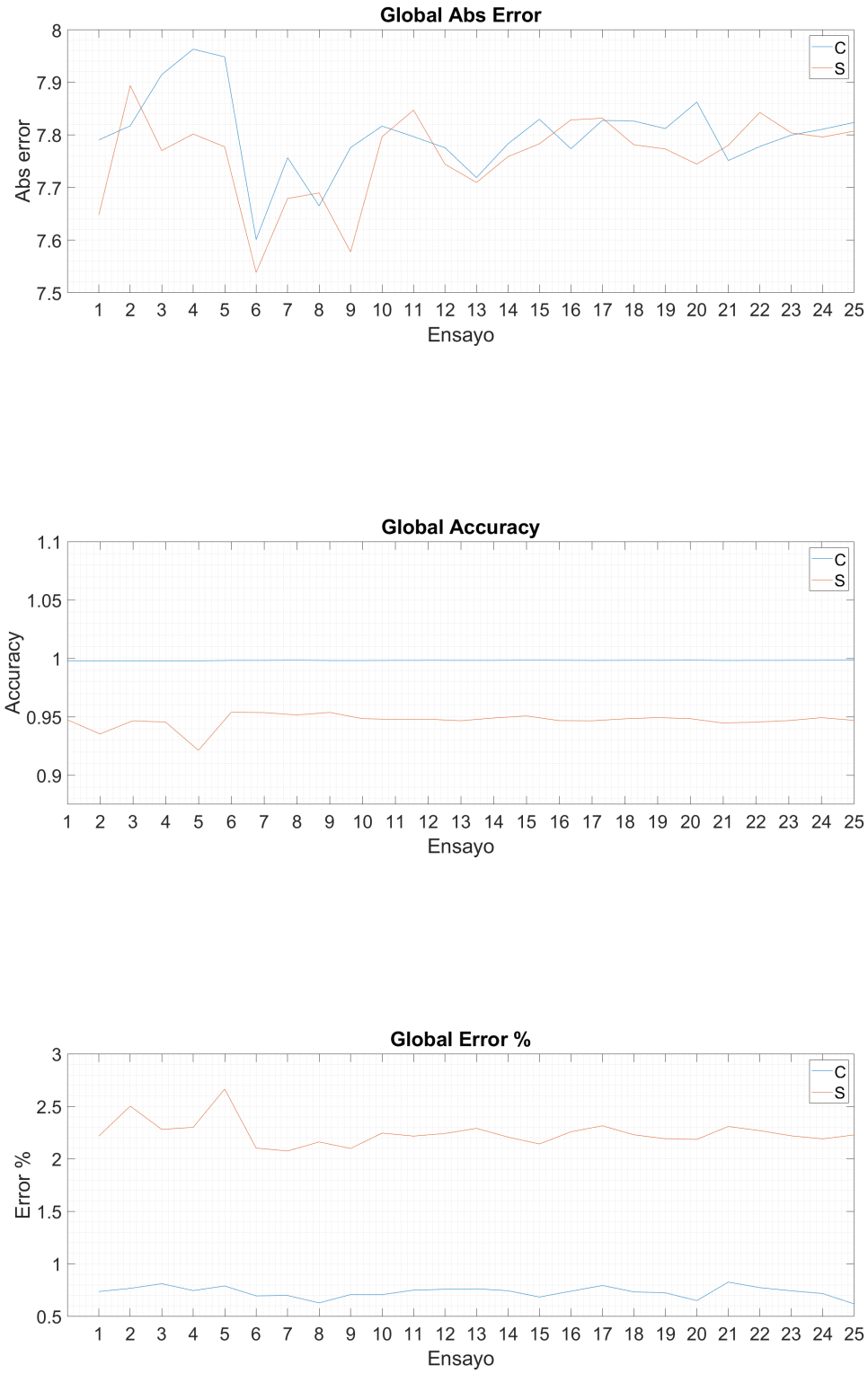

Figura C.1: Métricas numéricas globales del análisis 2D *ILR-DF*.

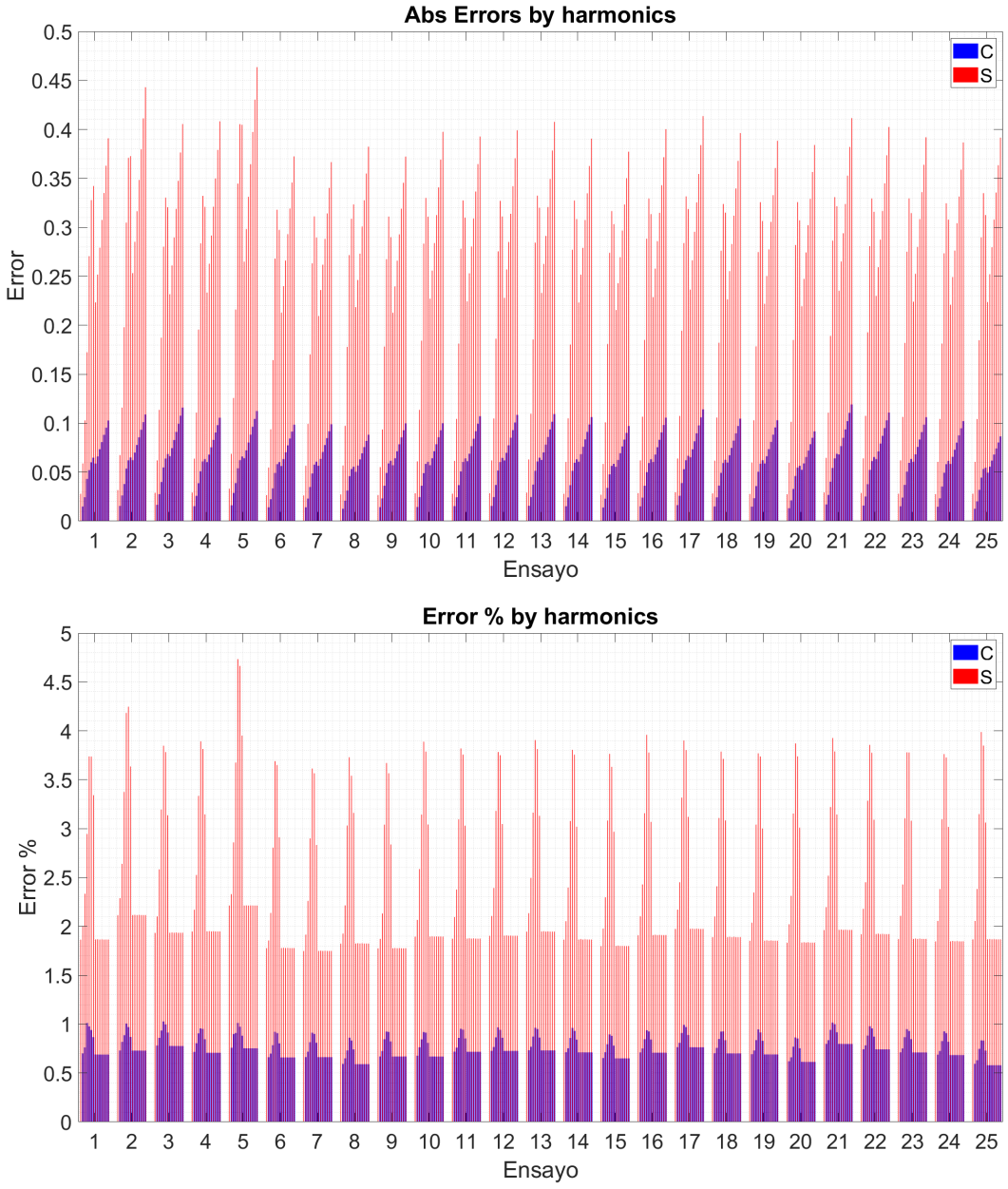

Figura C.2: Métricas numéricas por harmónicos del análisis 2D *ILR-DF*.

C.2. Métricas 2D ILR - DP

Ensayo	ILR	DP	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	DF	V.Freq	Epochs
1	0.075	250	100	100	0.99	2000	8000
2	0.075	500	100	100	0.99	2000	8000
3	0.075	1000	100	100	0.99	2000	8000
4	0.075	1500	100	100	0.99	2000	8000
5	0.075	2000	100	100	0.99	2000	8000
6	0.05	250	100	100	0.99	2000	8000
7	0.05	500	100	100	0.99	2000	8000
8	0.05	1000	100	100	0.99	2000	8000
9	0.05	1500	100	100	0.99	2000	8000
10	0.05	2000	100	100	0.99	2000	8000
11	0.01	250	100	100	0.99	2000	8000
12	0.01	500	100	100	0.99	2000	8000
13	0.01	1000	100	100	0.99	2000	8000
14	0.01	1500	100	100	0.99	2000	8000
15	0.01	2000	100	100	0.99	2000	8000
16	0.0075	250	100	100	0.99	2000	8000
17	0.0075	500	100	100	0.99	2000	8000
18	0.0075	1000	100	100	0.99	2000	8000
19	0.0075	1500	100	100	0.99	2000	8000
20	0.0075	2000	100	100	0.99	2000	8000
21	0.005	250	100	100	0.99	2000	8000
22	0.005	500	100	100	0.99	2000	8000
23	0.005	1000	100	100	0.99	2000	8000
24	0.005	1500	100	100	0.99	2000	8000
25	0.005	2000	100	100	0.99	2000	8000

Tabla C.2: Ensayos del análisis 2D *ILR - DP*.

Volver al informe

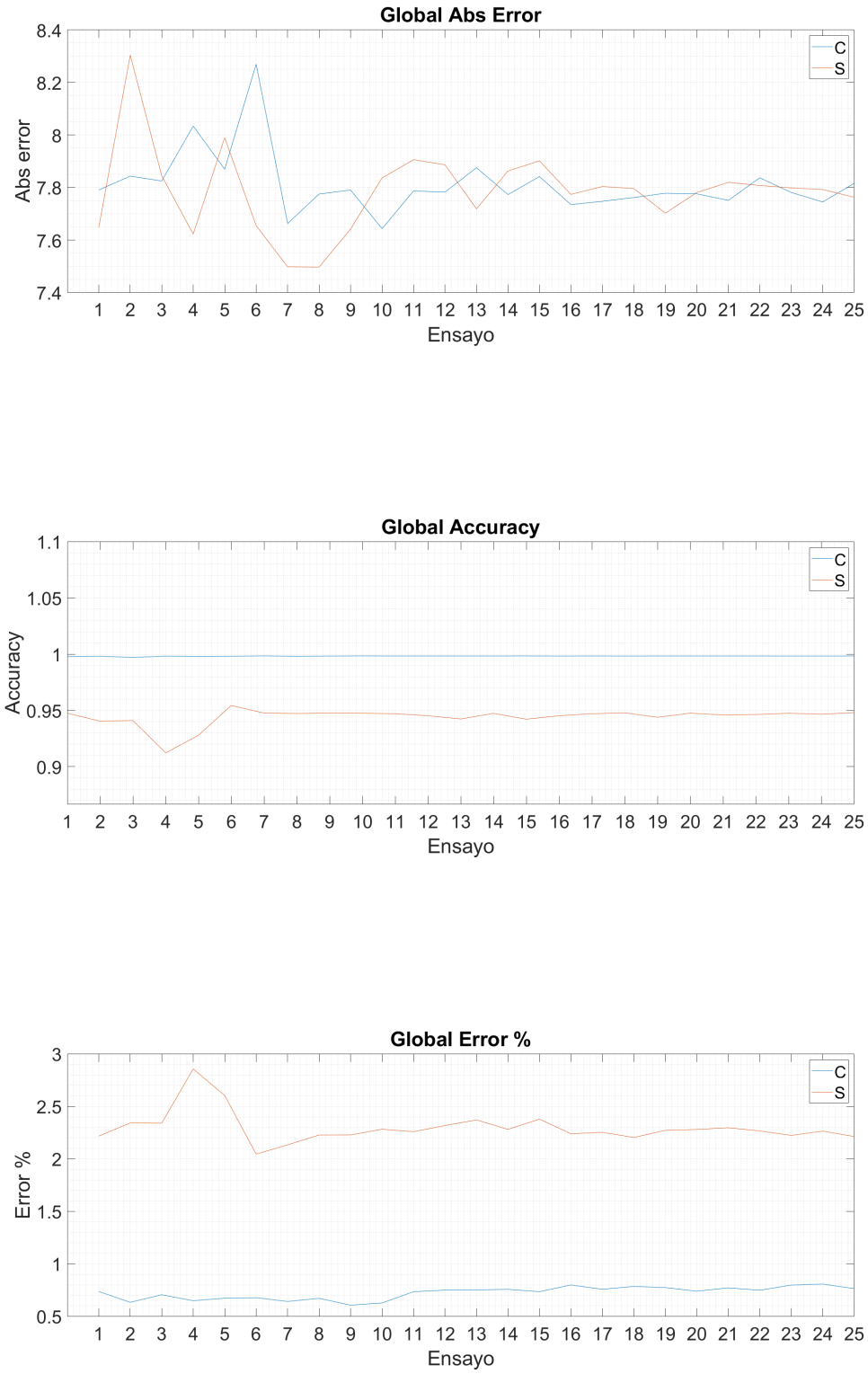

Figura C.3: Métricas numéricas globales del análisis 2D *ILR-DP*.

