UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Generación de prediseños estructurales de retículas topológicamente optimizadas mediante el uso de inteligencia artificial

Trabajo Fin de Grado

Grado en Ingeniería Aeroespacial

AUTOR/A: Saleh Walie, Ahmad

Tutor/a: Navarro Jiménez, José Manuel

Cotutor/a: Martínez Martínez, Antolín

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Generación de prediseños estructurales de retículas topológicamente optimizadas mediante el uso de inteligencia artificial

Trabajo Fin de Grado

Grado en Ingeniería Aeroespacial

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Autor:

Ahmad Saleh Walie

Tutores:

José Manuel Navarro Jiménez

Antolín Martínez Martínez

VALENCIA, JULIO de 2023

Resumen

El uso de estructuras reticulares está siendo de gran interés en la actualidad en el sector aeroespacial, ya que proveen gran ligereza y rigidez permitiendo, además, absorber cargas con cierta variabilidad respecto de las de diseño, como ocurre por ejemplo en huesos. La obtención de estas estructuras mediante técnicas de optimización convencionales tiene un coste computacional elevado ya que requiere una gran resolución de manera que, si se aborda de manera directa, exigiría el uso de supercomputadores. Por ello, se plantea el uso de retículas, pasando de un problema de optimización global a uno local en cada retícula. En otras palabras, cada una de las retículas se convierte en un problema de optimización individual. El objetivo es la obtención de estructuras reticulares donde cada retícula está optimizada según las cargas a las que está sometida. A pesar de tener un problema de optimización local de tamaño reducido, su resolución para todas las retículas del componente a diseñar sigue siendo excesivamente costosa si se utilizan las técnicas clásicas de optimización topológica. Por ejemplo, el método *Solid Isotropic Material Penalization*, SIMP, es el más utilizado de los métodos de optimización basados en densidad y cada una de sus iteraciones requiere la resolución del problema elástico mediante el método de los elementos finitos (MEF). Por ello, en este trabajo se van a utilizar técnicas existentes de inteligencia artificial, como las redes neuronales convolucionales (CNN) y "*long short-term memory*" (LSTM) para acelerar este proceso de optimización, obteniendo prediseños estructuralmente optimizados de cada retícula con un menor coste computacional.

Resum

L'ús d'estructures reticulars està sent de gran interés en l'actualitat en el sector aeroespacial, ja que proveeixen gran lleugeresa i rigidesa permetent, a més, absorbir càrregues amb una certa variabilitat respecte de les de disseny, com ocorre per exemple en ossos. L'obtenció d'aquestes estructures mitjançant tècniques d'optimització convencionals té un cost computacional elevat ja que requereix una gran resolució de manera que, si s'aborda de manera directa, exigiria l'ús de supercomputadors. Per això, es planteja l'ús de reticles, passant d'un problema d'optimització global a un local en cada reticle. En altres paraules, cadascuna dels reticles es converteix en un problema d'optimització individual. L'objectiu és l'obtenció d'estructures reticulars on cada reticle està optimitzat segons les càrregues a les quals està sotmesa. Malgrat tindre un problema d'optimització local de grandària reduïda, la seua resolució per a tots els reticles del component a dissenyar continua sent excessivament costosa si s'utilitzen les tècniques clàssiques d'optimització topològica. Per exemple, el mètode *Solid Isotropic Material Penalization*, SIMP, és el més utilitzat dels mètodes d'optimització basats en densitat i cadascuna de les seues iteracions requereix la resolució del problema elàstic mitjançant el mètode dels elements finits (MEF). Per això, en aquest treball es faran servir tècniques existents d'intel·ligència artificial, com les xarxes neuronals convolucionals (CNN) i "*long short-term memory*" (LSTM) per a accelerar aquest procés d'optimització, obtenint predissenys estructuralment optimitzats de cada reticle amb un menor cost computacional.

Abstract

The use of lattice structures is currently of great interest in the aerospace sector since they provide great lightness and stiffness, allowing, in addition, to absorb loads with variability to the design loads, as occurs, for example, in bones. Obtaining these structures utilizing conventional optimization techniques has a high computational cost since it requires a high resolution that, if approached directly, would require supercomputers. Therefore, the use of grids is proposed, going from a global optimization problem to a local one in each grid. In other words, each of the grids becomes an individual optimization problem. The objective is to obtain lattice structures where each lattice is optimized according to the loads to which it is subjected. Despite having a local optimization problem of reduced size, its resolution for all the lattices of the component to be designed is still excessively expensive if classical topological optimization techniques are used. For example, the SIMP method is the most widely used of the density-based optimization methods, and each of its iterations requires the resolution of the elastic problem using the finite element method (FEM). Therefore, in this work, existing artificial intelligence techniques, such as convolutional neural networks (CNN) and long short-term memory (LSTM), will be used to accelerate this optimization process, obtaining structurally optimized pre-designs of each grid with a lower computational cost.

Índice

Índice de Tablas

Índice de Figuras

Siglas

CNN *Convolutional Neural Network* - Red Neuronal Convolucional.

DNN *Dense Neural Network* - Redes Neuronales Densas.

IA Inteligencia Artificial.

LSTM *Long short-term memory*.

MAE *Mean Absolute Error* - Error Absoluto Medio.

MEF Método de los Elementos Finitos.

ML *Machine learning* - Aprendizaje Automático.

MSE *Mean Squared Error* - Error Cuadrático Medio.

MSLE *Mean Squared Logarithmic Error Loss*.

NN *Neural Network* - Redes Neuronales.

RNN *Recurrent Neural Network* - Redes Neuronales Recurrentes.

SIMP *Solid Isotropic Material Penalization* - Método de Material Isotrópico Sólido con Penalización.

Parte I

Memoria

1 | Introducción

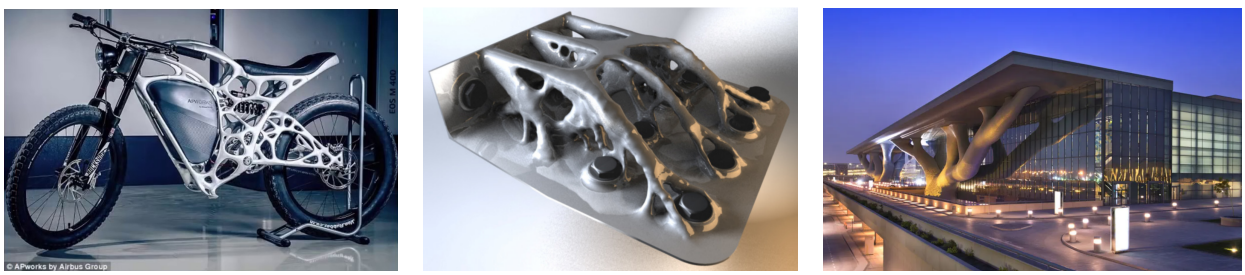
El concepto de «optimización» y el de ingeniería han estado siempre intrínsecamente vinculados. El objetivo de cualquier proceso de optimización se puede formular como la minimización de una función dada, cumpliendo a su vez con una serie de restricciones. La herramienta de optimización se encuentra presente en diversos ámbitos como son los de mejorar la productividad y eficiencia en los proceso de producción, minimizar el coste relacionado con la gestión de recursos, o el diseño o modificación de estructuras que satisfagan unos objetivos dados, entre otros [1].

La etapa de diseño de un proyecto de ingeniería tradicionalmente ha sido un proceso de prueba y error; se propone un diseño inicial a partir del conocimiento previo, la creatividad, la normativa y la *intuición* del diseñador, y se realiza un análisis para evaluar tanto el rendimiento como el cumplimiento de requisitos. A partir de los resultados de este análisis, se proponen modificaciones del diseño original, se repite el proceso de análisis, y así sucesivamente, hasta que no se puede efectuar ninguna mejora adicional [2].

El gran desarrollo de los ordenadores, y el consecuente abaratamiento de la potencia de cálculo ha posibilitado revolucionar el proceso de diseño tradicional mediante la introducción de la herramienta de optimización en la etapa de diseño de los distintos campos de la ingeniería. Uno de los primeros ámbitos donde se introdujeron los procesos de optimización fue en la ingeniería estructural, [3]. No es para menos, en el sector aeroespacial el objetivo principal de la fase de diseño de los componentes con función estructural es la reducción del peso sin causar un detrimento en las propiedades mecánicas.

De acuerdo con Olhoff *et al.* [4], el proceso de optimización estructural tiene como objetivo encontrar la mejor estructura posible (en cuanto a peso, rigidez, o deflexión máxima...) cumpliendo con restricciones físicas o geométricas de cualquier tipo, de materiales, del proceso de fabricación, de imposición de peso mínimo o de tensión máxima, entre otras.

Se pueden seleccionar tres ejemplos representativos en la literatura de la aplicación de la técnica de optimización estructural; por una parte la colaboración entre Airbus y APworks en el diseño de un chasis de moto [5], la creación de un soporte aeroespacial [6] y los soportes del palacio de exposiciones y congresos de Catar [7], mostrados en la Figura 1.1.



(a) Chasis Light-Rider (b) Soporte aeroespacial (c) Qatar Convention Center

Figura 1.1: Ejemplos representativos optimización estructural.

1.1. Optimización topológica

Según Zhu *et al.* [8] la optimización topológica es un tipo de optimización estructural que se puede definir como una herramienta matemática que permite distribuir de manera óptima el material dentro de un dominio de trabajo, cumpliendo con una serie de restricciones. Esta distribución de material puede ser arbitraria, al contrario que otras técnicas que se basan en modificar topologías ya existentes [9]. El resultado obtenido mediante un proceso de optimización topológica, tanto en cuanto cumple las restricciones impuestas, es desconocida *a priori*, permitiendo un alto grado de versatilidad.

Una ilustración típica que pone en relieve el funcionamiento de la técnica de optimización topológica es la de optimización de un domino de trabajo rectangular al que se aplica una carga vertical con ciertas condiciones de contorno en la base, tal y como se muestra en la Figura 1.2. A la vista del problema, mediante el método es posible colocar el material de forma óptima, dando lugar a una estructura de barras.

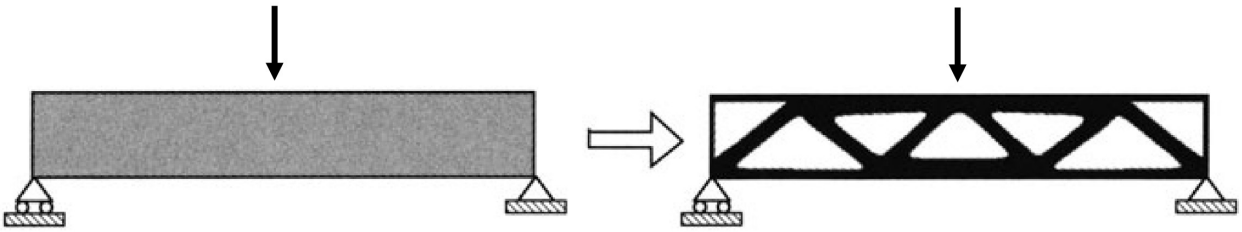


Figura 1.2: Estructura biapoyada óptima ante carga vertical calculada modificando topología, [10].

Es importante señalar que tanto el dominio de trabajo, Ω, como las condiciones de contorno y las condiciones de carga son definidos antes de comenzar el proceso de optimización topológica.