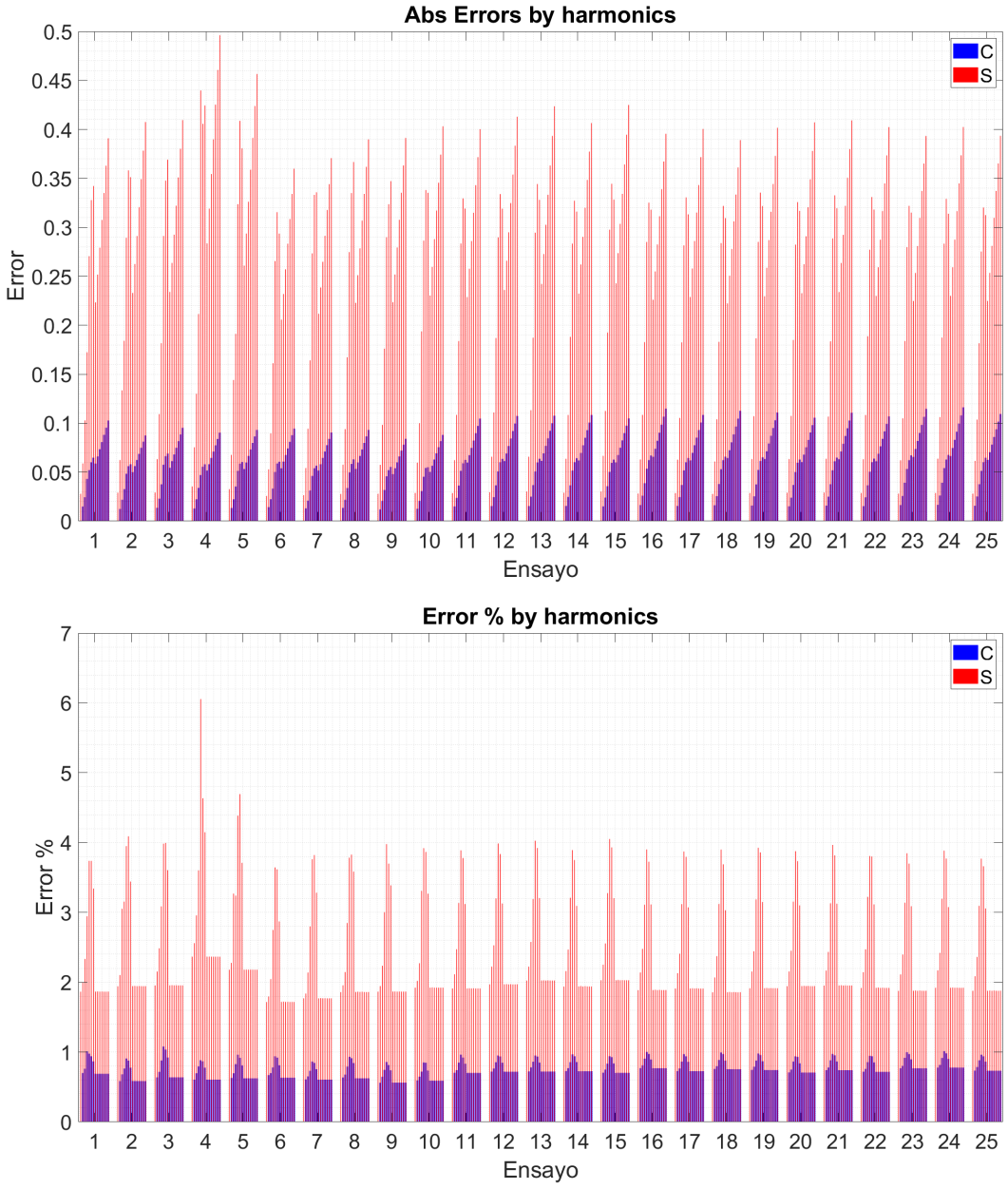

Figura C.4: Métricas numéricas por harmónicos del análisis 2D *ILR-DP*.

C.3. Métricas 2D DF - DP

Ensayo	DF	DP	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	ILR	V.Freq	Epochs
1	0.99	50	100	100	0.05	2000	8000
2	0.99	100	100	100	0.05	2000	8000
3	0.99	250	100	100	0.05	2000	8000
4	0.99	500	100	100	0.05	2000	8000
5	0.99	1000	100	100	0.05	2000	8000
6	0.95	50	100	100	0.05	2000	8000
7	0.95	100	100	100	0.05	2000	8000
8	0.95	250	100	100	0.05	2000	8000
9	0.95	500	100	100	0.05	2000	8000
10	0.95	1000	100	100	0.05	2000	8000
11	0.9	50	100	100	0.05	2000	8000
12	0.9	100	100	100	0.05	2000	8000
13	0.9	250	100	100	0.05	2000	8000
14	0.9	500	100	100	0.05	2000	8000
15	0.9	1000	100	100	0.05	2000	8000
16	0.85	50	100	100	0.05	2000	8000
17	0.85	100	100	100	0.05	2000	8000
18	0.85	250	100	100	0.05	2000	8000
19	0.85	500	100	100	0.05	2000	8000
20	0.85	1000	100	100	0.05	2000	8000
21	0.8	50	100	100	0.05	2000	8000
22	0.8	100	100	100	0.05	2000	8000
23	0.8	250	100	100	0.05	2000	8000
24	0.8	500	100	100	0.05	2000	8000
25	0.8	1000	100	100	0.05	2000	8000

Tabla C.3: Ensayos del análisis 2D *DF - DP*.

Volver al informe

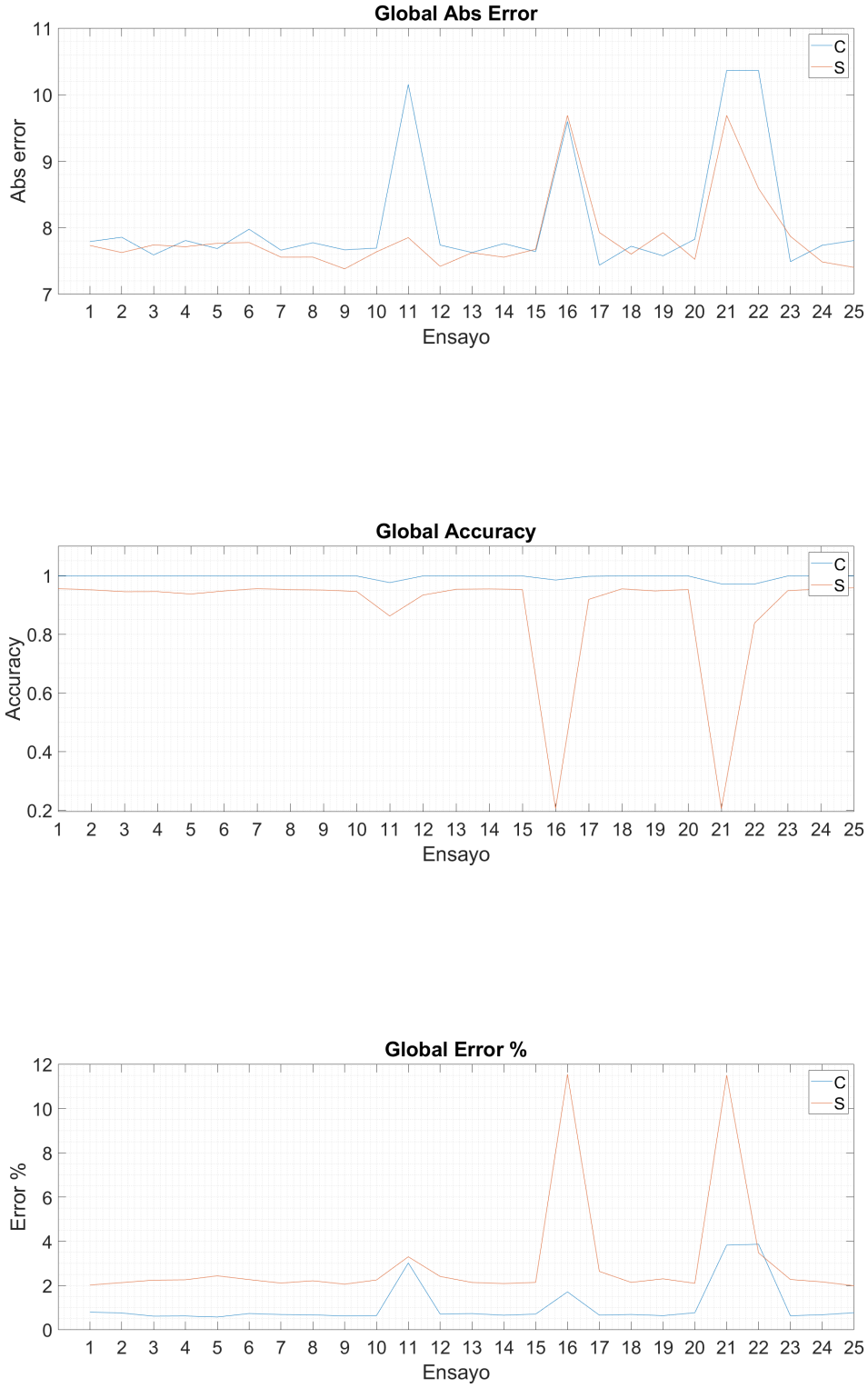

Figura C.5: Métricas numéricas globales del análisis 2D *DF-DP*.

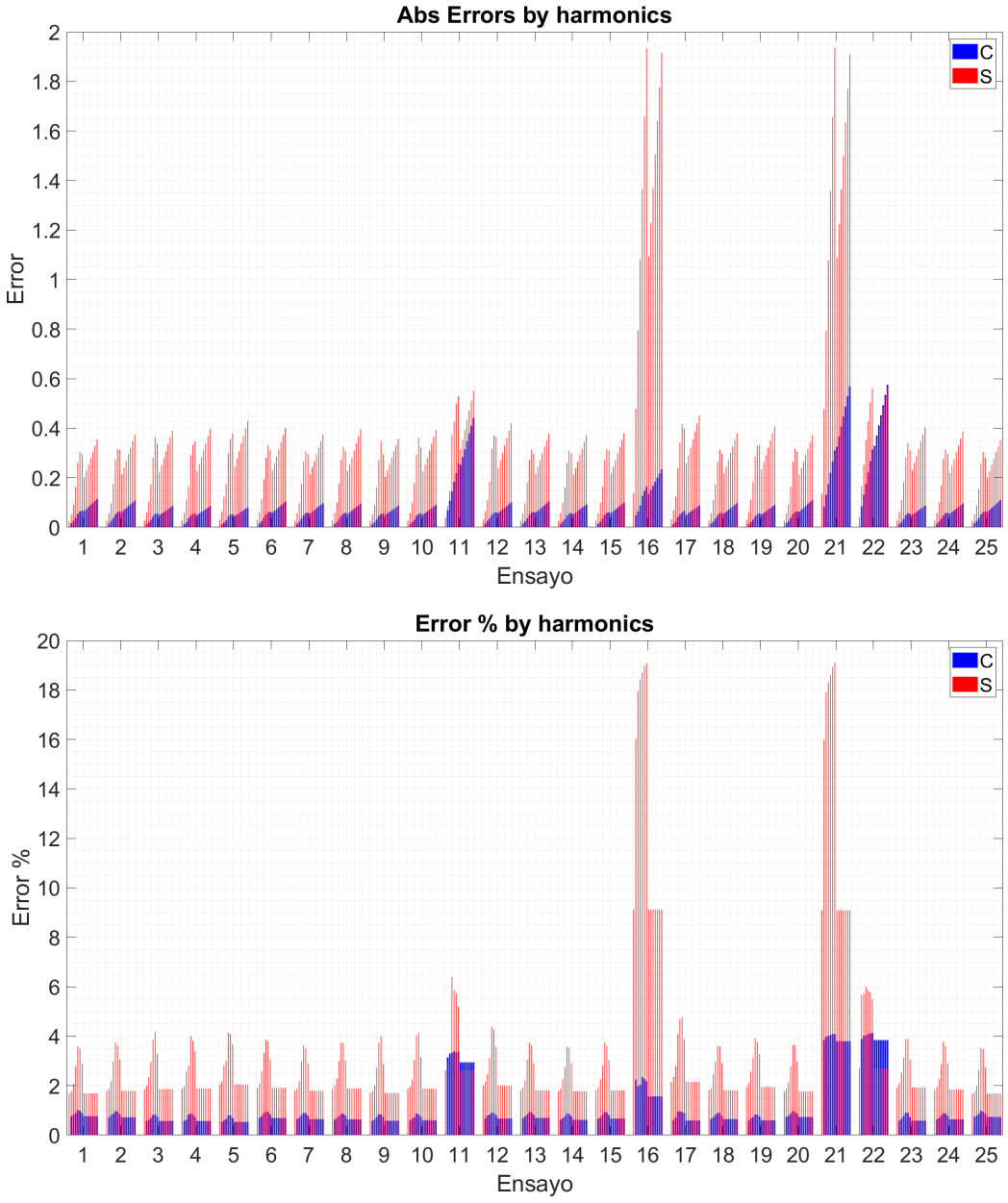

Figura C.6: Métricas numéricas por harmónicos del análisis 2D *DF-DP*.

C.4. Métricas 2D Número de neuronas por capa - ILR

Ensayo	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	ILR	DP	DF	V.Freq	Epochs
1	50	25	0.1	1000	0.8	2000	8000
2	50	25	0.05	1000	0.8	2000	8000
3	50	25	0.025	1000	0.8	2000	8000
4	50	25	0.01	1000	0.8	2000	8000
5	50	25	0.005	1000	0.8	2000	8000
6	100	25	0.1	1000	0.8	2000	8000
7	100	25	0.05	1000	0.8	2000	8000
8	100	25	0.025	1000	0.8	2000	8000
9	100	25	0.01	1000	0.8	2000	8000
10	100	25	0.005	1000	0.8	2000	8000
11	50	50	0.1	1000	0.8	2000	8000
12	50	50	0.05	1000	0.8	2000	8000
13	50	50	0.025	1000	0.8	2000	8000
14	50	50	0.01	1000	0.8	2000	8000
15	50	50	0.005	1000	0.8	2000	8000
16	100	50	0.1	1000	0.8	2000	8000
17	100	50	0.05	1000	0.8	2000	8000
18	100	50	0.025	1000	0.8	2000	8000
19	100	50	0.01	1000	0.8	2000	8000
20	100	50	0.005	1000	0.8	2000	8000
21	100	75	0.1	1000	0.8	2000	8000
22	100	75	0.05	1000	0.8	2000	8000
23	100	75	0.025	1000	0.8	2000	8000
24	100	75	0.01	1000	0.8	2000	8000
25	100	75	0.005	1000	0.8	2000	8000

Tabla C.4: Ensayos del análisis 2D del número de neuronas por capa - *ILR*.

Volver al informe

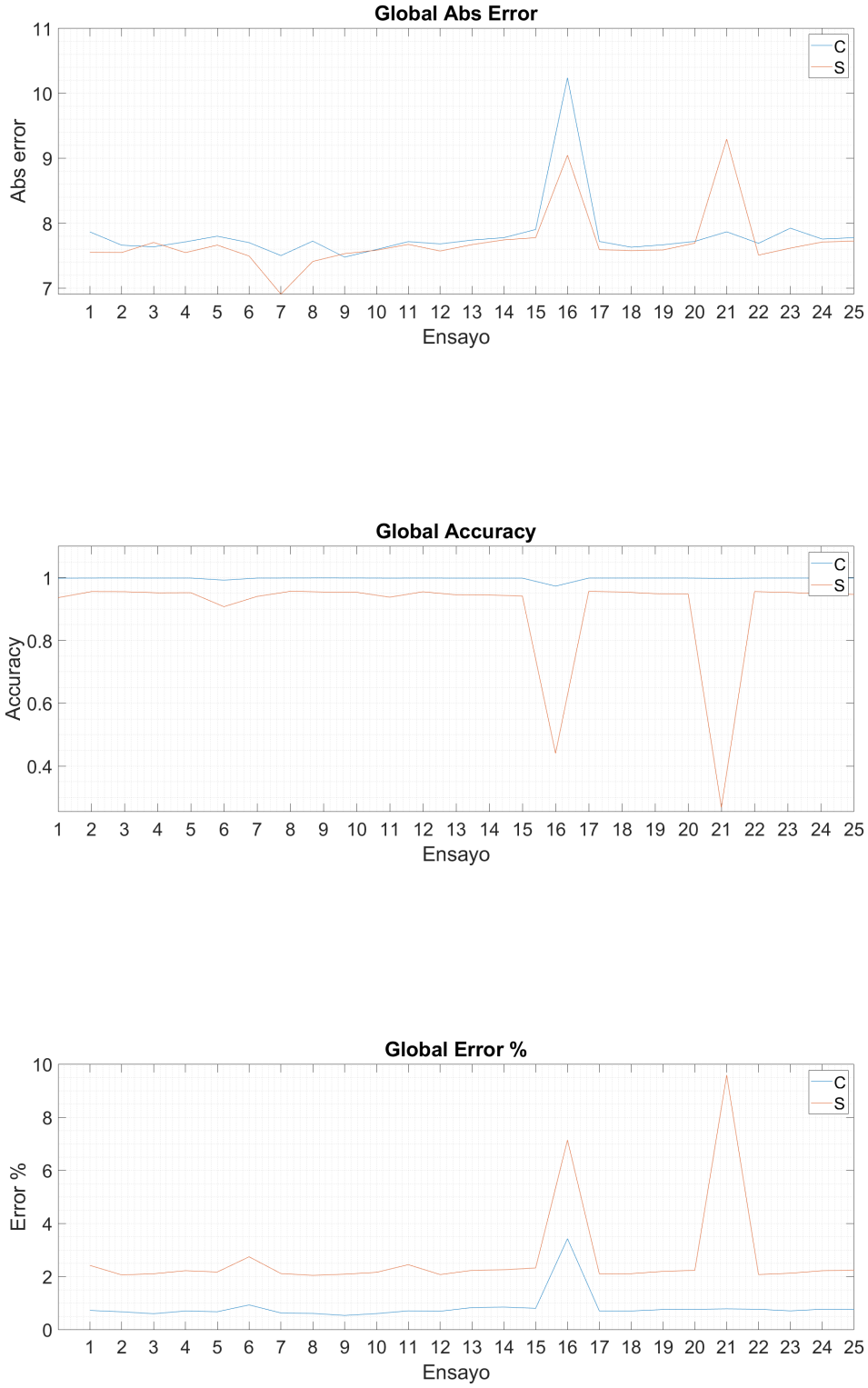

Figura C.7: Métricas numéricas globales del análisis 2D del número de neuronas por capa - *ILR*.

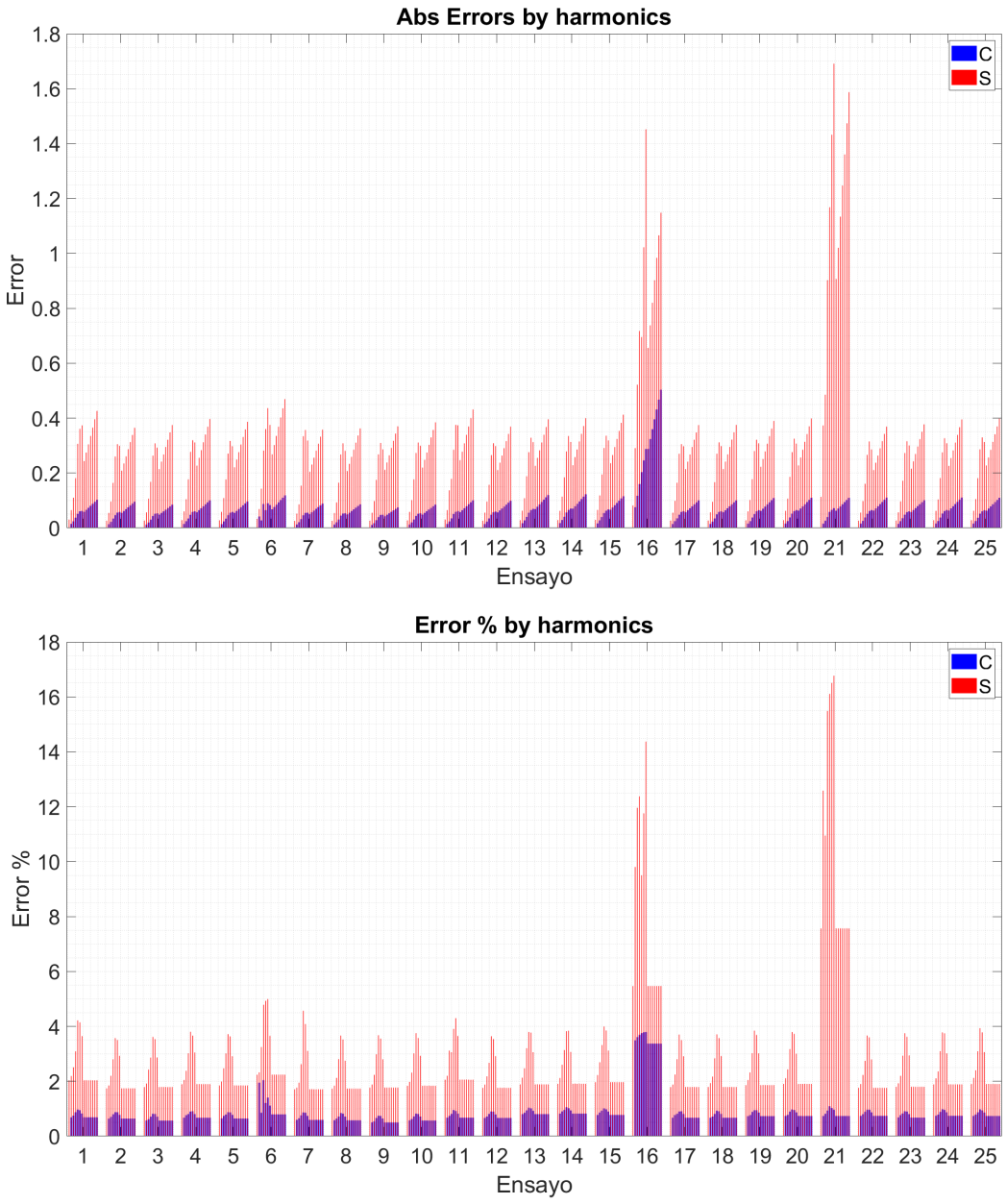

Figura C.8: Métricas numéricas por harmónicos del análisis 2D del número de neuronas por capa - *ILR*.

C.5. Métricas 2D Número de neuronas por capa - DF

Ensayo	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	DF	ILR	DP	V.Freq	Epochs
1	50	25	0.99	0.025	1000	2000	8000
2	50	25	0.95	0.025	1000	2000	8000
3	50	25	0.9	0.025	1000	2000	8000
4	50	25	0.85	0.025	1000	2000	8000
5	50	25	0.8	0.025	1000	2000	8000
6	100	25	0.99	0.025	1000	2000	8000
7	100	25	0.95	0.025	1000	2000	8000
8	100	25	0.9	0.025	1000	2000	8000
9	100	25	0.85	0.025	1000	2000	8000
10	100	25	0.8	0.025	1000	2000	8000
11	50	50	0.99	0.025	1000	2000	8000
12	50	50	0.95	0.025	1000	2000	8000
13	50	50	0.9	0.025	1000	2000	8000
14	50	50	0.85	0.025	1000	2000	8000
15	50	50	0.8	0.025	1000	2000	8000
16	100	50	0.99	0.025	1000	2000	8000
17	100	50	0.95	0.025	1000	2000	8000
18	100	50	0.9	0.025	1000	2000	8000
19	100	50	0.85	0.025	1000	2000	8000
20	100	50	0.8	0.025	1000	2000	8000
21	100	75	0.99	0.025	1000	2000	8000
22	100	75	0.95	0.025	1000	2000	8000
23	100	75	0.9	0.025	1000	2000	8000
24	100	75	0.85	0.025	1000	2000	8000
25	100	75	0.8	0.025	1000	2000	8000

Tabla C.5: Ensayos del análisis 2D del número de neuronas por capa - *DF*.

Volver al informe

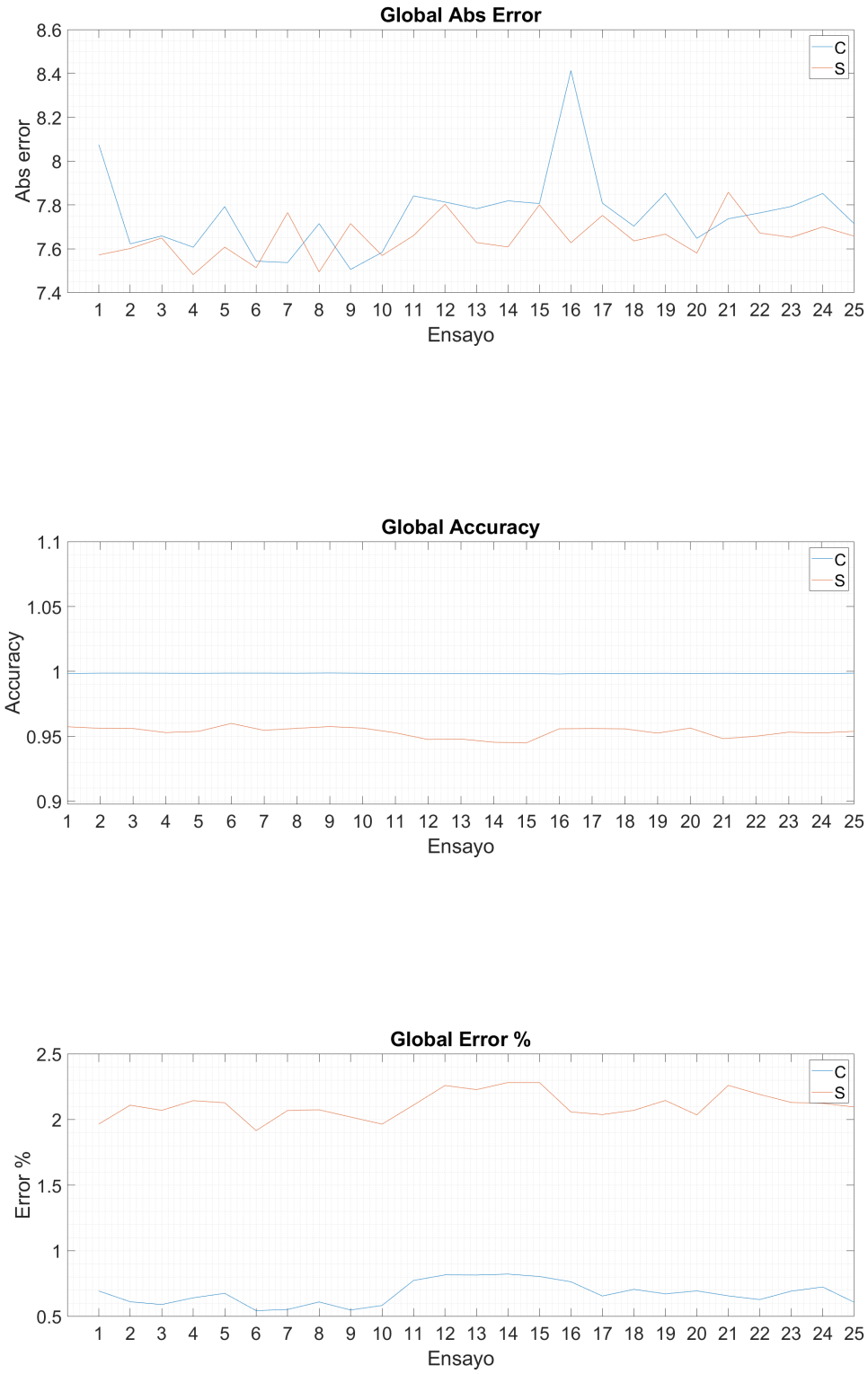

Figura C.9: Métricas numéricas globales del análisis 2D del número de neuronas por capa - *DF*.