Existen programas comerciales que implementan algoritmos de optimización estructural que permiten al diseñador hacer uso de estas técnicas, como Abaqus [11] o Altair [12], es este último, de hecho, el que se utilizó para optimizar topológicamente las costillas del borde de ataque del Airbus A380 como se aprecia en la Figura 1.3, [13].

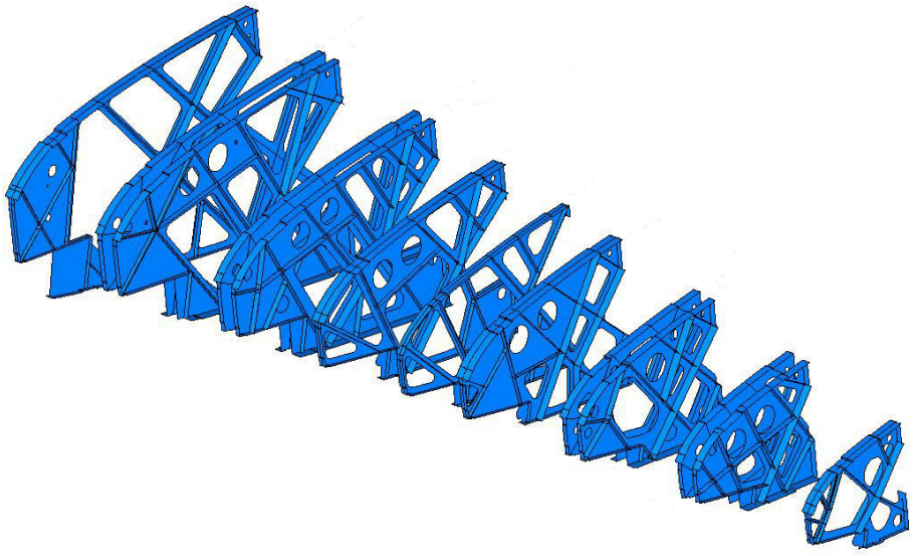


Figura 1.3: Costillas del borde de ataque del semiala izquierda del A380.

El método de optimización topológica más popular, por su simplicidad, es el método *Solid Isotropic Material Penalization*, SIMP. Este método permite maximizar la rigidez de un componente dadas unas condiciones de contorno, unas condiciones de carga, y una fracción de volumen. Para configurar el método, se requiere definir una región de diseño dentro de la cual se lleva a cabo el proceso de optimización (ver Figura 1.4). Este dominio es discretizado en distintas celdas que, por la naturaleza del método, tendrá como solución valores de densidad relativa entre 0 y 1. De este modo, el método de optimización propone como solución al problema el valor óptimo de densidad para cada una de las celdas de discretización.

Es importante adelantar que el método SIMP hace uso a nivel interno del Método de los Elementos finitos, MEF, que es un algoritmo que requiere resolver la inversa de una matriz que tiene como dimensión el número de elementos de la discretización, y que por tanto tiene un coste computacional creciente.

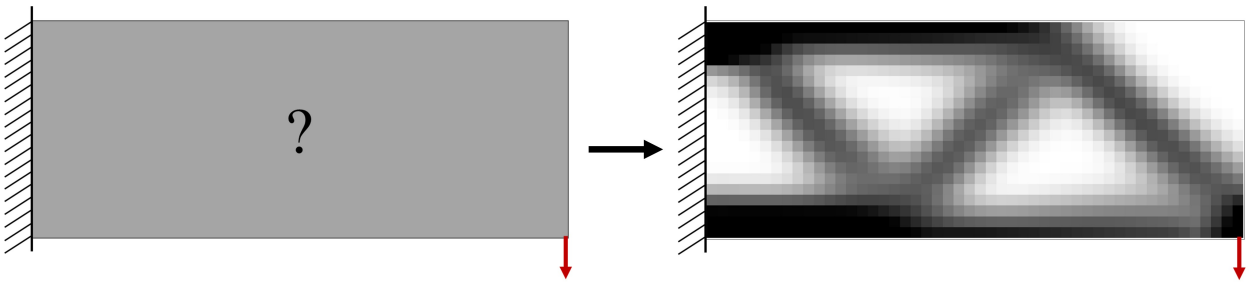


Figura 1.4: Optimización topológica dominio empotrado con carga puntual en extremo inferior derecha, con fracción de material de 50 % del volumen.

1.2. Optimización multinivel

El proceso de optimización topológica expuesto hasta ahora aborda el problema en una única escala (ver Figuras 1.2 y 1.4). En el año 2016 Oliver *et al.* propusieron un nuevo paradigma de optimización: la optimización multinivel [14]. Una de las estrategias para llevar a cabo la optimización multinivel consiste, en primer lugar, optimizar topológicamente el dominio de la escala macro, tal y como se muestra en la Figura 1.5, y tras esto, optimizar topológicamente **cada una** de las celdas que conforman esta la escala macro.

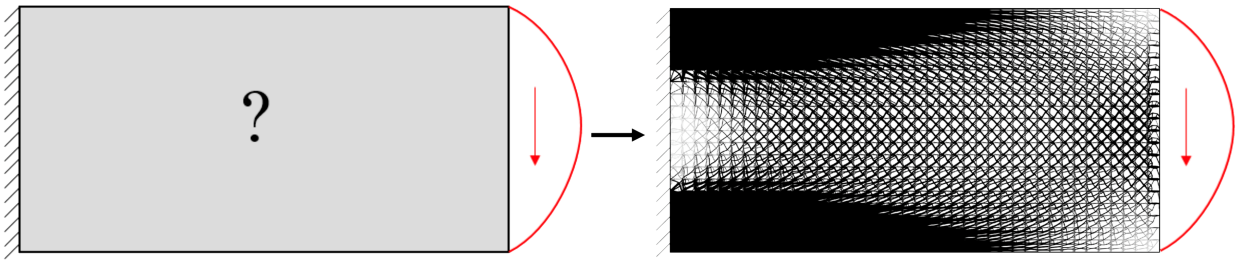


Figura 1.5: Problema con borde empotrado y carga cortante en extremo.

Es esencial señalar que las condiciones de contorno, la fracción de volumen, y las restricciones sobre el dominio de cada celda se obtienen de la solución de la escala macro. De este modo, se pueden resolver los dos problemas de manera desacoplada. Si se siguen los dos pasos descritos, se puede generar la estructura reticular que se muestra en la Figura 1.6.

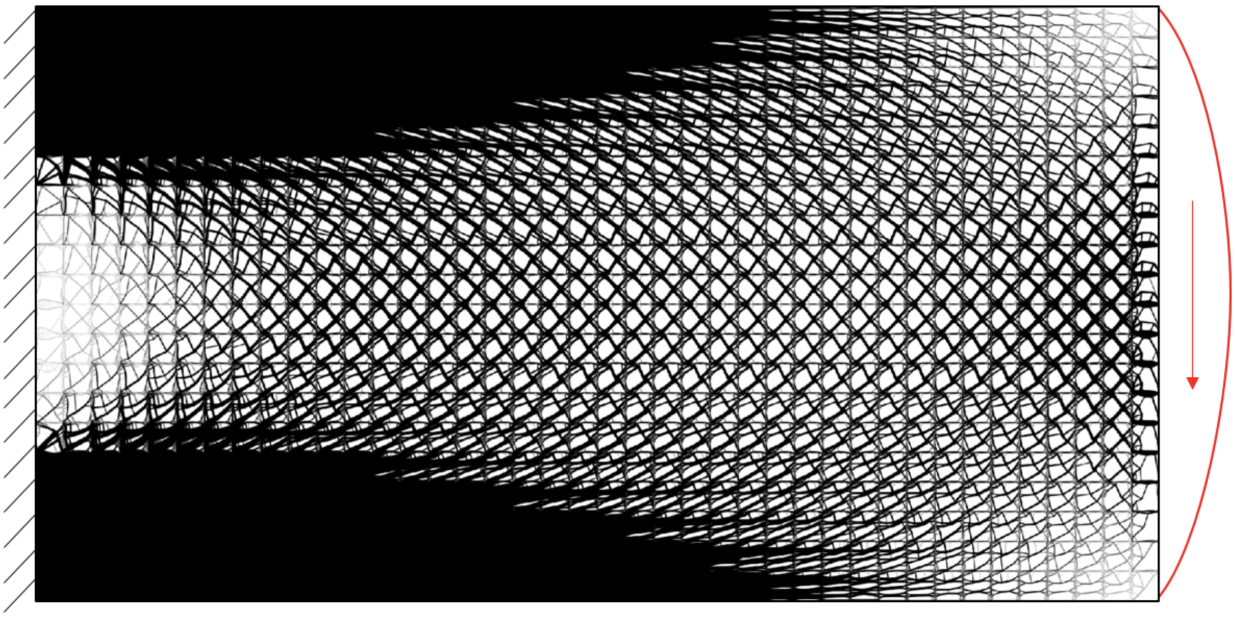


Figura 1.6: Solución del problema multinivel con estructura de carácter reticular, [15].

Según Wu *et al.* [16] este tipo de estructuras tienen multitud de ventajas, entre las cuales se encuentran las siguientes: son estructuras tolerantes al daño puesto que mantienen la rigidez aunque algunas de las retículas se rompan, son robustas ante cambios sensibles de carga, y son estructuras ligeras. Además, la popularización de la tecnología de fabricación aditiva facilita la creación de este tipo de estructuras.

Sin embargo, desde el punto de vista del cálculo, existe una gran desventaja. Al contrario de la optimización topológica en una sola escala, en un problema multinivel se ha de optimizar la estructura no solo desde el punto de vista del problema macro, sino que se ha de resolver también cada retícula. Aunque las retículas suelen ser pequeñas, y el coste computacional de resolver cada una no es muy importante, hay una gran cantidad de ellas.

El propio artículo de Oliver *et al.* propone la creación de una especie de prontuario en el que se guarden las retículas optimizadas junto con sus condiciones de contorno y de carga [14]. De esta manera, se puede recuperar la retícula guardada para otros problemas cuando las condiciones de contorno y las cargas sean las mismas. Este enfoque tiene el problema de que la variabilidad de condiciones es extremadamente elevada, y tendría un coste muy elevado, en términos de memoria, guardar la solución de cada *set* de condiciones de entrada sin realizar aproximaciones.

En este Trabajo se articula una estrategia para reducir el coste computacional de la etapa de optimización de las retículas mediante el uso de modelos de inteligencia artificial, IA.

1.2.1. Aprendizaje automático

La IA es un campo de la informática que tiene como objetivo crear máquinas o programas informáticos capaces de realizar tareas como el aprendizaje, el razonamiento, y la resolución de problemas, [17, 18]. Uno de sus campos de estudio es el que se encarga de desarrollar sistemas que aprenden automáticamente a partir de grandes conjuntos de datos, denominado *Machine Learning*, ML, [19].

Un modelo de ML, a grandes rasgos, define la relación entre una o más entradas y una o más salidas, [20]. Este modelo no se construye mediante reglas predefinidas, sino

que es el propio modelo el que, a través de un proceso de aprendizaje autónomo llamado entrenamiento, analiza un conjunto de datos proporcionado y descubre las relaciones internas relevantes.

Estos modelos tienen la ventaja fundamental de que, una vez entrenados, permiten obtener el valor de las variables de salida a partir de las variables de entrada de manera muy rápida. Por lo que, si son implementados con éxito, pueden suponer un ahorro en coste computacional en el proceso de optimización de la escala a nivel de retícula, también conocida como escala fina.

Sin embargo, según Woldseth et al. [21], la literatura científica que estudia la combinación de la inteligencia artificial y la optimización topológica: "Parece sobreestimar las habilidades de la tecnología de *Machine Learning* actual, ignorando que esos modelos son simplemente versiones complejas de regresión y clasificación y tratándolos como una especie de caja negra 'mágica' que es capaz de resolver cualquier problema".

Es decir, no se pueden emplear los modelos de ML de manera generalizable para cualquier problema. Para poder utilizar estos modelos es necesario delimitar el ámbito de su aplicación. En el caso de este trabajo concurren varias circunstancias que simplifican el problema y permiten el uso de estos modelos, entre ellas; todas las retículas tienen el mismo tamaño y este es reducido y, además, tanto el valor de la densidad, como las condiciones de contorno y de carga están acotadas.

1.3. Objetivos del trabajo

Este TFG se plantea como objetivo principal desarrollar una herramienta informática basada en IA que permita reducir el coste computacional de la optimización multiescala. Para ello se obtiene un modelo de IA que es capaz de obtener topologías de la escala fina con menor coste computacional que el algoritmo basado en SIMP, pero de manera aproximada.

Para lograr el objetivo principal del TFG se establecen una serie de objetivos parciales que se presentan a continuación:

- Definir el problema de la escala fina de manera desacoplada de la escala macro.
- Acotar el problema y determinar una arquitectura de IA que permita obtener las topologías óptimas a partir de las cargas aplicadas en el contorno de cada celda.
- Entrenar los modelos de IA para obtener las topologías óptimas para las cargas dadas.
- Fijar criterios que permitan evaluar la calidad de las soluciones obtenidas mediante esta técnica.

El resto del documento se estructura de la siguiente manera:

- En el Capítulo 2 se exponen con más detalle las herramientas empleadas, se presenta el algoritmo de optimización topológica SIMP, se expone el fundamento del método de los elementos finitos, y se explican los modelos de IA empleados.
- En el Capítulo 3 se detalla la metodología empleada para diseñar las arquitecturas y para generar los datos para entrenarlas.
- En el Capítulo 4 se realiza por una parte un análisis de resultados obtenidos mediante modelos de IA, y por otra parte, un análisis de carácter global de los resultados, comparando los resultados con el método SIMP.
- En el Capítulo 5 se detallan las conclusiones generadas tras la realización del proyecto y del análisis de los resultados obtenidos.

2 | Fundamento teórico

En esta sección se presentan con más detalle las herramientas empleadas para abordar el presente trabajo. En primer lugar, se expondrán las bases de MEF, que es la técnica que emplea el método SIMP en el proceso de optimización topológica. Finalmente, se explicarán las bases del modelo de aprendizaje supervisado empleado.

2.1. Método de los elementos finitos

Existe un problema clásico en la ingeniería mecánica denominado "problema elástico", este problema consiste en obtener el campo de desplazamientos de una estructura sometida a una carga externa. Conocer el campo de desplazamientos permite, además, conocer el valor de las deformaciones y de tensiones en dicha estructura.

En este trabajo es necesario resolver el problema elástico en varios escenarios; el primero de ellos es para conocer el campo de desplazamientos con el objetivo de poder hacer uso de la técnica SIMP como se verá más adelante, el segundo escenario es para poder calcular las condiciones de carga de cada una de las retículas antes de iniciar la resolución del problema de la escala fina.

Sin embargo, el problema elástico implica resolver ecuaciones diferenciales en derivadas parciales que no tienen solución analítica para la mayoría de problemas reales. Esto obliga a encontrar una solución aproximada al problema [22]. El método más extendido actualmente en la industria es MEF. En este trabajo se emplea esta herramienta para resolver el problema 2D, con material isótropo exclusivamente. Se puede afirmar que se ha resuelto el problema elástico cuando se conocen las siguientes variables:

$$\{u\} = \{u(x,y)\} = \begin{Bmatrix} u \\ v \end{Bmatrix} \quad \{\sigma\} = \{\sigma(x,y)\} = \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix} \quad \{\varepsilon\} = \{\varepsilon(x,y)\} = \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} \tag{2.1}$$

El MEF transforma el problema elástico continuo (Figura 2.1 izq.) en un problema discreto mediante una división del dominio Ω de trabajo. Así, se fragmenta el dominio en un subconjunto de elementos que están definidos por sus nodos. Estos nodos se encuentran localizados en los vértices de estos elementos, tal y como se aprecia en la derecha de la Figura 2.1.

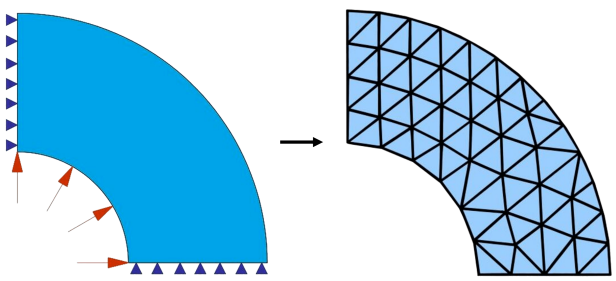


Figura 2.1: Discretización de dominio de trabajo. Fuerzas en rojo y condiciones de contorno en azul.

El método plantea la solución de la función de interés, normalmente desplazamientos en los nodos. El valor de esta función para otros puntos del elemento es interpolado a partir del valor de la función de los nodos mediante el uso de funciones de interpolación (ver Figura 2.2), que en este contexto se conocen como funciones de forma.

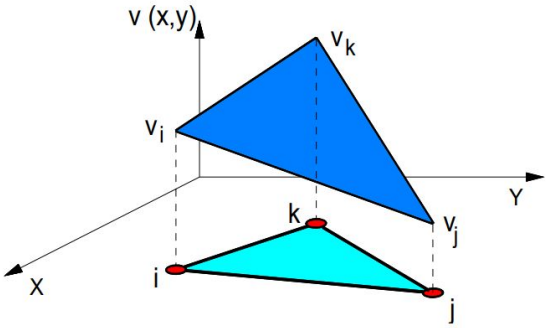


Figura 2.2: Representación de interpolación lineal en un elemento triangular con tres nodos para el desplazamiento v.

Escrito en forma de ecuación se tiene que:

$$\{u(x,y)\} \approx [N(x,y)] \{u_e\} \tag{2.2}$$

Donde u_e es vector que contiene el valor del desplazamiento en los nodos.

El dominio de trabajo se puede discretizar en mayor o menor medida. Cuanto más pequeño sea el elemento de la discretización, más exacta será la solución, pero a cambio más ecuaciones se involucrarán en ella, y más costosa será obtenerla.

Se puede demostrar que resolver el problema elástico consiste en resolver la Ec. 2.3. Esta ecuación relaciona las fuerzas ($\mathbf{F}$) aplicadas en los nodos, los desplazamientos en dichos nodos ($\mathbf{U}$) y la matriz de rigidez global ($\mathbf{K}$).

$$\mathbf{K} \cdot \mathbf{U} = \mathbf{F} \tag{2.3}$$

Para ensamblar la matriz $\mathbf{K}$ se han de calcular las matrices k_e de rigidez y los vectores de fuerza f_e de cada elemento, que se relacionan con u_e como: $k_e \cdot u_e = f_e$. Esta ecuación puede resultar familiar ya que constituye una suerte de generalización de la ley de Hooke. Así, resolver el problema elástico y obtener el campo de desplazamientos pasa por obtener la inversa de la matriz $\mathbf{K}$. El tamaño de la matriz $\mathbf{K}$ es directamente proporcional al número de discretizaciones que tiene el dominio.

Es conveniente recordar que el coste computacional que tiene resolver una matriz inversa es elevado, y según [23] está acotado superiormente por:

$$\mathcal{O}(n^{2.37}) \tag{2.4}$$

Es decir, el coste computacional de emplear MEF es polinómico respecto al número de elementos.

2.2. Método SIMP

Uno de los algoritmos de utilización más utilizados es el método SIMP. Fue planteado inicialmente por Bendsoe y Kikuchi (1988) y fue ampliado por Zozvany y Zhou (1992), [24]. Como se ha expuesto anteriormente, el método SIMP tiene como objetivo encontrar la distribución óptima de material dentro de un dominio específico cumpliendo las condiciones de contorno, las restricciones impuestas, teniendo en cuenta las cargas aplicadas.

El dominio con el que trabaja el método SIMP también se encuentra discretizado en un conjunto de elementos. De hecho, esta discretización es la misma que hace MEF. Esto es así porque es necesario conocer el valor del desplazamiento de cada elemento u^e, y esto se consigue ejecutando MEF sobre el dominio.

La idea fundamental del método es que las propiedades mecánicas de cada elemento se puedan escribir en relación con la densidad relativa. Así, se puede expresar la densidad que tiene cada elemento $\rho(x)$ en relación con la densidad máxima ρ_0 según la Ec. 2.5.

$$\rho(x) = x_e \rho_0 \tag{2.5}$$

donde x_e es la densidad relativa de un elemento en particular, y toma un valor entre 0 y 1.

Por otra parte, el módulo de Young de un elemento dado está descrito en la Ec. 2.6.

$$E(x) = x_e^p E_0 \tag{2.6}$$

siendo E_0 el módulo de Young máximo, y p un factor de penalización que evita valores intermedios de densidad relativa. Esto es, si el valor de p es elevado, significa que el módulo de Young de este elemento es muy bajo mientras que la densidad (que no está elevada a este factor), es alta; esta es una situación donde el material se emplea de forma ineficiente, y que el algoritmo evita. El factor de penalización es mayor que la unidad, y es habitual que tome el valor de 3, [25].

Como cualquier algoritmo de optimización se ha de definir una función a minimizar, en este caso se define la función de "cumplimiento" (a partir de ahora *compliance*). Esta función da cuenta de la energía de deformación de la estructura cuando está sometida a una carga. La interpretación es la siguiente: cuanto menor sea el valor de *compliance*, menor será la energía de deformación de la estructura, por tanto, más rígida será la estructura. La *compliance*, c, se presenta en la Ec. 2.7.

$$c = \mathbf{U}^{\mathrm{T}} \cdot \mathbf{K} \cdot \mathbf{U} = \sum_{e=1}^{N} (x_e)^p u_e^{\mathrm{T}} k_0 u_e \tag{2.7}$$

donde $\mathbf{U}$, $\mathbf{K}$ y u_e tienen el mismo significado que en el apartado que describía el MEF, y N es el número de elementos. Por otra parte, $k_0 = k_e/(x_e)^p$.

Además de minimizar el funcional, el algoritmo está sujeto a las restricciones de la Ec. 2.8.

$$
\begin{aligned}
&G = \frac{V}{V_0} - V_r = \frac{\sum_{e=1}^{N} V_e\, x_e}{V_0} - V_r = 0, \\
&\mathbf{K} \cdot \mathbf{U} = \mathbf{F}, \\
&0 < x_{min} < x_e < 1
\end{aligned}
\tag{2.8}
$$

donde G es la función que da cuenta de la restricción del volumen. La fracción de volumen viene dada por V_r, y será un criterio que fijará el diseñador. Por otra parte, también se ha de cumplir la ecuación del problema elástico, mientras que x_{min} es un valor pequeño que se fija para evitar que aparezcan regiones del dominio aisladas completamente del resto, lo cual provocaría un mal condicionamiento de la matriz.

Dependiendo del problema, se pueden generar estructuras con una distribución de densidades que se asemeja al patrón de un tablero, esto es, con densidades alternantes (ver Figura 2.3). Para evitar esta situación se hace uso de un filtro tal que la densidad de cada elemento está ponderada con la densidad de los elementos de su alrededor [26].