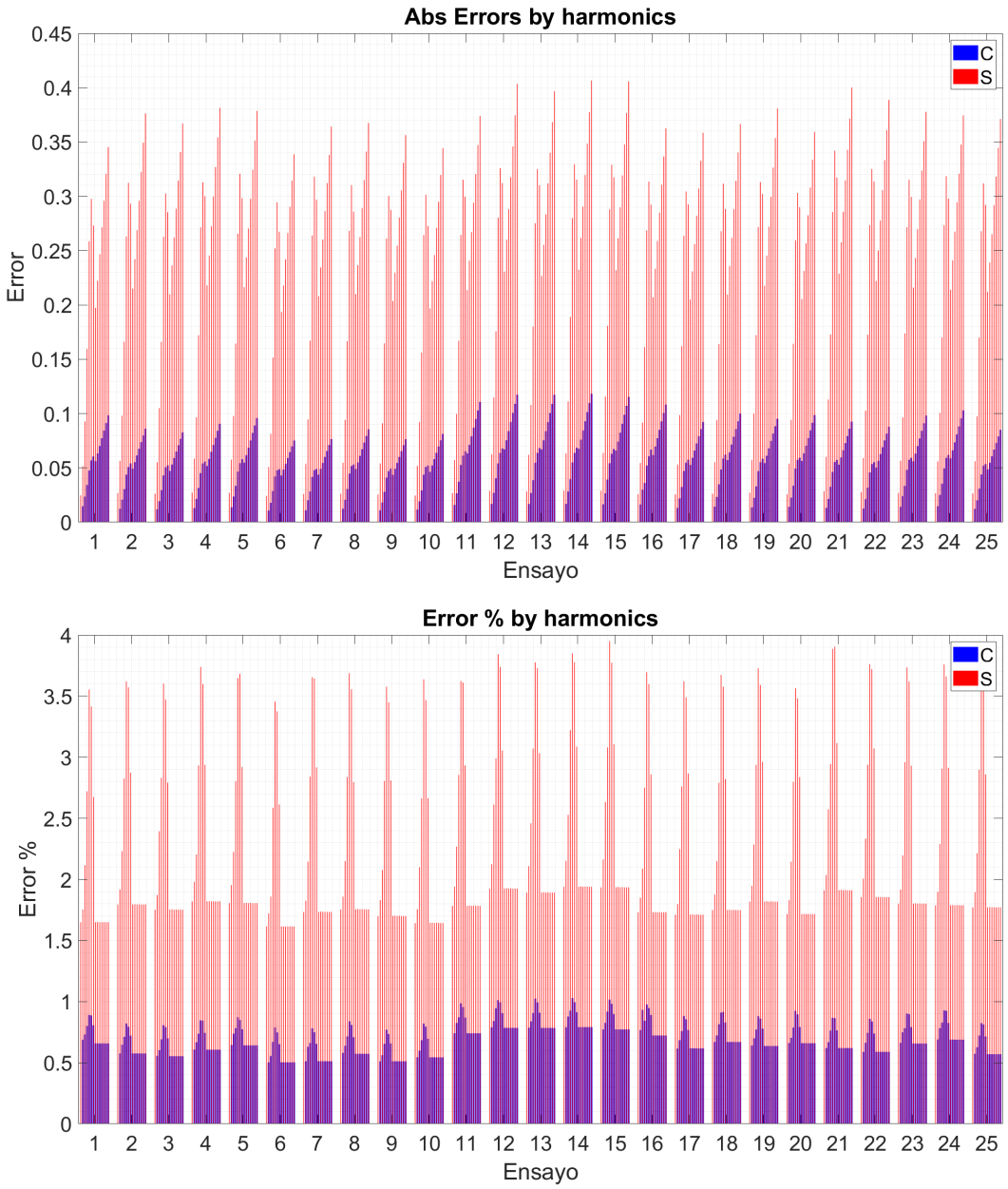

Figura C.10: Métricas numéricas por harmónicos del análisis 2D del número de neuronas por capa - *DF*.

C.6. Métricas 2D Número de neuronas por capa - DP

Ensayo	Nº Neuronas 1ªCapa	Nº Neuronas 2ªCapa	DP	ILR	DF	V.Freq	Epochs
1	50	25	50	0.025	0.95	2000	8000
2	50	25	100	0.025	0.95	2000	8000
3	50	25	500	0.025	0.95	2000	8000
4	50	25	1000	0.025	0.95	2000	8000
5	50	25	2500	0.025	0.95	2000	8000
6	100	25	50	0.025	0.95	2000	8000
7	100	25	100	0.025	0.95	2000	8000
8	100	25	500	0.025	0.95	2000	8000
9	100	25	1000	0.025	0.95	2000	8000
10	100	25	2500	0.025	0.95	2000	8000
11	50	50	50	0.025	0.95	2000	8000
12	50	50	100	0.025	0.95	2000	8000
13	50	50	500	0.025	0.95	2000	8000
14	50	50	1000	0.025	0.95	2000	8000
15	50	50	2500	0.025	0.95	2000	8000
16	100	50	50	0.025	0.95	2000	8000
17	100	50	100	0.025	0.95	2000	8000
18	100	50	500	0.025	0.95	2000	8000
19	100	50	1000	0.025	0.95	2000	8000
20	100	50	2500	0.025	0.95	2000	8000
21	100	75	50	0.025	0.95	2000	8000
22	100	75	100	0.025	0.95	2000	8000
23	100	75	500	0.025	0.95	2000	8000
24	100	75	1000	0.025	0.95	2000	8000
25	100	75	2500	0.025	0.95	2000	8000

Tabla C.6: Ensayos del análisis 2D del número de neuronas por capa - *DP*.

Volver al informe

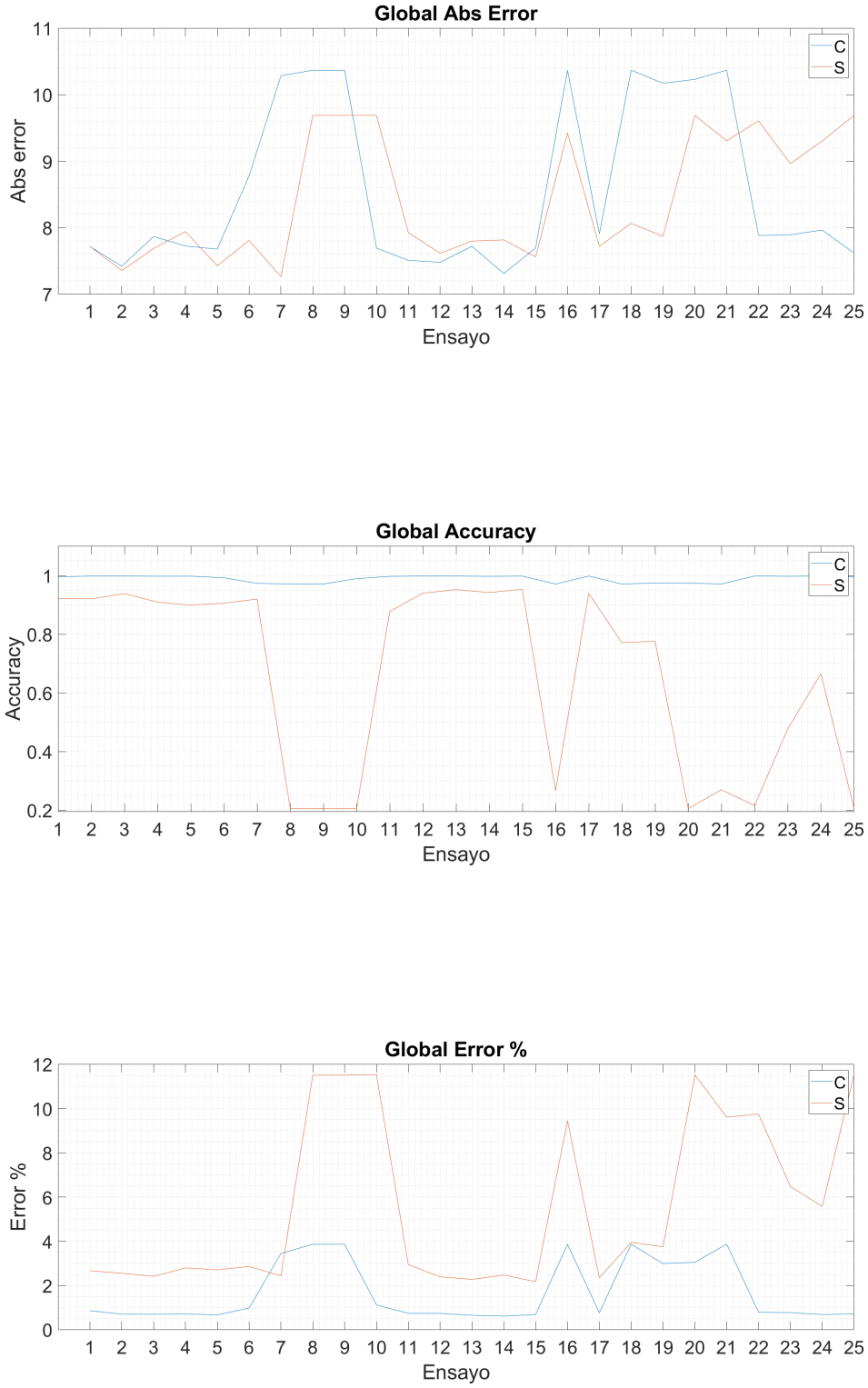

Figura C.11: Métricas numéricas globales del análisis 2D del número de neuronas por capa - *DP*.

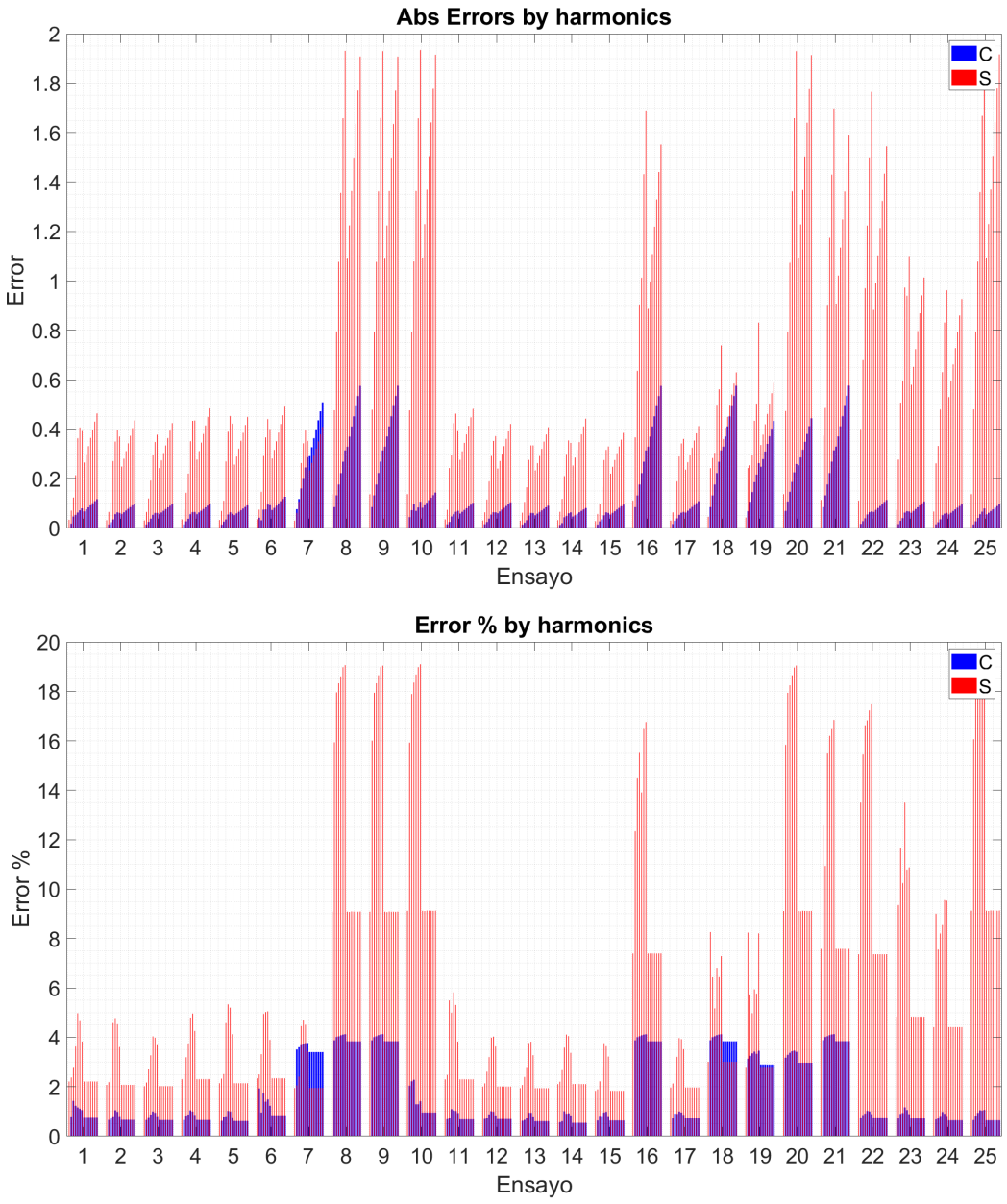

Figura C.12: Métricas numéricas por harmónicos del análisis 2D del número de neuronas por capa - *DP*.