Figura 2.3: Estructura optimizada con *checkerboard patron* (patrón de tablero), [26].

En este trabajo se hace uso la herramientas SIMP y MEF a través de su implementación en el lenguaje MATLAB®.

2.3. Redes neuronales

Un modelo de redes neuronales es una herramienta de *Machine Learning* que puede ser utilizada para numerosas aplicaciones, como las de clasificación, donde se predice la pertenencia de una entrada a un grupo, o de regresión, modelando la relación entre variables dependientes y variables independientes, entre otros.

En términos matemáticos las redes neuronales representan funciones no lineales que relacionan una entrada n-dimensional con una salida m-dimensional. Así, sencillamente, si la entrada es $\boldsymbol{x}$, y la salida es $\boldsymbol{y}$, es decir:

$$
\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}
\tag{2.9}
$$

Entonces la red se puede reducir a una función tal que:

$$
\boldsymbol{y} = \boldsymbol{N}(\boldsymbol{x})
\tag{2.10}
$$

Tanto el nombre como el funcionamiento de las redes neuronales están inspirados en la forma que tienen de relacionarse las neuronas en el sistema nervioso, puesto que están conformadas por un conjunto de neuronas artificiales relacionadas entre sí. La arquitectura de redes neuronales más sencilla está constituida en tres tipos de capa: la capa de entrada, la capa oculta y la capa de salida. Esta arquitectura se denomina red neuronal densa (DNN) y se representa en la Figura 2.4.

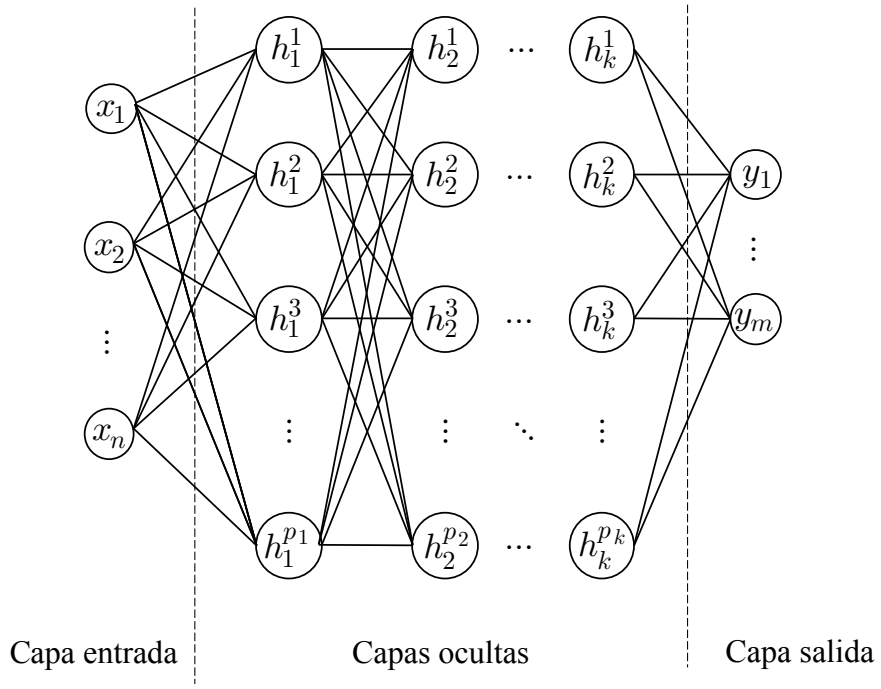


Figura 2.4: Representación arquitectura de red neuronal densa.

En la capa de entrada se organizan los parámetros de entrada a la red, esto es, de x_1 hasta x_n. En las capas ocultas es donde se encuentran la mayoría de neuronas artificiales, conectadas con la capa anterior y las siguientes, y así sucesivamente hasta llegar a la capa de salida. En esta última capa se disponen las neuronas que finalmente dan como resultado los parámetros de salida, desde y_1 hasta y_m. El número de capas ocultas y el número de neuronas de cada capa es una decisión de diseño, por lo que los posibles modelos son ilimitados.

La estructura interna de cada una de estas neuronas artificiales se muestra en la Figura 2.5; en concreto, a la vista del número de entradas, esta correspondería con una neurona de la primera capa oculta.

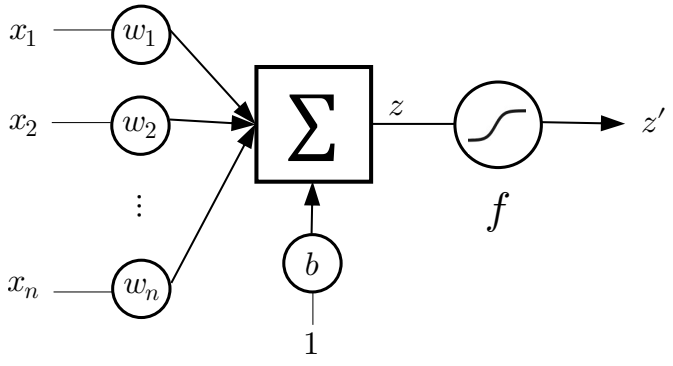


Figura 2.5: Representación neurona individual.

En el caso de una neurona artificial de la primera capa oculta, el número de entradas se corresponde con el número de entradas al modelo. En este caso habría n entradas. Cada una de estas entradas se multiplica por un parámetro w (de *weight*) y a continuación se suman todos los valores resultantes. Al sumatorio se le añade un parámetro b adicional (de *bias*). Se muestran estas operaciones de forma matemática en la Ec. 2.11.

$$z = \left(\sum_{1}^{n} x_n \cdot w_n \right) + b \tag{2.11}$$

El valor de salida de una neurona artificial es uno de los valores de entrada de las neuronas de la siguiente capa. El valor de salida de una neurona artificial no puede ser z, ya que el resultado de componer cualquier número de capas ocultas con neuronas lineales daría lugar a una función lineal sin capturar complejidad. Para abordar este problema se aplica una función no lineal, llamada función de activación a z, dando como lugar z'.

$$z' = f(z) \tag{2.12}$$

Se han propuesto multitud de funciones de activación, en la Figura 2.6 se muestran tres de las más populares: σ, tanh (función de tangente hiperbólica) y la función ReLU.

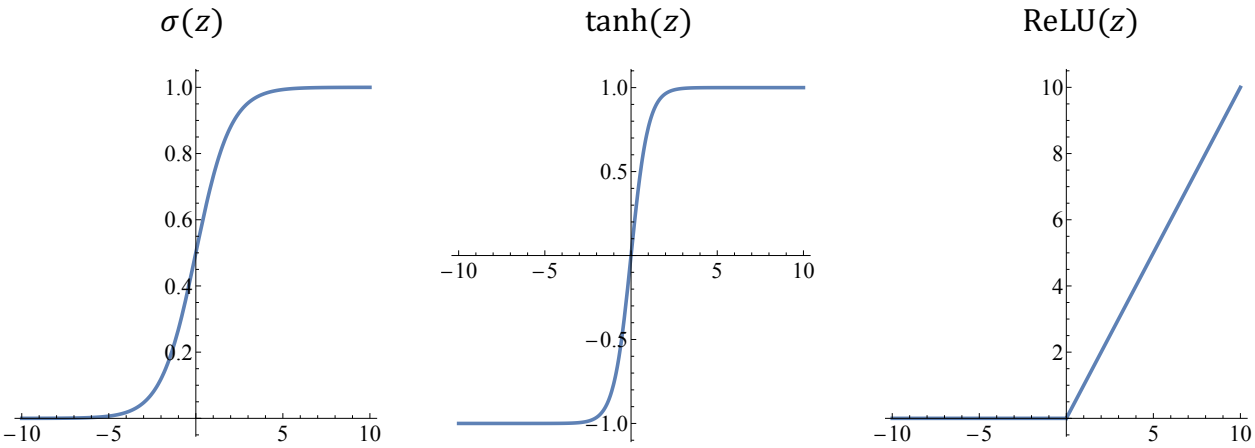


Figura 2.6: Representación funciones de activación habituales.

La función matemática que define a la función de activación σ se muestra en la Ec. 2.13.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.13}$$

Mientras que la función matemática que caracteriza la función ReLU es a trozos: z, para valores positivos y 0 para valores negativos y cero.

El valor z' que se obtiene tras aplicar la función de activación, ahora sí, es la salida de la neurona artificial. Las demás neuronas de la capa oculta y las de las demás capas tienen la misma estructura interna.

2.3.1. Descenso del gradiente

De manera análoga a la obtención de las constantes de una función matemática en el proceso de ajuste de regresión, es esencial encontrar el valor de los parámetros $\boldsymbol{w}$ y b para cada una de las neuronas artificiales. El proceso de cálculo de los parámetros de un modelo neuronal se denomina «entrenamiento». El objetivo es que los valores obtenidos logren que el modelo neuronal represente de manera fiel la relación entre las variables de entrada y las variables de salida.

Por lo tanto es conveniente cuantificar la calidad de un modelo neuronal y por consiguiente la elección del valor de los parámetros, definiendo un indicador de error que tiene como base de su funcionamiento la comparación del valor de las variables de salida predichas por el modelo neuronal frente a las variables de salida conocidas ante una misma entrada.

Existen múltiples definiciones del indicador de error del modelo neuronal, una definición habitual es el Error Cuadrático Medio (MSE) presentada en la Ec. 2.14.

$$E_{\text{total}} = \frac{1}{m}\sum_{i=1}^{m}(y_i - y_i^*)^2 \tag{2.14}$$

donde y_i es la variable de salida predicha por el modelo, y y_i^* es la variable de salida real.

Otra definición usual es el *Mean Squared Logarithmic Error Loss* (MSLE), que se presenta en la Ec. 2.15.

$$E_{\text{total}} = \frac{1}{m}\sum_{i=1}^{m}\left(\log\frac{y_i+1}{y_i^*+1}\right)^2 \tag{2.15}$$

de nuevo, y_i es la variable de salida predicha por el modelo, y y_i^* es la variable de salida real.

El proceso de entrenamiento tiene como objetivo reducir el valor de la *loss function*, en el contexto de este trabajo eso se corresponde con disminuir el error MSLE. Para ello en primer lugar se asignan valores aleatorios a los parámetros de las redes neuronales; y tras esto se calcula la derivada del error respecto a cada uno de estos parámetros, es decir, se calcula el gradiente. El gradiente se utiliza para actualizar el valor de cada uno de los parámetros de manera iterativa. Este procedimiento se denomina *backpropagation*.

Por ejemplo, para actualizar el valor del parámetro w_1 de la primera neurona en la primera capa se ha de calcular la derivada del error respecto de w_1 haciendo uso de la regla de la cadena. El valor de esta derivada indica cómo aumenta el error al aumentar el valor del parámetro estudiado, por lo que se puede usar para disminuir dicho error como se muestra en la Ec. 2.16.

$$w_1^* = w_1 - \eta\,\frac{\partial E_{total}}{\partial w_1} \tag{2.16}$$

donde η se denomina ritmo de aprendizaje y tiene un valor típico de 0.001. w_1^* es el valor nuevo del parámetro.

El valor del gradiente depende del valor de los propios parámetros, que se actualizan a cada iteración. Es por esto por lo que en la siguiente iteración se ha de recalcular el gradiente y actualizar de nuevo los valores. Este proceso se repite hasta que el error del modelo sea suficientemente bajo, a criterio del diseñador.