Apéndice D

Métricas Análisis Global redes MLP

Tabla D.1: Ensayos del análisis global de las redes MLP.

Ensayo	Neuronas 1ªCapa	Neuronas 2ªCapa	D.Period	D.Factor	ILR	V.Freq	Epochs
1	50	25	50	0.99	0.1	2000	8000
2	50	25	50	0.99	0.075	2000	8000
3	50	25	50	0.99	0.05	2000	8000
4	50	25	50	0.95	0.1	2000	8000
5	50	25	50	0.95	0.075	2000	8000
6	50	25	50	0.95	0.05	2000	8000
7	50	25	50	0.9	0.1	2000	8000
8	50	25	50	0.9	0.075	2000	8000
9	50	25	50	0.9	0.05	2000	8000
10	50	25	100	0.99	0.1	2000	8000
11	50	25	100	0.99	0.075	2000	8000
12	50	25	100	0.99	0.05	2000	8000
13	50	25	100	0.95	0.1	2000	8000
14	50	25	100	0.95	0.075	2000	8000
15	50	25	100	0.95	0.05	2000	8000
16	50	25	100	0.9	0.1	2000	8000
17	50	25	100	0.9	0.075	2000	8000
18	50	25	100	0.9	0.05	2000	8000
19	50	25	250	0.99	0.1	2000	8000
20	50	25	250	0.99	0.075	2000	8000
21	50	25	250	0.99	0.05	2000	8000
22	50	25	250	0.95	0.1	2000	8000
23	50	25	250	0.95	0.075	2000	8000
24	50	25	250	0.95	0.05	2000	8000
25	50	25	250	0.9	0.1	2000	8000
26	50	25	250	0.9	0.075	2000	8000
27	50	25	250	0.9	0.05	2000	8000
28	100	25	50	0.99	0.1	2000	8000
29	100	25	50	0.99	0.075	2000	8000
30	100	25	50	0.99	0.05	2000	8000

Continúa en la siguiente página

Tabla D.1 – Continuación de la página anterior

Ensayo	Neuronas 1ªCapa	Neuronas 2ªCapa	D.Period	D.Factor	ILR	V.Freq	Epochs
31	100	25	50	0.95	0.1	2000	8000
32	100	25	50	0.95	0.075	2000	8000
33	100	25	50	0.95	0.05	2000	8000
34	100	25	50	0.9	0.1	2000	8000
35	100	25	50	0.9	0.075	2000	8000
36	100	25	50	0.9	0.05	2000	8000
37	100	25	100	0.99	0.1	2000	8000
38	100	25	100	0.99	0.075	2000	8000
39	100	25	100	0.99	0.05	2000	8000
40	100	25	100	0.95	0.1	2000	8000
41	100	25	100	0.95	0.075	2000	8000
42	100	25	100	0.95	0.05	2000	8000
43	100	25	100	0.9	0.1	2000	8000
44	100	25	100	0.9	0.075	2000	8000
45	100	25	100	0.9	0.05	2000	8000
46	100	25	250	0.99	0.1	2000	8000
47	100	25	250	0.99	0.075	2000	8000
48	100	25	250	0.99	0.05	2000	8000
49	100	25	250	0.95	0.1	2000	8000
50	100	25	250	0.95	0.075	2000	8000
51	100	25	250	0.95	0.05	2000	8000
52	100	25	250	0.9	0.1	2000	8000
53	100	25	250	0.9	0.075	2000	8000
54	100	25	250	0.9	0.05	2000	8000
55	50	50	50	0.99	0.1	2000	8000
56	50	50	50	0.99	0.075	2000	8000
57	50	50	50	0.99	0.05	2000	8000
58	50	50	50	0.95	0.1	2000	8000
59	50	50	50	0.95	0.075	2000	8000
60	50	50	50	0.95	0.05	2000	8000
61	50	50	50	0.9	0.1	2000	8000
62	50	50	50	0.9	0.075	2000	8000
63	50	50	50	0.9	0.05	2000	8000
64	50	50	100	0.99	0.1	2000	8000
65	50	50	100	0.99	0.075	2000	8000
66	50	50	100	0.99	0.05	2000	8000
67	50	50	100	0.95	0.1	2000	8000
68	50	50	100	0.95	0.075	2000	8000
69	50	50	100	0.95	0.05	2000	8000
70	50	50	100	0.9	0.1	2000	8000
71	50	50	100	0.9	0.075	2000	8000
72	50	50	100	0.9	0.05	2000	8000
73	50	50	250	0.99	0.1	2000	8000
74	50	50	250	0.99	0.075	2000	8000
75	50	50	250	0.99	0.05	2000	8000
76	50	50	250	0.95	0.1	2000	8000

Continúa en la siguiente página

Tabla D.1 – Continuación de la página anterior

Ensayo	Neuronas 1ªCapa	Neuronas 2ªCapa	D.Period	D.Factor	ILR	V.Freq	Epochs
77	50	50	250	0.95	0.075	2000	8000
78	50	50	250	0.95	0.05	2000	8000
79	50	50	250	0.9	0.1	2000	8000
80	50	50	250	0.9	0.075	2000	8000
81	50	50	250	0.9	0.05	2000	8000

Volver al informe

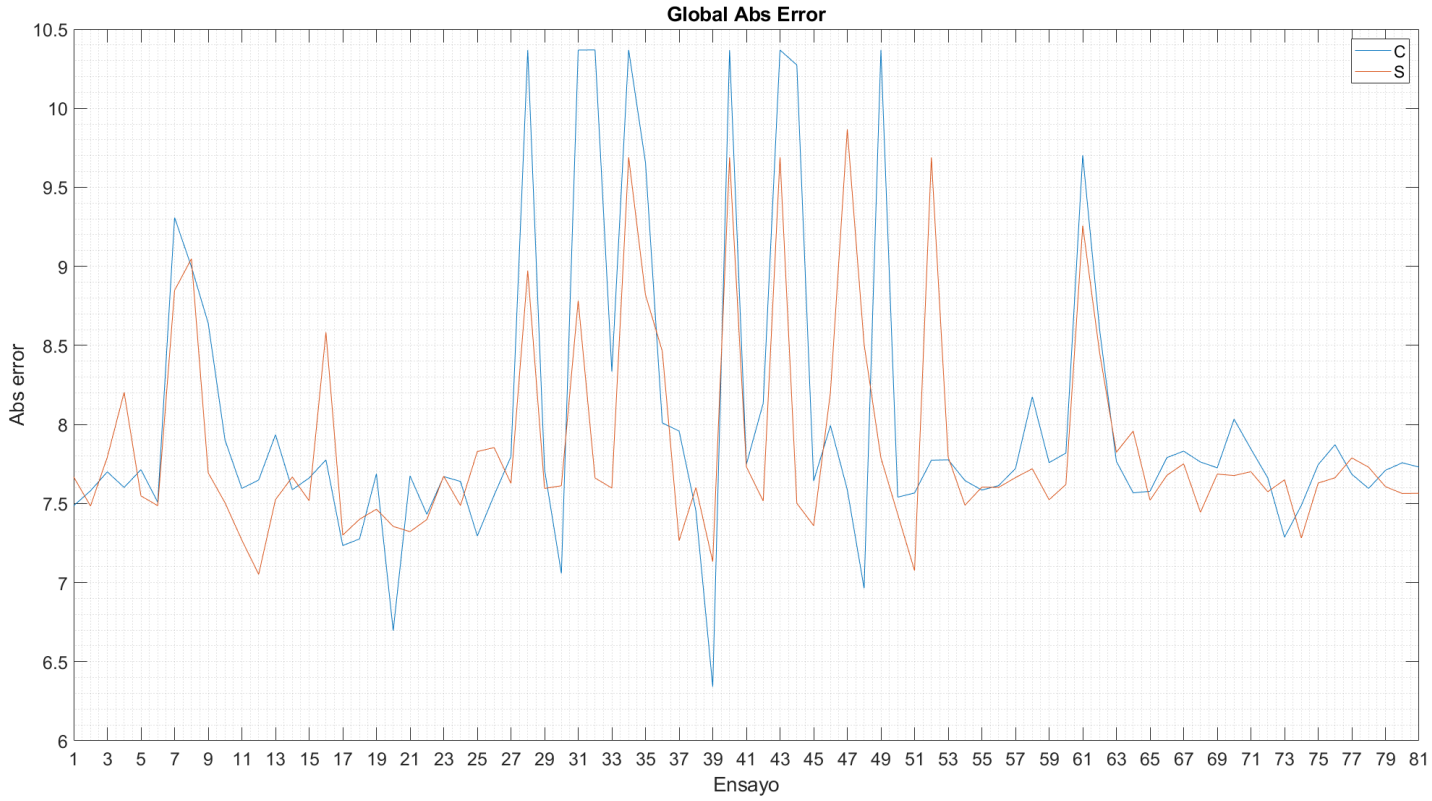

Figura D.1: Error absoluto global del análisis global de las redes *MLP*.

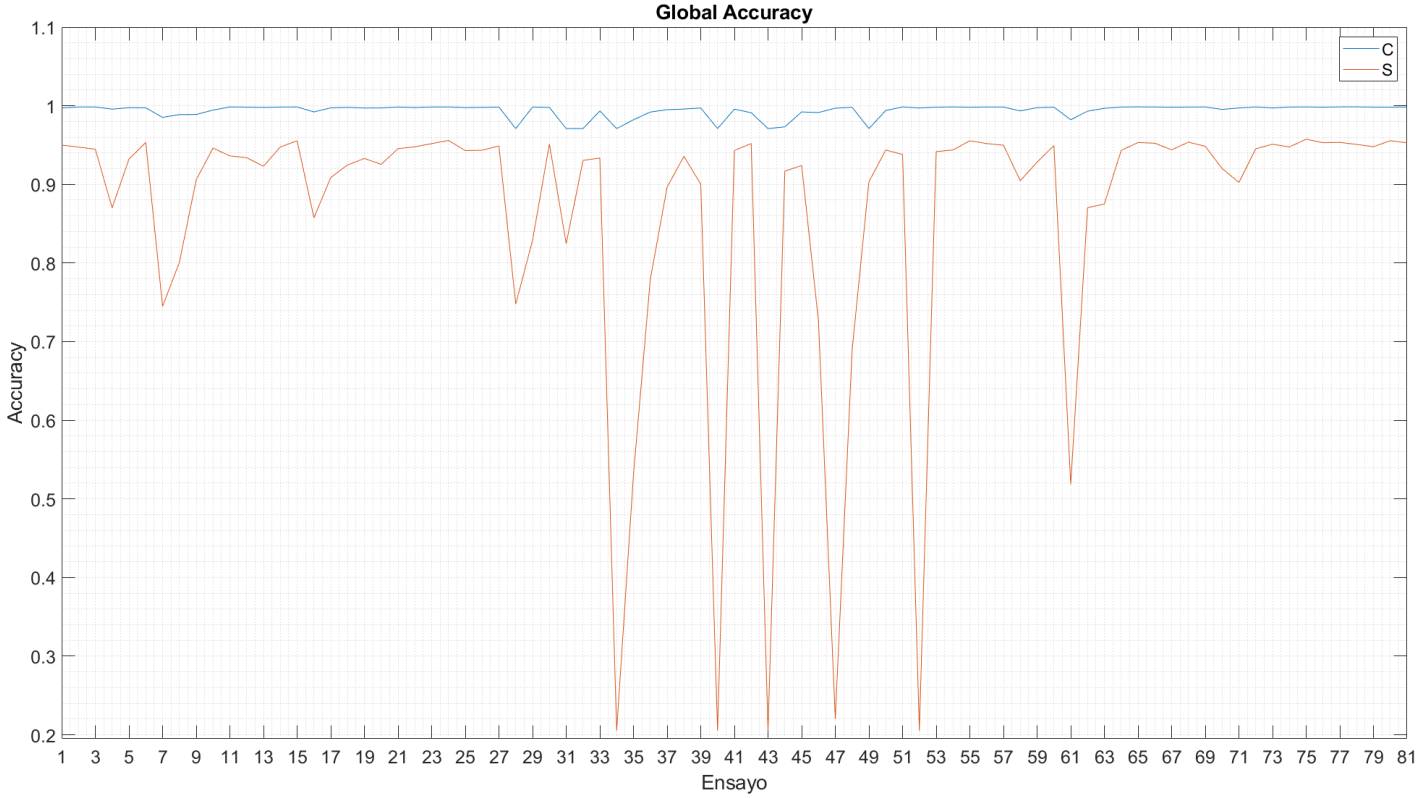

Figura D.2: *Accuracy* global del análisis global de las redes *MLP*.

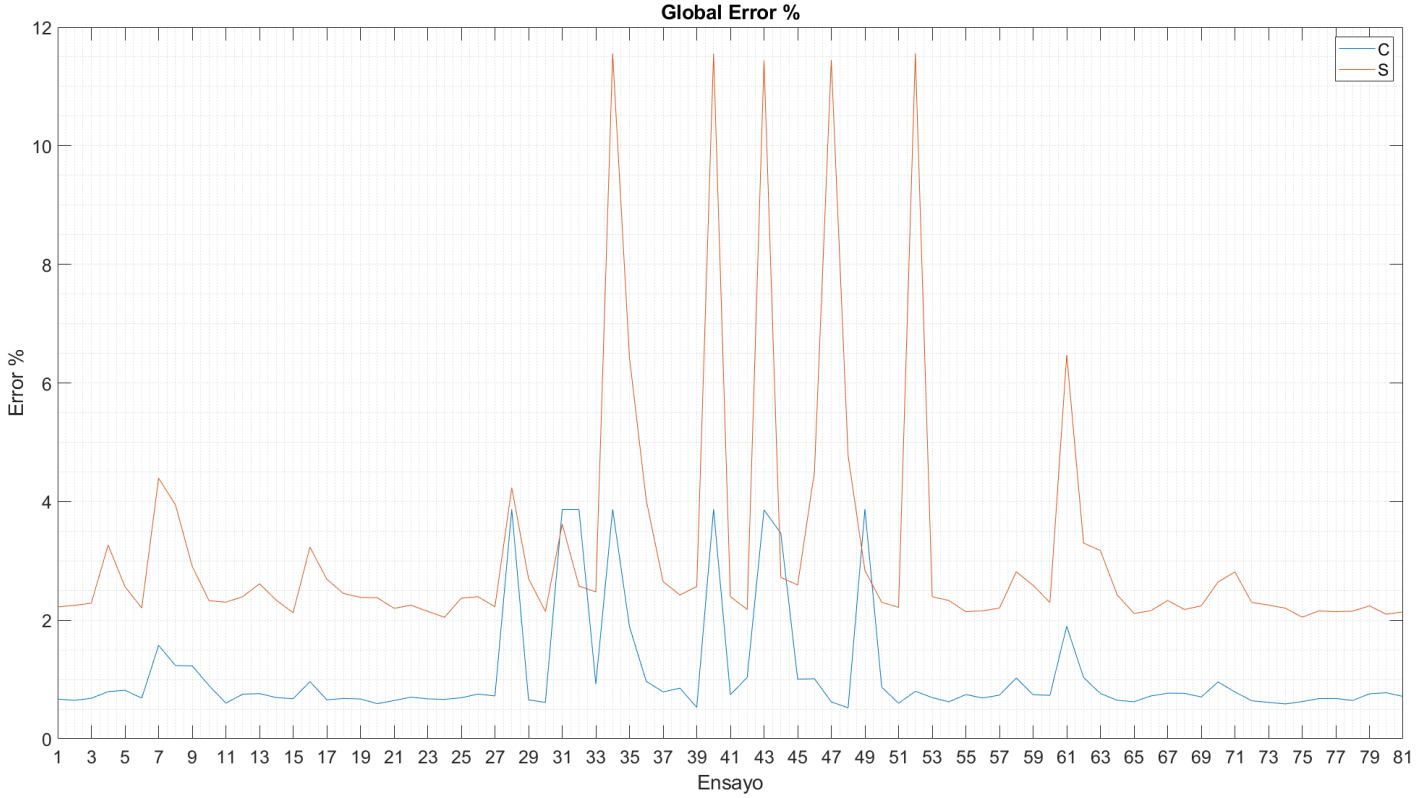

Figura D.3: Error porcentual global del análisis global de las redes *MLP*.

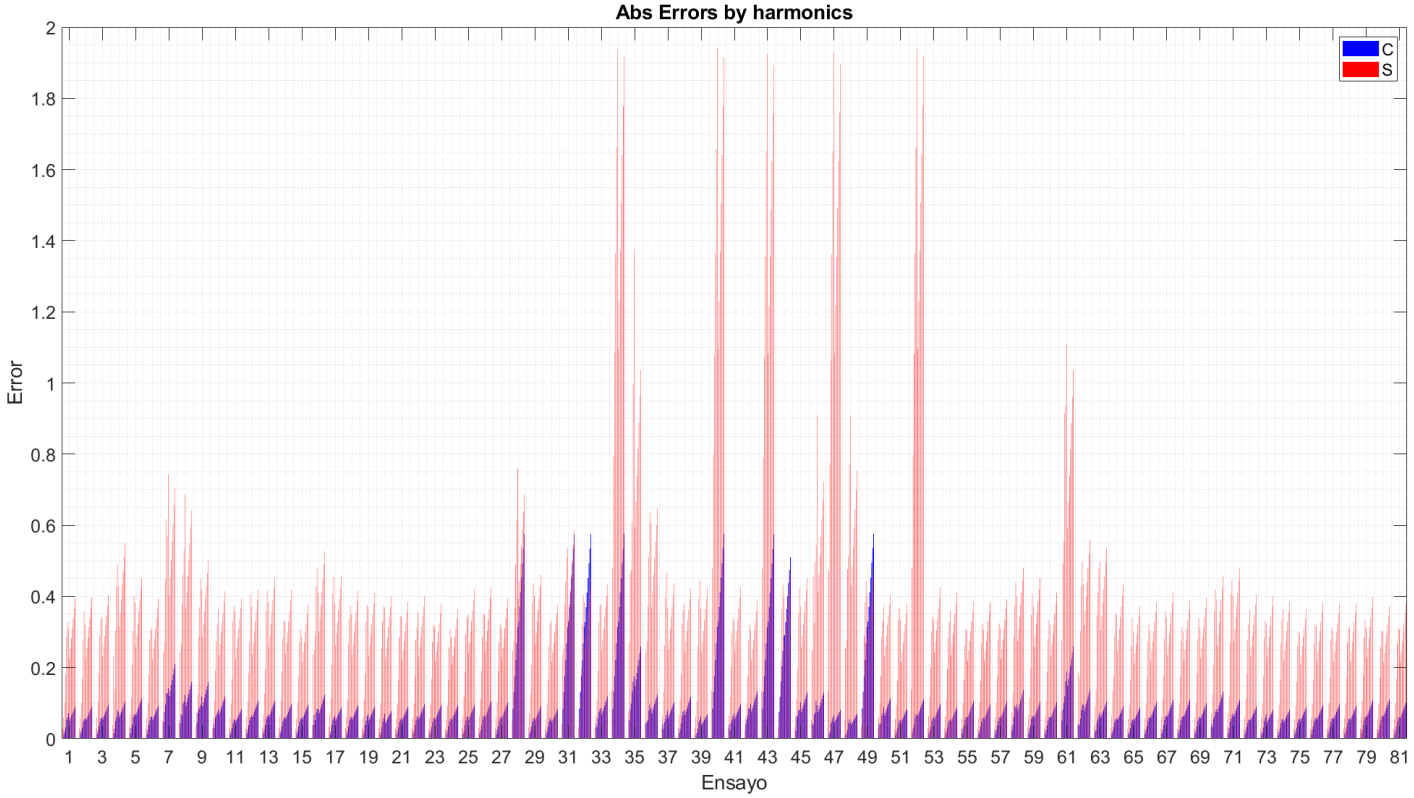

Figura D.4: Error absoluto por harmónicos del análisis global de las redes *MLP*.

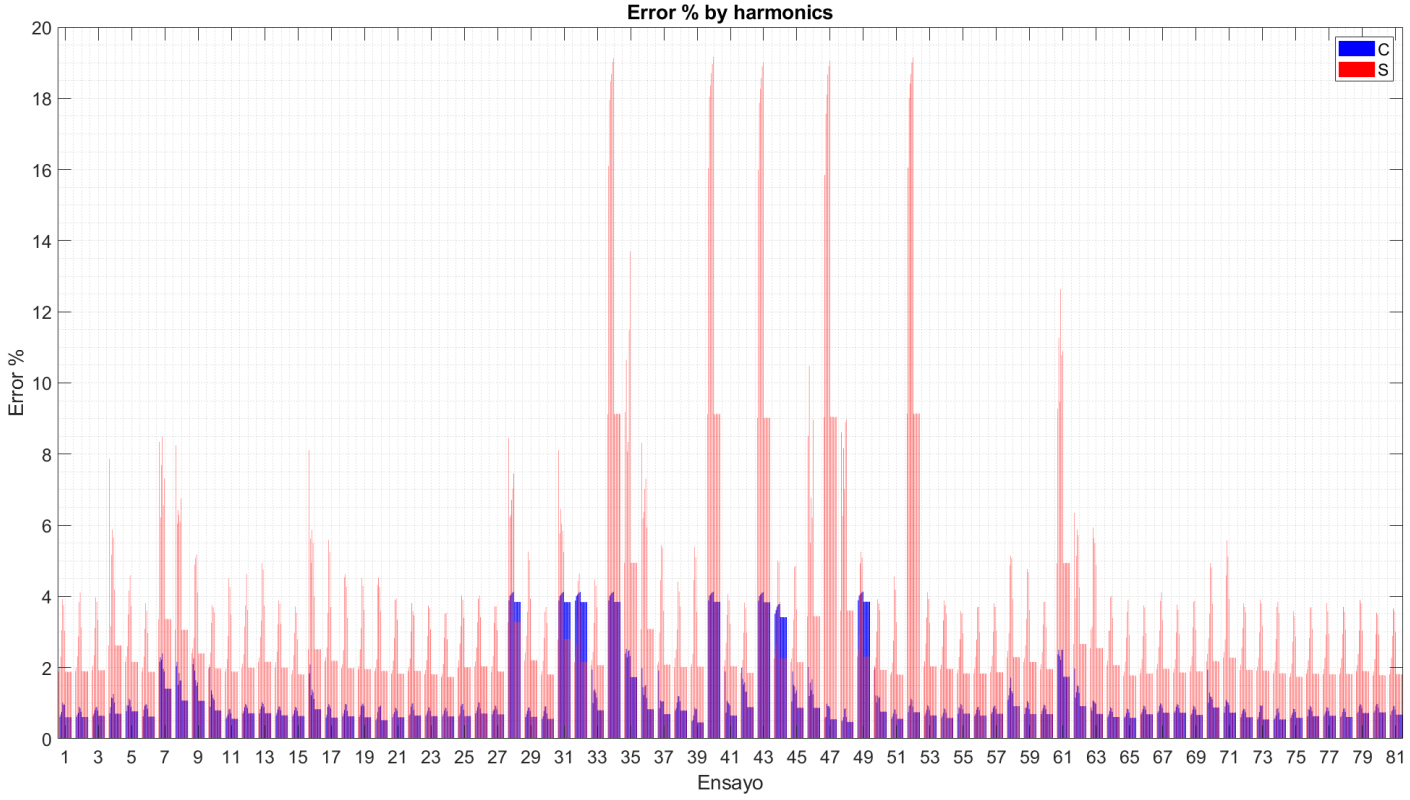

Figura D.5: Error porcentual por harmónicos del análisis global de las redes *MLP*.

Apéndice E

Métricas Análisis redes CNN

E.1. Métricas Num Filters

Volver al informe

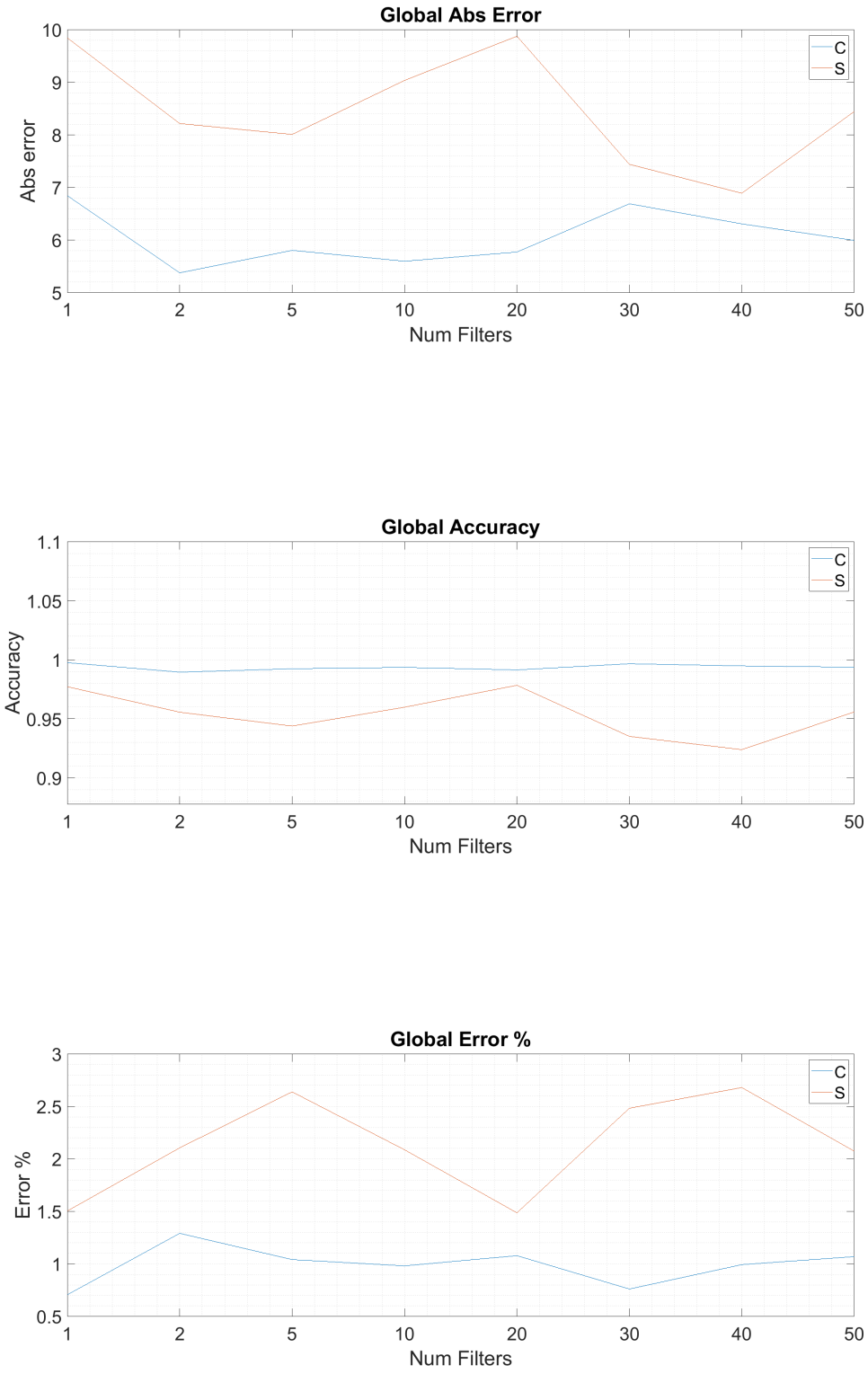

Figura E.1: Métricas numéricas globales del análisis *Num Filters*.