La arquitectura de redes neuronales densas ha sido empleada en numerosas aplicaciones con un gran éxito. Sin embargo, existen numerosas desventajas que hacen que su uso no sea una buena elección cuando se trabaja con datos en forma de imágenes:

- Cuando el modelo tiene un gran número de entradas, tales como en aplicaciones con imágenes o vídeos, el número de parámetros es extremadamente elevado. Esto hace que el coste computacional de entrenar el modelo RNN sea inasumible y que, además, se requiera una gran cantidad de datos.
- Esta arquitectura es propensa a sobrejustar los datos de entrenamiento, y por tanto, no suelen generalizar bien con nuevos datos [27].
- Esta arquitectura permite identificar patrones globales en los datos de forma eficiente, pero no patrones locales [28]. Es decir, una red neuronal convencional no toma ventaja de la topología de los datos de entrada.

2.3.2. Redes neuronales convolucionales

En los escenarios de uso donde se requiera trabajar con imágenes existe una alternativa a las redes RNN, estas son las redes neuronales convolucionales (CNN). Las redes CNN son particularmente útiles para encontrar patrones en las topologías de los datos de entrada. Una red CNN se compone de una sucesión de objetos que operan transformaciones matemáticas a unos datos de entrada, obteniendo otros datos de salida. Los objetos más habituales son los siguientes: filtros, *pooling* y desconvolución.

Filtros

En una red CNN el proceso de filtrado consiste en la obtención de un determinado número de mapas de características aplicando la operación de convolución de una serie de matrices denominadas núcleo sobre un conjunto de canales de entrada (por ejemplo, una imagen en RGB).

La operación matemática de convolución en este contexto corresponde a multiplicar elemento a elemento los valores del núcleo con los de la matriz de entrada, y sumar todos estos resultados. Para obtener el mapa de características se aplica múltiples veces la operación de convolución, desplazando el núcleo sobre la matriz de entrada. Se puede apreciar la operación de convolución en la Figura 2.7, donde el núcleo tiene un tamaño de 3x3 elementos, y a partir de una matriz de entrada de 5x5 entradas, se obtiene un mapa de características de 3x3.

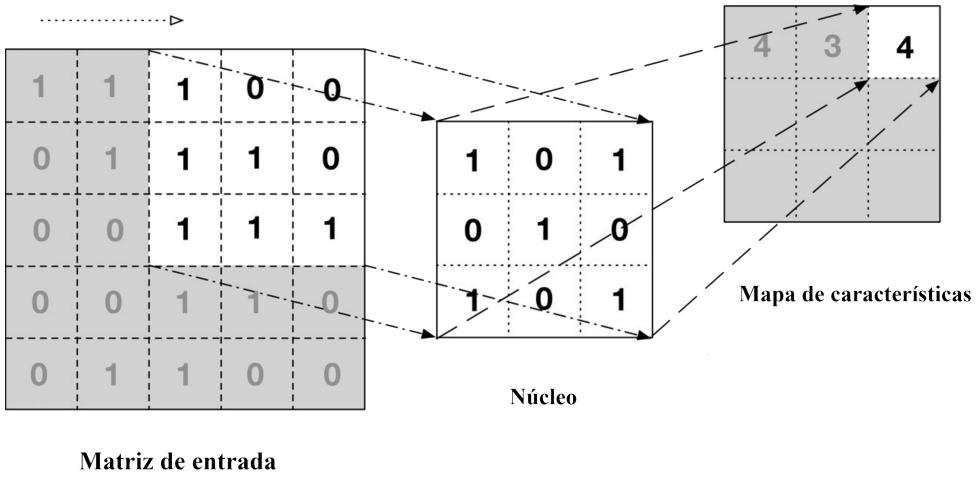


Figura 2.7: Representación operación de convolución, [29].

El proceso de filtrado en el ejemplo de la Figura 2.8 es representativo, la variable de entrada al filtro son tres matrices; por los colores que emplean en este ejemplo se puede asumir que se corresponde con una imagen RGB. Se aprecia que para obtener el mapa de características se aplica la operación de convolución con tres núcleos diferentes, y se suma un sesgo para cada mapa de características de salida.

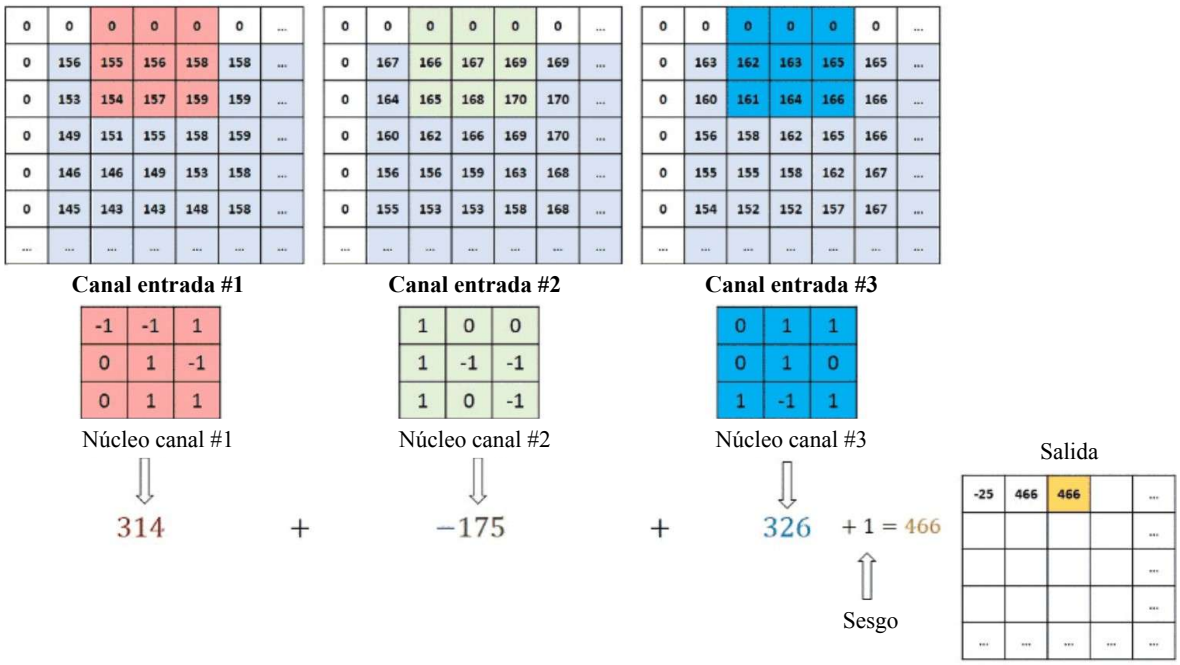


Figura 2.8: Representación proceso de filtrado, [29].

Siguiendo con el ejemplo, si el diseñador decide que ese filtro tendrá como salida dos mapas de características, significa que hay 9 parámetros en cada núcleo por entrenar, más un sesgo, por dos filtros. Es decir: 56 parámetros a entrenar. Es interesante darse cuenta que el número de parámetros a entrenar no depende del tamaño de la imagen de entrada.

Pooling

En el contexto de una red CNN, el proceso que se denomina *pooling* tiene como función reducir el tamaño de una matriz. Esto se hace para minimizar el coste computacional del entrenamiento reduciendo el número de parámetros a entrenar.

El proceso de *pooling* se puede hacer diversas maneras, en la Figura 2.9 se muestra la operación de *pooling* aplicada realizando un promedio de los elementos.

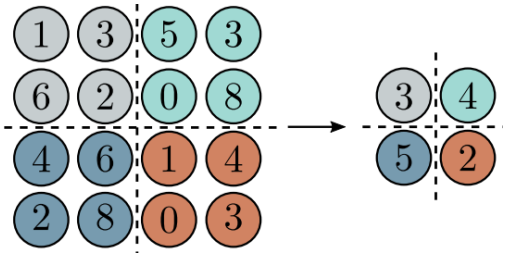


Figura 2.9: Representación operación *average pooling*, [20].

Por otra parte, en la Figura 2.10 se realiza la operación de *pooling* tomando el valor más alto de cada cuadrante. Esta es la manera más habitual de realizar este procedimiento puesto que, según la literatura, permite conservar los datos más importantes de la matriz original.

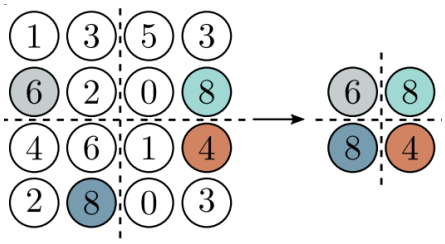


Figura 2.10: Representación operación *max pooling*, [20].

Deconvolución

También es habitual la operación llamada "deconvolución". Su objetivo es obtener una matriz de salida más grande que la matriz de entrada. Existen multitud de implementaciones posibles para esta operación, sin embargo, habitualmente consiste en realizar la operación de convolución mediante un núcleo de cierto tamaño, sobre la matriz de entrada y sus alrededores. En la Figura 2.11 se muestra la entrada, el núcleo y el resultado de una operación de este tipo. Se pueden seguir los pasos ejecutados para obtener el resultado en la Figura 2.12.

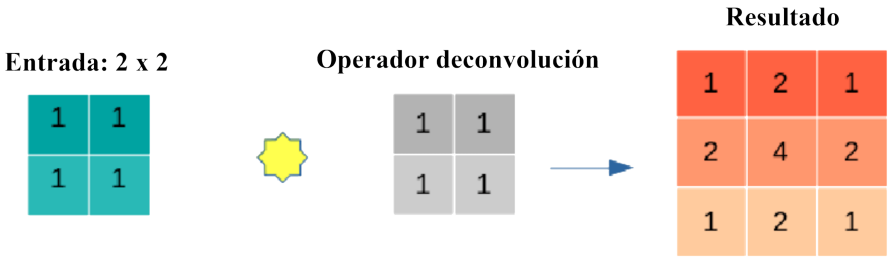


Figura 2.11: Planteamiento ejemplo de deconvolución. Se muestra entrada, operador, y resultado, [30].

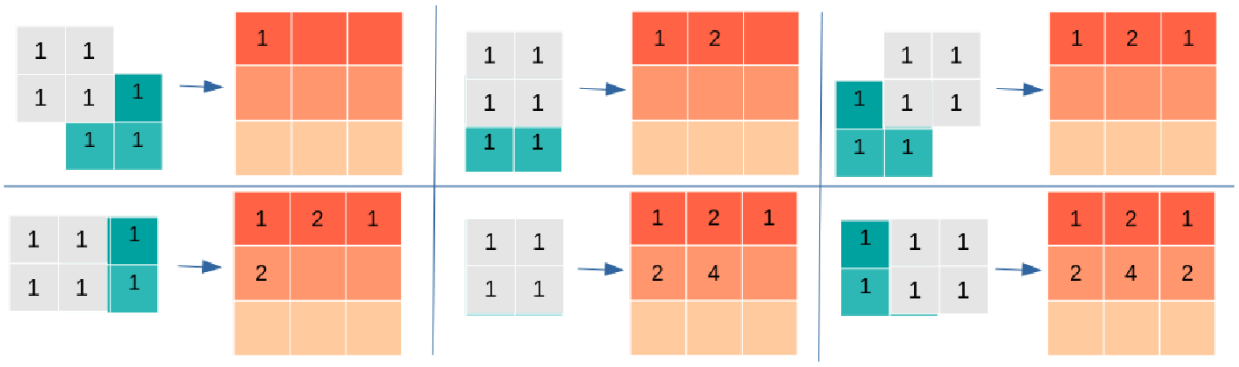


Figura 2.12: Planteamiento ejemplo de deconvolución. Se muestran pasos de la operación, [30].