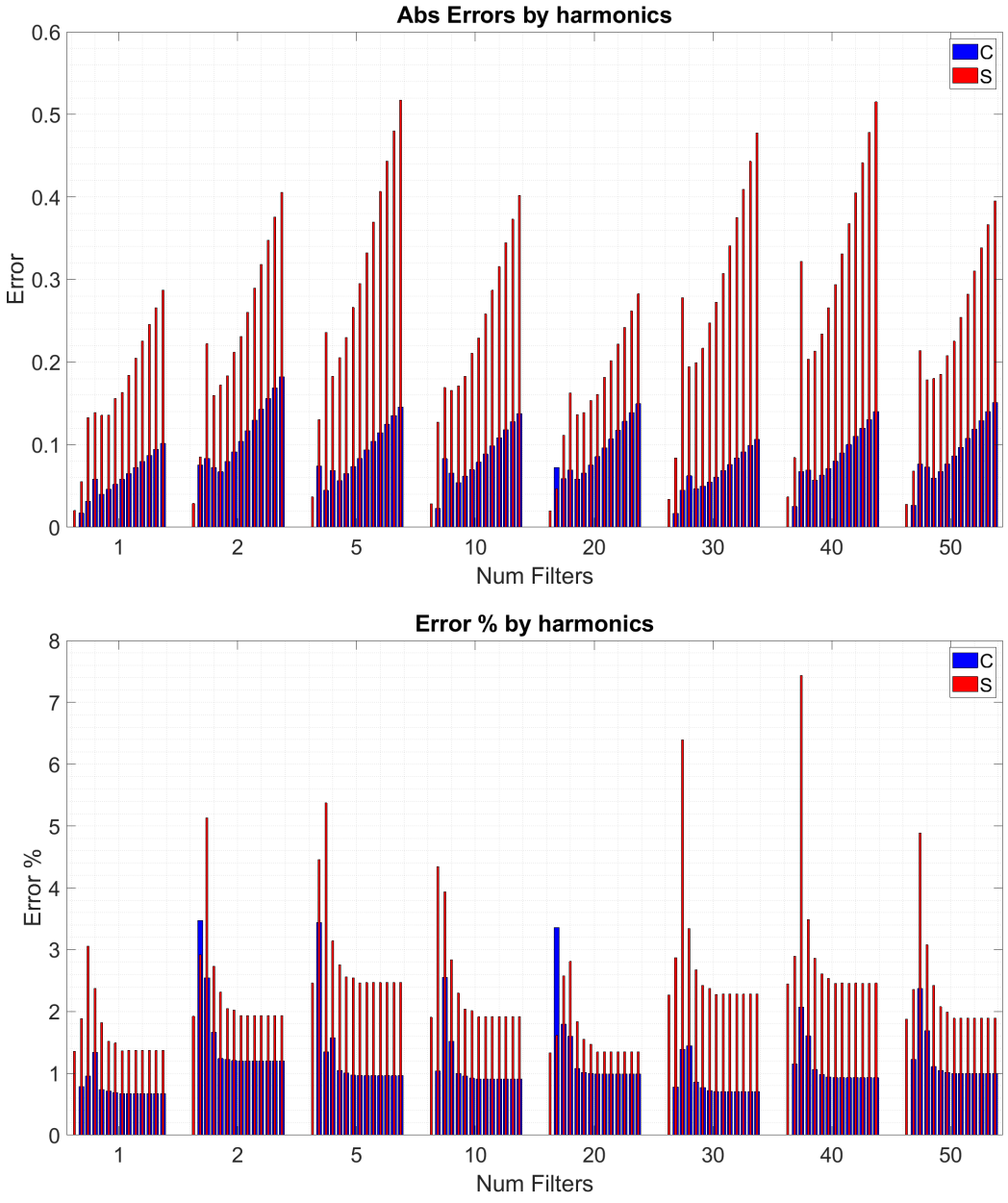

Figura E.2: Métricas numéricas por harmónicos del análisis *Num Filters*.

E.2. Métricas Filter Size

Volver al informe

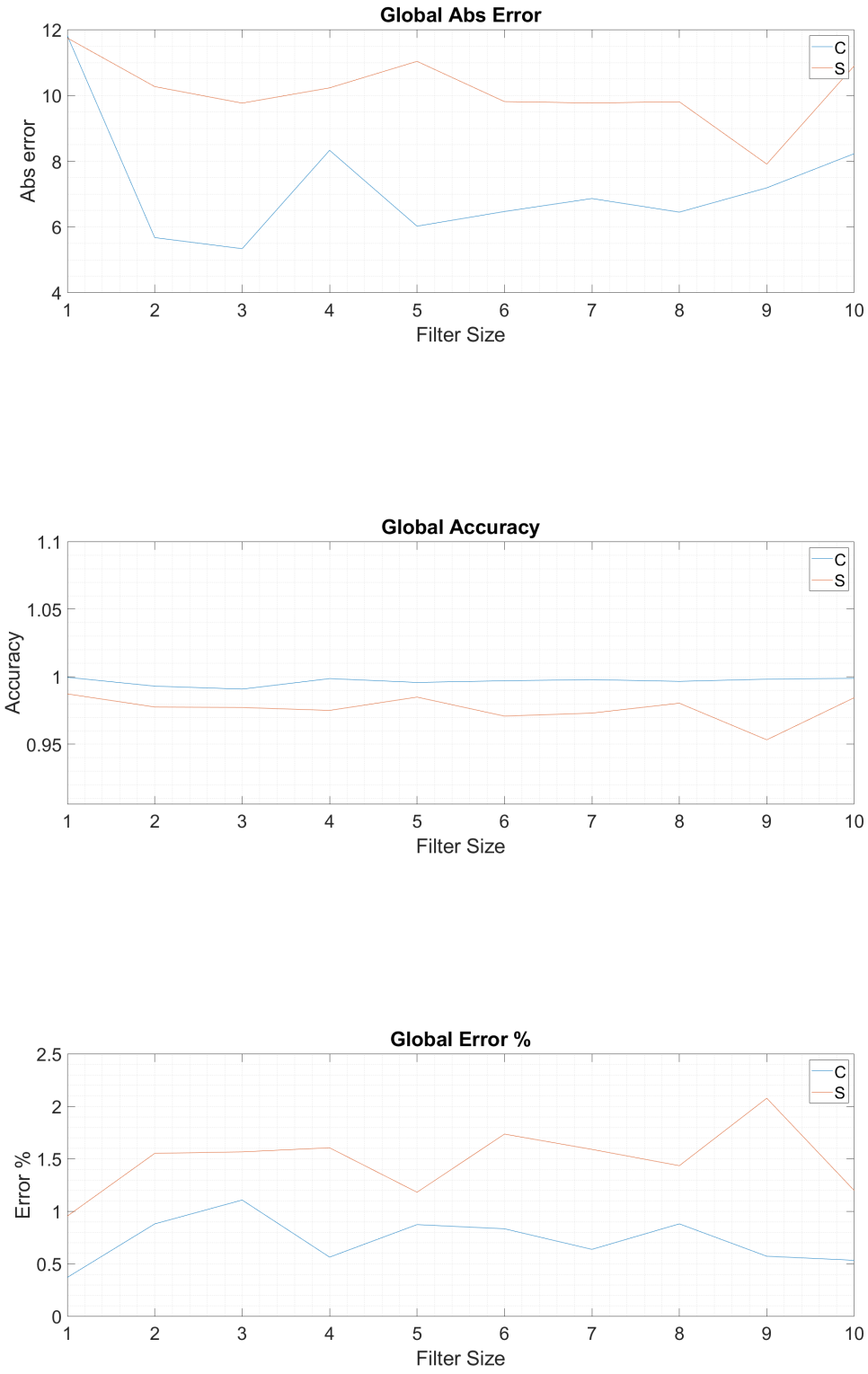

Figura E.3: Métricas numéricas globales del análisis *Filter Size*.

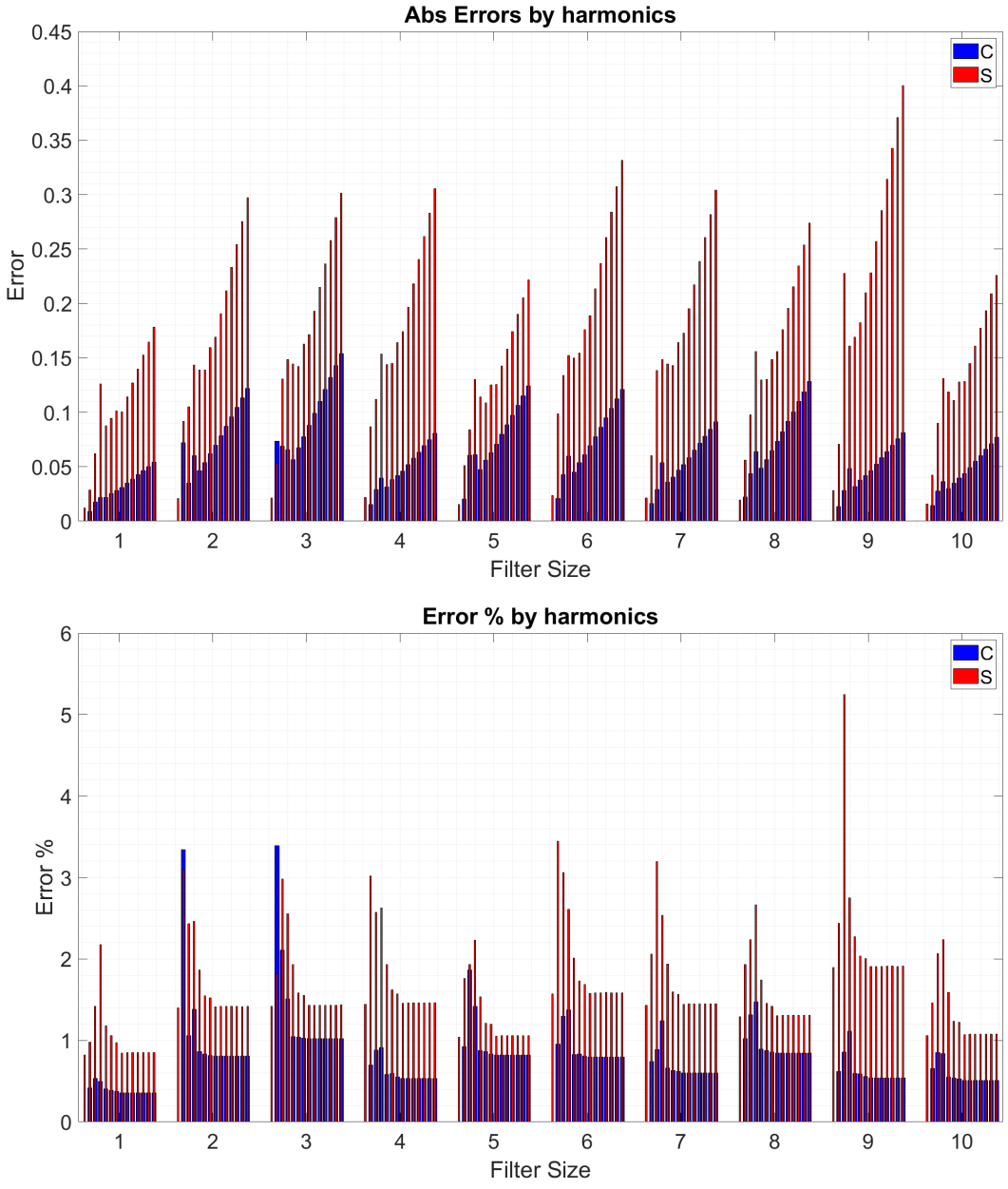

Figura E.4: Métricas numéricas por harmónicos del análisis *Filter Size*.

Apéndice F

Métricas Análisis Exploratorio

Ensayo	Neuronas 1ªCapa	Neuronas 2ªCapa	DF	ILR	DP	Epochs	Mini B.S	N.F	F.S
MLP 1	50	25	0.95	0.05	125	8000	128	-	-
MLP 2	50	25	0.95	0.05	125	12000	128	-	-
MLP 3	100	25	0.95	0.05	125	10000	128	-	-
MLP 4	50	50	0.95	0.05	125	10000	128	-	-
MLP 5	50	50	0.95	0.05	200	10000	128	-	-
MLP 6	50	50	0.95	0.05	125	10000	1024	-	-
CNN 1	50	25	0.95	0.05	25	400	256	3	5
CNN 2	50	25	0.95	0.05	12	400	256	3	5
CNN 3	50	25	0.95	0.05	8	400	256	1	5
CNN 4	50	25	0.85	0.05	12	400	256	3	5
CNN 5	50	25	0.95	0.01	12	400	256	3	5
CNN 6	50	25	0.95	0.05	12	400	512	3	5
CNN 7	50	25	0.95	0.05	8	750	512	3	5
CNN 8	50	25	0.98	0.03	5	800	512	3	5
CNN 9	50	25	0.95	0.05	8	750	512	5	5
CNN 10	50	25	0.95	0.05	8	750	256	3	5
CNN 11	50	25	0.95	0.05	8	750	1024	3	5

Tabla F.1: Ensayos exploratorios de las redes *MLP* y *CNN*.