Funciones de activación

En el contexto de las CNN se aplican las funciones de activación que se han definido previamente a las matrices de salida de los filtros, elemento a elemento.

2.3.3. Redes neuronales recurrentes

Las redes neuronales presentadas anteriormente no tienen en consideración el orden con que se introducen los datos. Esto es, el modelo da la misma respuesta ante una misma entrada, sin importar el orden con que se introducen estas entradas. Se puede representar este tipo de redes en la Figura 2.13. Sin embargo, esto no es conveniente en algunos escenarios de uso donde la secuencia es importante, tales como problemas que involucren vídeos, predicción meteorológica, secuencias de texto, etc.

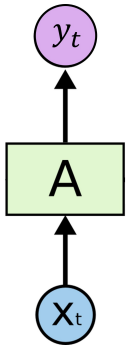


Figura 2.13: Representación entrada salida de un modelo neuronal A no recurrente, [31].

La literatura propone emplear modelos basados en redes neuronales recurrentes (RNN) en problemas donde la secuencia sea importante. En los modelos RNN ciertas variables internas del modelo son empleadas como entradas adicionales del mismo modelo pero de un instante de la secuencia posterior, de ahí su carácter recurrente. Se puede representar el modelo RNN en la Figura 2.14, donde se aprecia que el orden con que se introducen los datos sí que es importante puesto que la entrada adicional al modelo varía dependiendo de la posición en la secuencia.

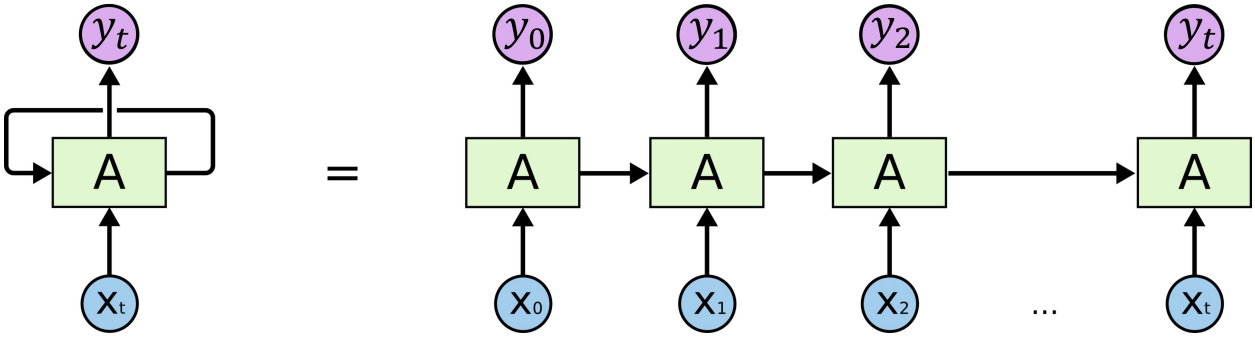


Figura 2.14: Representación entrada salida de un modelo neuronal A recurrente, [31].

En concreto, existe una arquitectura de red RNN denominada *long short-term memory* (LSTM) que es particularmente eficiente detectando patrones en una secuencia de datos. La pieza fundamental de la arquitectura LSTM es la célula LSTM. En su versión básica esta célula tiene como entrada un vector de datos de dimensión n y como salida un vector de dimensión m. La capacidad que tiene la célula de detectar patrones en una secuencia de datos se basa en la existencia de dos variables que juegan el papel de memoria; por una parte la variable C, que actúa como memoria a largo plazo, y por otra la variable h, que actúa como memoria a corto plazo y que es igual a la salida para cada instante de la secuencia. Para exponer con detalle el funcionamiento de una célula es conveniente contar con la representación de una unidad; mostrada en la Figura 2.15.

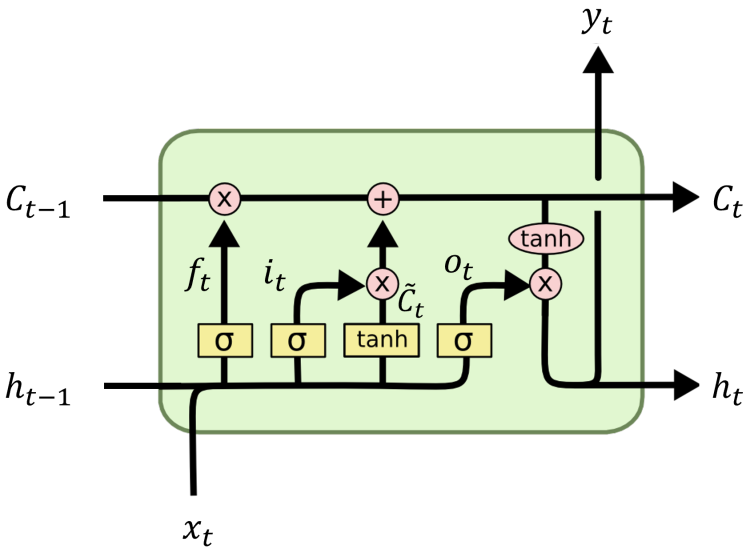


Figura 2.15: Representación célula LSTM para un instante de la secuencia, [31].

Esta representación captura la célula LSTM para un instante de la secuencia, donde la variable de entrada es x_t y la variable de salida es y_t. Cada instante de la secuencia cuenta, además, con dos entradas adicionales: C_{t-1} y h_{t-1} que son los vectores de memoria de largo y corto plazo respectivamente, generados por la célula en instante anterior, y que se inicializan en cero para t = 0. El valor de las variables de memoria de largo y corto plazo son modificados en cada instante de tiempo. De acuerdo con la Figura 2.15 se siguen los siguientes pasos:

En primer lugar se calcula el vector f_t de dimensiones $\mathbb{R}^m$ que contiene números que, por acción de la función de activación σ, se encuentran comprendidos entre 0 y 1. Este vector se multiplicará elemento a elemento con el vector C_{t-1} disminuyendo más o menos su valor; por lo tanto, esta parte de la célula se comporta como una «puerta para el olvido». El cálculo del vector f_t involucra una matriz W_f de dimensiones $\mathbb{R}^{(mxd)}$, una matriz U_f de dimensiones $\mathbb{R}^{(mxm)}$ y un vector b_f de dimensión $\mathbb{R}^m$, que contienen parámetros con valores que se obtendrán durante el proceso de entrenamiento. Matemáticamente estas operaciones se expresan como:

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \tag{2.17}$$

En segundo lugar se calculará el vector $\tilde{c}_t$ de dimensiones $\mathbb{R}^m$, que contiene números que, por acción de la función de activación tanh, se encuentran comprendidos entre -1 y 1. Este vector se multiplicará elemento a elemento con el vector i_t, tras esta operación se suma el resultado a la memoria de largo plazo; por lo tanto, esta parte de la célula se comporta como una «puerta para nueva memoria». El cálculo del vector $\tilde{c}_t$ involucra a una matriz W_c de dimensiones $\mathbb{R}^{(m \text{ x } d)}$, una matriz U_c de dimensiones $\mathbb{R}^{(m \text{ x } m)}$ y un vector b_c de dimensión $\mathbb{R}^m$ con parámetros a obtener durante el proceso de entrenamiento. El vector i_t es análogo al vector f_t, con matrices de las mismas dimensiones, y se emplea para disminuir más o menos el valor del vector $\tilde{c}_t$ antes de sumarlo a la memoria de largo plazo. El cálculo de $\tilde{c}_t$ se encuentra en la Ec. 2.18, mientras que el cálculo de i_t se muestra en la Ec. 2.19:

$$\tilde{c}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c) \tag{2.18}$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \tag{2.19}$$

Entre los dos primeros pasos se tienen todos los elementos necesarios para calcular el nuevo valor de C_t:

$$C_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{2.20}$$

Donde «∘» es el operador de multiplicación de elemento a elemento.

Finalmente, se calcula el nuevo valor de la memoria a corto plazo h_t que se corresponde con la variable de salida y_t. Para ello se multiplica el vector o_t, elemento a elemento, con el resultado de aplicar la función de activación a la memoria de salida $\tanh(C_t)$. Esta parte de la célula se denomina «puerta de salida». El vector o_t se calcula de manera análoga a f_t y a i_t, con matrices de las mismas dimensiones. Matemáticamente estas operaciones se muestran en:

$$h_t = o_t \circ \tanh c_t \tag{2.21}$$

$$y_t = h_t \tag{2.22}$$

El valor de los parámetros de las matrices W y los vectores b de cada una de las «puertas» de la célula es invariante para todos los instantes de la secuencia puesto que no se trata de células distintas, sino de una representación extendida en el tiempo de la misma unidad. Esta configuración del modelo, basado en «puertas» permite tanto «olvidar» parte de la memoria a largo plazo, modificar dicha memoria, y usarla para generar la memoria a corto plazo y la salida de cada uno de los instantes temporales. La representación extendida de una célula LSTM se encuentra en la Figura 2.16.

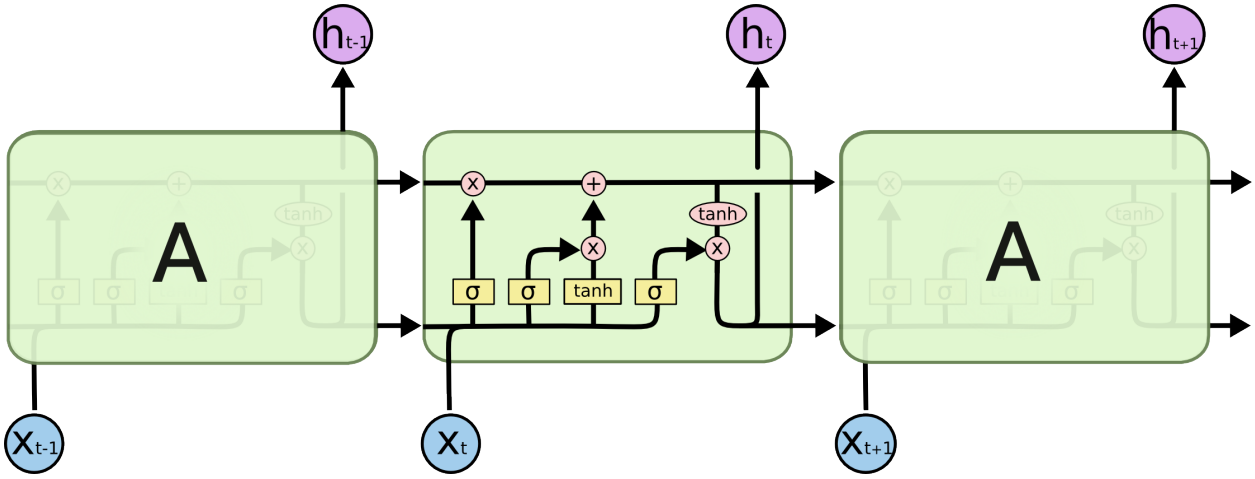


Figura 2.16: Representación varias celdas de LSTM, [31].

Esta arquitectura puede ser adaptada para el uso de imágenes, sustituyendo los operadores entre matriz y vector, por los operadores de filtrado, basados en operaciones de convolución mostradas anteriormente.

Para implementar esta tecnología en el presente proyecto se emplea la librería TensorFlow Keras, la cual tiene una interfaz simple y consistente, [32], y se utiliza el lenguaje de programación Python que, además, permite acceder a una gran variedad de extensiones útiles.

3 | Metodología

Una vez presentado el fundamento teórico sobre el que se edifica el trabajo, en esta sección se expondrá la metodología seguida para introducir una herramienta de ML que reduzca el coste computacional de la optimización topológica multinivel.

3.1. Estrategia multinivel

Es conveniente exponer con detalle las distintas etapas que tiene la optimización multinivel tradicional y cómo se relacionan estas etapas entre sí para comprender cómo se puede articular una estrategia alternativa que incluya la utilización de redes neuronales.

Como se ha expuesto con anterioridad, la estrategia de optimización multinivel tiene como objetivo obtener una estructura reticular, en el seno de un dominio de trabajo fijado, a partir de unas condiciones de carga y unas condiciones de contorno determinadas, tal y como se aprecia en la Figura 3.1.

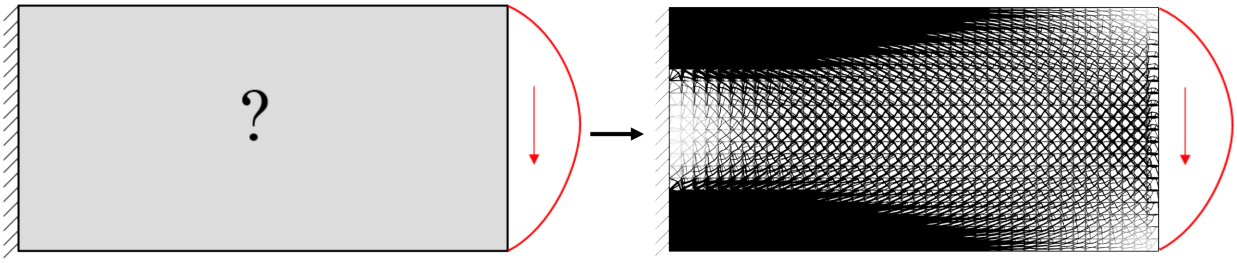


Figura 3.1: Dominio de diseño (izq.) y solución del problema de optimización multinivel (dcha.) de una viga empotrada con una carga tangencial en el extremo.

La obtención de la estructura reticular se consigue integrando dos grandes etapas:

En la primera etapa se discretiza el domino Ω de trabajo en un conjunto de elementos, mediante un proceso denominado mallado. Por su simplicidad, se emplea una malla cartesiana con elementos cuadrados de igual tamaño que cubren todo el dominio. Cada una de estas celdas constituirá el dominio de las retículas en la segunda etapa. Tras el mallado del dominio se emplea el método SIMP, que internamente hace uso de MEF, para calcular de manera iterativa la distribución de densidades óptima. Esto es, cada iteración tiene como resultado una topología más rígida que la anterior. Cuando se alcance el criterio de convergencia, se obtendrá la topología óptima del problema de escala macro.

La segunda etapa tiene como objetivo optimizar cada una de las celdas de la etapa macro. Para ello es preciso analizar a qué cargas está sometida cada celda, empleando MEF sobre la topología macroscópica para averiguarlo. Las cargas sobre cada celda son aproximadas a un conjunto de tracciones lineales sobre cada una de las caras. Cada una de las celdas, asemejadas a retículas, son discretizadas con una malla cartesiana equidimensional con una dimensión fija de 32×32 elementos. En el proceso de optimización multinivel habitual se aplica nuevamente el método SIMP durante la segunda etapa para obtener el mapa de densidades óptimo de cada una de las retículas.

En la Figura 3.2 se muestra un diagrama de flujo en que se aprecian las dos etapas que se han descrito en este apartado.

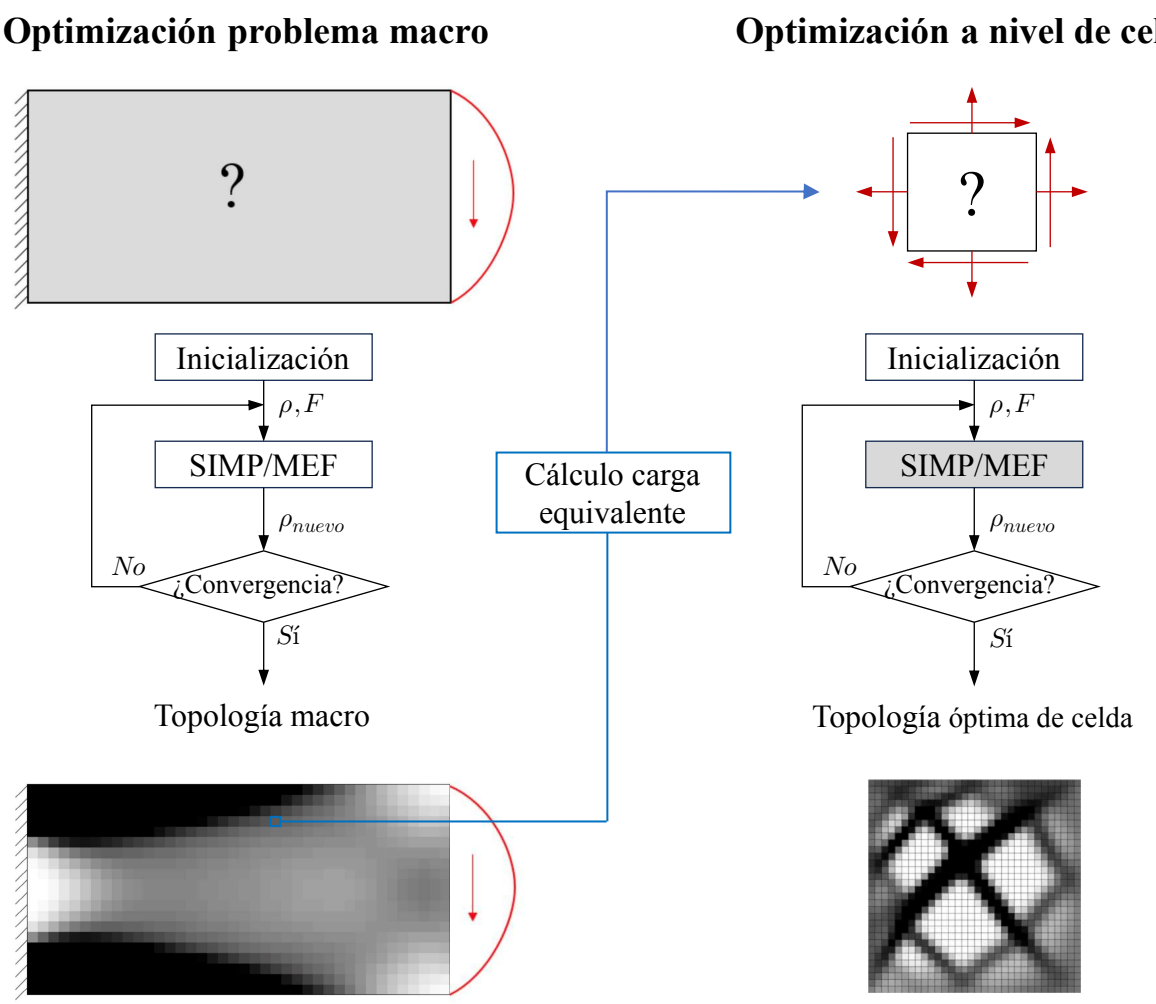


Figura 3.2: Diagrama de flujo problema multiescala tradicional.

Tal y como se puede apreciar en el diagrama de flujo de la Figura 3.2 el proceso de optimización a nivel de celda se ha de aplicar a cada una de las retículas de la topología macro. Aunque en este proyecto cada retícula comprende solamente 32×32, y con esa cantidad de elementos se puede realizar un proceso de optimización con un coste computacional reducido, hay una gran cantidad de estas retículas, lo que provoca que al final el coste sea elevado.

Como se ha expuesto, el proceso de optimización topológica involucra hacer uso de la estrategia SIMP que tiene asociada la utilización del método MEF, que es una herramienta cuya ejecución tiene un elevado coste computacional. Por eso, es razonable tratar de sustituir esas herramientas por otra, en el caso de este proyecto se propone utilizar un modelo basado en redes neuronales (NN).

En la Figura 3.3 se muestra la estrategia propuesta donde una red (NN) propone la nueva distribución de densidades a partir de la distribución en la iteración anterior y las condiciones de carga. Es conveniente señalar, a la vista del diagrama, que la estrategia propuesta basada en el modelo NN no sustituye el carácter iterativo del proceso de optimización de la escala fina. El nuevo proceso seguirá siendo de naturaleza iterativa. Este apunte no es menor, puesto que se sigue la recomendación que hacen diversas fuentes en la literatura que indican que la obtención directa de una estructura optimizada a partir de una entrada, con la tecnología actual, no es posible [21].

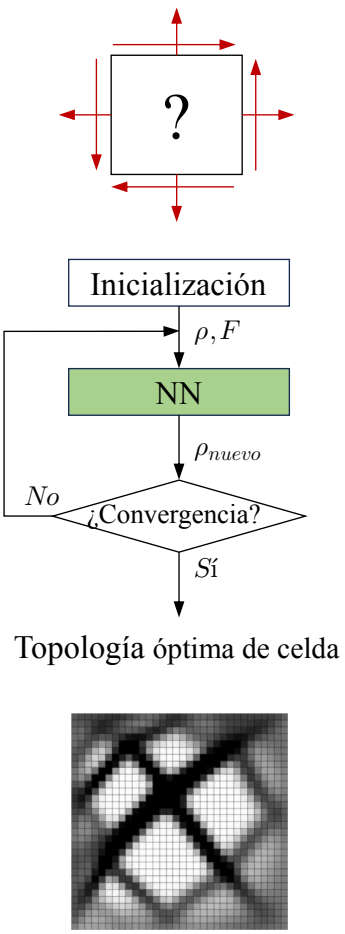


Figura 3.3: Diagrama de flujo problema a nivel de celda, con redes neuronales.

Ahora, es necesario dilucidar qué arquitectura o combinación de arquitecturas se han de implementar en el modelo de NN de la caja verde del diagrama. Las posibles arquitecturas y combinaciones son ilimitadas. Sin embargo, existen algunos indicios significativos en el problema tratado que permiten acotar la variedad de herramientas a utilizar:

- Cada celda del problema macro es de igual tamaño.
- A su vez, cada celda del dominio macro es discretizada mediante la misma malla cartesiana con elementos cuadrados equidimensiones; comprendiendo un total de 32 $\times$ 32 elementos.
- El valor de la densidad en el mapa de densidades está acotado entre 0 y 1.
- Dado que se está trabajando en la zona lineal del material, el valor de las tensiones en las caras de las celdas se puede escalar a valores entre -1 y 1.

El elevado número de elementos, en total 1024, hace que se descarten directamente las redes neuronales densas (DNN), ya que para obtener modelos con poco error asociado en escenarios con muchas variables de entrada y de salida, se requeriría un número extremadamente alto de parámetros de entrenamiento. Esto haría que el modelo ocupara un tamaño enorme en memoria, y que el coste computacional del entrenamiento del modelo fuera inasumible.