Volver al informe

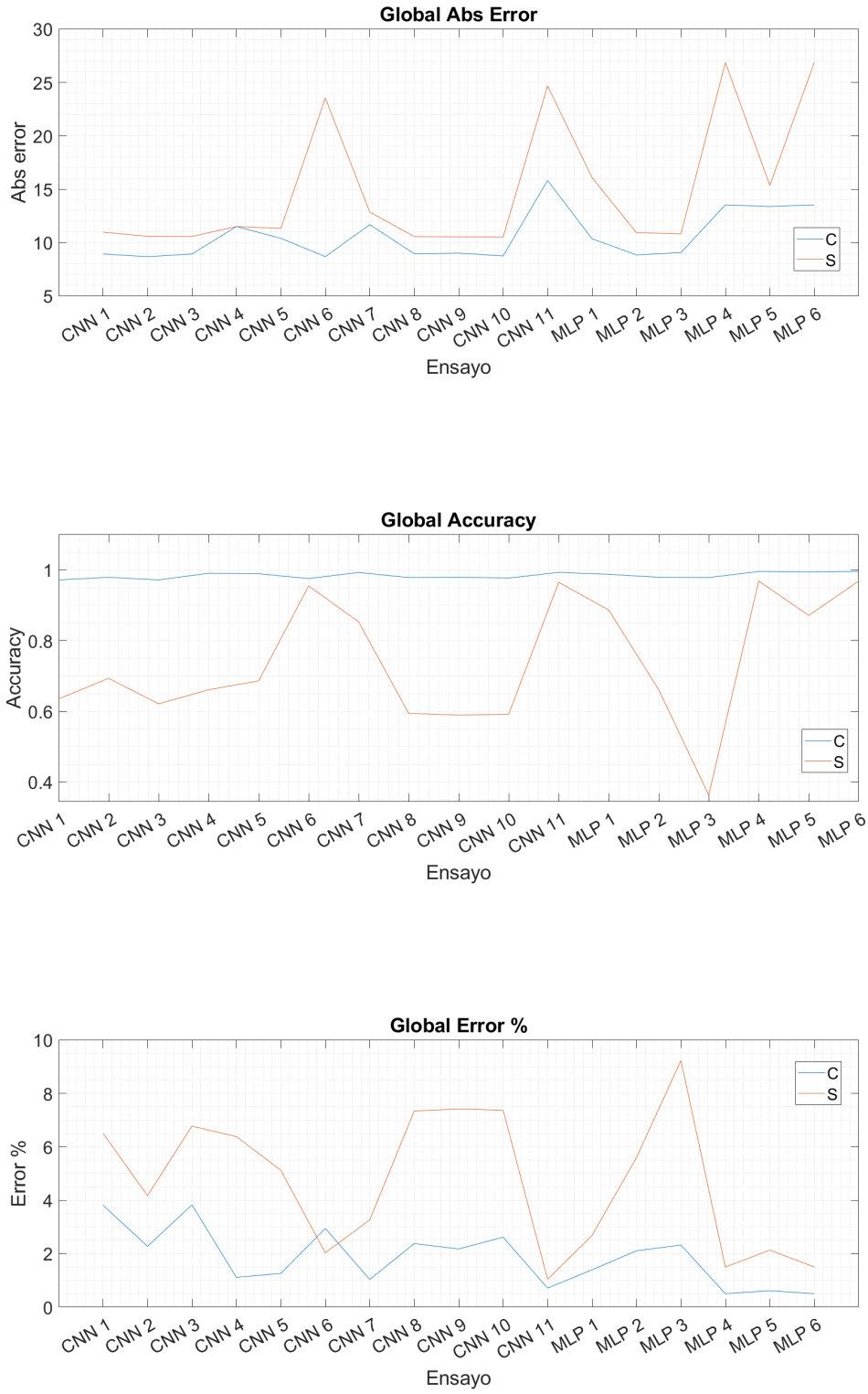

Figura F.1: Métricas numéricas globales del análisis exploratorio.

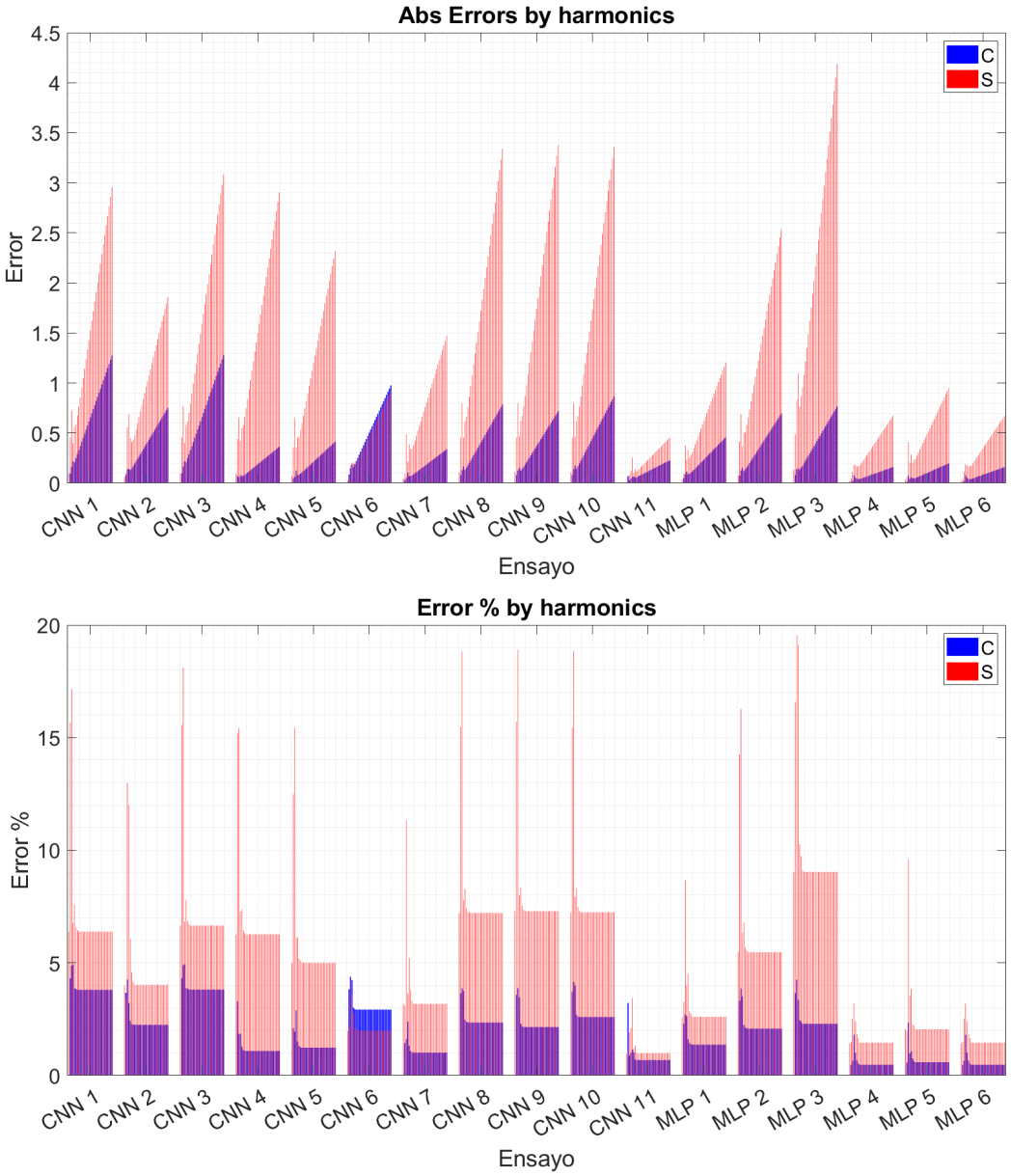

Figura F.2: Métricas numéricas por harmónicos del análisis exploratorio.

Bibliografía

[1] NASA. Asteroids. https://solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/overview/?page=0&per_page=40&order=name+asc&search=&condition_1=101.

[2] M. O'Neill. Dinosaur-killing asteroid struck earth at deadliest possible angle. https://www.livescience.com/dinosaur-killing-asteroid-struck-earth, 2022.

[3] T.D. Cole, M.T. Boies, A.S. El-Dinary, and et al. The near-earth asteroid rendezvous laser altimeter. https://doi.org/10.1023/A:1005056828065, 1997.

[4] Viorel Badescu. *Asteroids: Prospective Energy and Material Resources*. Springer-Verlag Berlin Heidelberg, 2013.

[5] J.K. Miller, A.S. Konopliv, P.G. Antreasian, J.J. Bordi, S. Chesley, C.E. Helfrich, W.M. Owen, T.C. Wang, B.G. Williams, D.K. Yeomans, and D.J. Scheeres. Determination of shape, gravity, and rotational state of asteroid 433 eros. *Icarus*, 155(1):3–17, 2002.

[6] Isaac Newton. *Philosophiae naturalis principia mathematica*. Royal Society, 1687.

[7] Howard Curtis. *Orbital Mechanics for Engineering Students*. Elvesier Aerospace Engineering Series, 2005.

[8] G.W. Hill. Researches in the lunar theory. *American Journal of Mathematics*, 1878.

[9] Wikimedia Commons. File:latitude and longitude graticule on a sphere.svg. https://commons.wikimedia.org/w/index.php?title=File:Latitude_and_longitude_graticule_on_a_sphere.svg&oldid=734437327, 2023. [Online; accessed 14-December-2022].

[10] William M. Kaula. *Theory of satellite geodesy. Applications of satellites to geodesy.* Blaisdell Publishing Company, 1966.

[11] Adrien-Marie Legendre. Recherches sur l'attraction des sphéroïdes homogènes. *Mémoires de Mathématiques et de Physique, présentés à l'Académie Royale des Sciences, par divers savans, et lus dans ses Assemblées*, 1782.

[12] J. E. J. Physical geodesy. by W. A. Heiskanen and H. Moritz. pp. ix 364, and numerous diagrams. W. H. Freeman and Co., San Francisco and London, 1967. Price 100s. *Geological Magazine*, 104(3):302–302, 1967.

[13] B. Jones. *Efficient Models for the Evaluation and Estimation of the Gravity Field.* University of Colorado Boulder, 2010.

[14] Russel Herman. *Introduction to partial differential equations.* University of North Carolina Wilmington, 2015.

[15] W.O. Amrein, A.M. Hinz, and D.B. Pearson. *Sturm-Liouville Theory: Past and Present.* Birkhäuser Basel, 2005.

[16] W. N. Bailey. *The Theory of Spherical and Ellipsoidal Harmonics. By E. W. Hobson. Pp. xi, 500. 37s. 6d. net. 1931. (Cambridge University Press)*, volume 16. Cambridge University Press, 1932.

[17] George Arfken. *Mathematical Methods for Physicists.* Academic Press, Inc., San Diego, third edition, 1985.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[19] Feng hsiung Hsu. *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion.* Princeton University Press, 2022.

[20] Tom M Mitchell et al. *Machine learning.* McGraw-hill New York, 1997.

[21] Aayush Bajaj. Performance metrics in machine learning [complete guide]. `https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide`.

[22] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[23] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning with applications in r: by gareth james, daniela witten, trevor hastie, and robert tibshirani, new york, springer science and business media, 2013, eisbn: 978-1-4614-7137-7. *Statistical Theory and Related Fields*, pages 1–1, 09 2021.

[24] B. Illowsky and S.L. Dean. *Introductory Statistics*. Open Textbook Library. OpenStax, Rice University, 2013.

[25] Terence Shin. Basic statistics you need to know for data science. `https://towardsdatascience.com/basic-statistics-you-need-to-know-for-data-science-1fdd290f59b5`, 2020.

[26] Rajesh Sharma. SIGGRAPH Think Beyond. In *Hands-on Workshop: Machine Learning and Neural Networks*, 2020.

[27] Wikimedia Commons. File:artificial neural network.svg. `https://commons.wikimedia.org/w/index.php?title=File:Artificial_neural_network.svg&oldid=494529927`, 2020. [Online; accessed 22-December-2022].

[28] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15, 01 2010.

[29] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *American Journal of Psychology*, 76:705, 1963.

[30] Robert Keim. How to train a basic perceptron neural network. `https://www.allaboutcircuits.com/technical-articles/how-to-train-a-basic-perceptron-neural-network`, November 2019.

[31] Filippo Minutella. Gnns - practical difficulties and applications. `https://medium.com/larus-team/gnns-practical-difficulties-and-applications-766262298421`, 2022.

[32] P.D. Wasserman and T. Schwartz. Neural networks. ii. what are they and why is everybody so interested in them now? *IEEE Expert*, 3(1):10–15, 1988.

[33] Offisong Emmanuel. The magic of word embedding in keras. https://medium.com/@emmanueloffisong2002/the-magic-of-word-embedding-in-keras-ff5afa98098c, 2022.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[35] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2018.

[36] IBM Cloud Education. Convolutional neural networks. https://ibm.com/cloud/learn/convolutional-neural-networks, 2020.

[37] NASA. 433 eros. https://solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/433-eros/in-depth, 2021.

[38] Alan Hale. Special topic: Asteroid (433) eros. https://www.rocketstem.org/2020/01/18/ice-and-stone-special-topic-04/, 2020.

[39] Planetary Science Institute. Near collected shape and gravity models. https://sbn.psi.edu/pds/resource/nearbrowse.html, 2001.

[40] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

[41] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models, 2013.

[42] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2015.

[43] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

Índice de figuras

Índice de tablas