Por otra parte, el hecho de que se discretice la retícula con una malla cartesiana cuadrada hace posible que tanto el mapa de densidades como el mapa de otras características de la retícula puedan ser tratadas como imágenes de 32×32 píxeles. Esto es positivo porque las herramientas de redes neuronales que trabajan con imágenes (en concreto las redes CNN) sí que son eficientes, tal y como se había señalado en secciones anteriores. Además, al contrario que NN, las redes CNN son capaces de aprender de forma eficiente la relación entre elementos vecinos, al hacer uso de filtros.

Sin embargo, el grupo de redes CNN también es amplio. Para seleccionar el tipo exacto de red CNN que se empleará, se realiza una búsqueda bibliográfica en aplicaciones de redes en problemas de optimización topológica. Esta búsqueda arroja como resultado la red U-NET: [33, 34, 35, 36, 37], y que por tanto, se utilizará en el problema. La red U-NET es una arquitectura que, dentro del grupo de redes CNN, se encuentra en el grupo de las redes denominadas *image-to-image* puesto que su variable de entrada es una imagen, y la variable de salida es otra imagen.

También se ha utilizado la arquitectura LSTM puesto que mejora el problema de la convergencia como es el caso de [38]. La redes LSTM aplicadas en problemas que involucran imágenes tienen como variable de entrada un número a definir de imágenes, y como salida, el mismo número de ellas.

Ahora, es necesario delimitar apropiadamente el ámbito de aplicación de cada arquitectura. Para ello se propone hacer un estudio, desde un punto de vista global, de la evolución de la *compliance*, c, de las topologías obtenidas mediante el algoritmo SIMP. La *compliance*, como se presentó anteriormente, da cuenta de la energía de deformación de una topología cuando es sometida a una carga. Por tanto, la compliance es inversamente proporcional a la rigidez, y de hecho, es la función que busca minimizar el algoritmo de optimización SIMP.

Para que este análisis sea generalizable es preciso que tenga en cuenta un promedio entre un gran número de casos distintos. Para poder realizar el promedio se calcula la *compliance* relativa, esto es, la *compliance* de una determinada iteración entre la *compliance* de la iteración inicial, c_0. Este cociente está normalizado y sí que es promediable. Si el valor de c_i/c_0 de una determinada iteración i es menor que uno, significa que el algoritmo ha logrado rigidizar la estructura respecto a la iteración inicial. En la iteración inicial (iteración 0) todos los elementos tienen asignado el valor de densidad relativa que es igual a la fracción de volumen impuesta, en este caso 0.5.

Entonces, se calcula la *compliance* relativa para las primeras iteraciones, de 500 casos de carga y se calcula el promedio de compliance relativa de cada iteración

$$\hat{r_i} = \frac{1}{500} \sum_{n=1}^{500} \frac{c_i}{c_0} \tag{3.1}$$

donde $\hat{r_i}$ es la *compliance* relativa promedio de cada iteración.

Así, se representa el valor de $\hat{r_i}$ en la Figura 3.3. En esta figura se puede apreciar el descenso pronunciado de la *compliance* relativa para las primeras nueve iteraciones, donde c/c_0 pasa de 1 a 0.7. Tras esto, se puede apreciar una segunda etapa de diez iteraciones donde el descenso sigue siendo intenso, pero con menos pendiente que la primera fase; c/c_0 pasa de 0.7 a 0.5. Durante la etapa final el descenso es mucho más lento, puesto que se acerca a la convergencia, y la *compliance* relativa pasa de 0.5 a, solamente, 0.45.

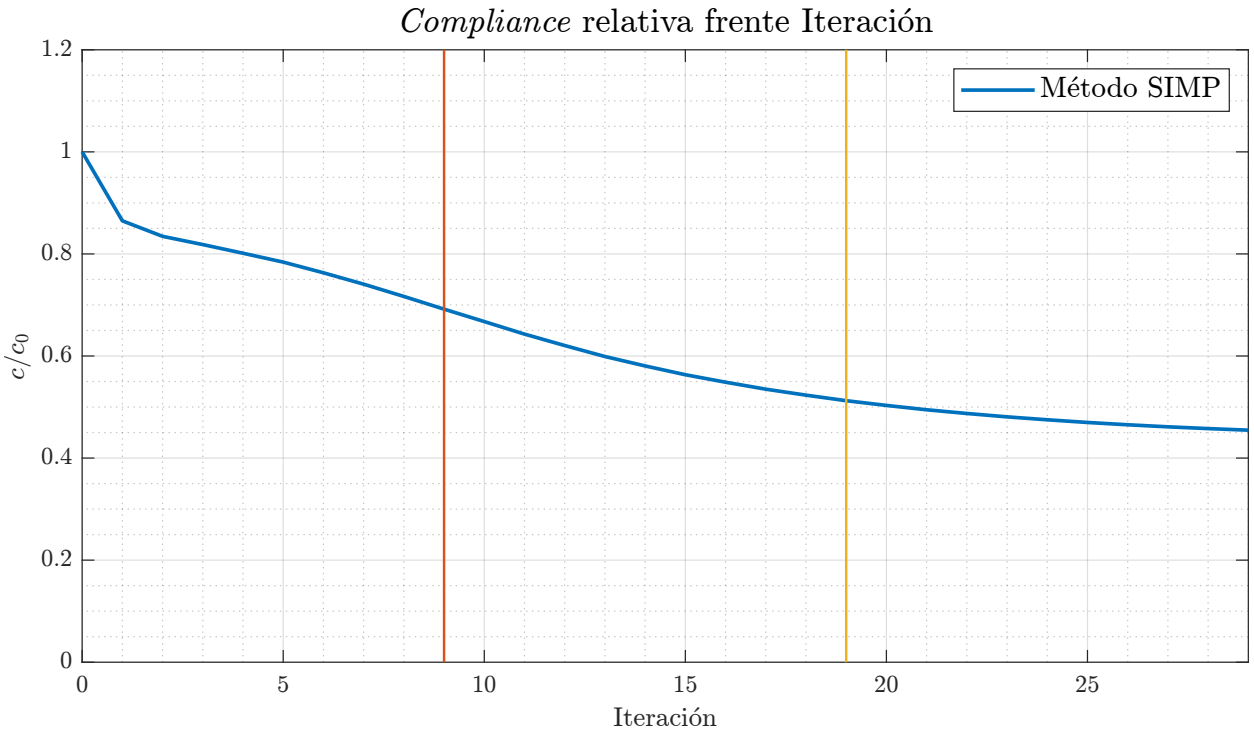


Figura 3.4: *Compliance* relativa promedio frente a iteración con algoritmo SIMP.

El estudio de la Figura 3.4 es muy esclarecedora puesto que a partir de ella se pueden obtener numerosos criterios de diseño:

- Durante las últimas iteraciones el descenso de la función objetivo es casi asintótico, por lo que se puede centrar el estudio y la aplicación de las redes neuronales a las primeras 30 iteraciones, (de la iteración 0, a la iteración 29).
- Hay tres etapas de descenso con diferentes pendientes. Para las dos primeras zonas se emplea la arquitectura U-NET, mientras que para la última etapa, con poca pendiente, se emplea la arquitectura LSTM.
- Las dos primeras etapas tienen diferente pendiente entre sí, por lo que en este trabajo se decide entrenar un modelo U-NET para generar topologías en la primera etapa, y otro modelo U-NET para generar topologías en la segunda zona. La decisión de entrenar dos modelos es el de capturar la complejidad de dos zonas diferentes, con características diferentes, de forma eficiente.
- El número de imágenes de entrada al modelo LSTM se fija en diez; de esta manera, el modelo genera topologías en la última zona teniendo como entrada una secuencia de topologías sin mucha variación de rigidez.

Hasta el momento, se ha presentado la justificación del uso de las arquitecturas que se utilizan para construir el mecanismo contenido en la caja verde del diagrama de la Figura 3.3, también su ámbito de utilización. El siguiente paso consiste en detallar su implementación, esto es, dar cuenta de los datos de entrada, datos de salida, configuración de parámetros, y otros aspectos relevantes.

3.1.1. Implementación U-NET y CNN-LSTM

Desde un punto de vista externo, el mecanismo toma como entrada el mapa de densidades de la iteración anterior, así como las condiciones de contorno y de la carga. Tras esto, predice como salida el mapa de densidades actualizado con una estructura más rígida, que constituye la iteración siguiente.

Debido a la naturaleza de la arquitectura U-NET es necesario codificar toda la información de entrada al mecanismo como una sola imagen. Esto es sencillo, puesto que el mapa de densidades se puede traducir directamente en una imagen de 32 × 32 píxeles, tal y como ya se había señalado. Por otra parte, las condiciones de carga lineales en las caras del dominio de la retícula también se pueden traducir a una imagen; aproximando el valor de las tensiones al valor de los píxeles del borde de dos imágenes de 32 × 32.

En síntesis, la información de entrada de la arquitectura U-NET es una imagen de tres canales, donde dos de ellos están relacionados con las condiciones de carga, y el otro con el mapa de densidades de la iteración anterior (i), mientras que la salida del modelo U-NET es el mapa de densidades de la iteración nueva (i+1), tal y como se aprecia en la representación de la Figura 3.5.

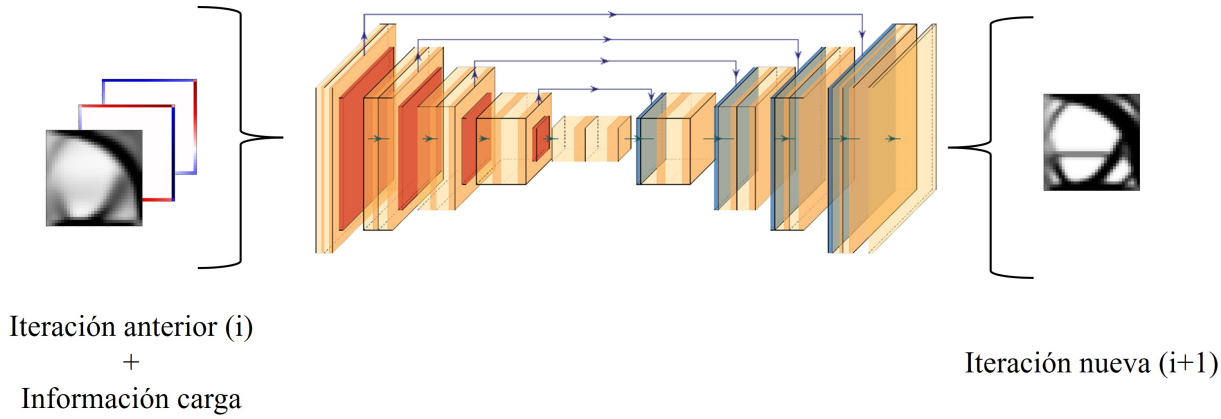


Figura 3.5: Representación datos de entrada y de salida modelo U-NET.

Por otra parte, la red CNN-LSTM tiene como variable de entrada una secuencia de 10 imágenes. Cada una de esas imágenes tiene tres canales, de hecho, se codifica la misma información que en la red U-NET, es decir; dos canales para información de condición de carga, y el otro canal para el mapa de densidades. La secuencia de datos de entrada comprende desde la iteración i-9, hasta la iteración i, todos esos datos son conocidos. Mientras que la secuencia de datos de salida comprende desde la iteración i-8 hasta la iteración i+1. Se hace una representación en la Figura 3.6 de los datos de entrada y de salida.

En otras palabras, se introduce una secuencia de las últimos diez mapas de densidad (y condiciones de carga) y se obtiene como salida una secuencia de los nueve últimos mapas de densidad y uno adicional, que se corresponde la iteración i+1, que es la topología predicha.

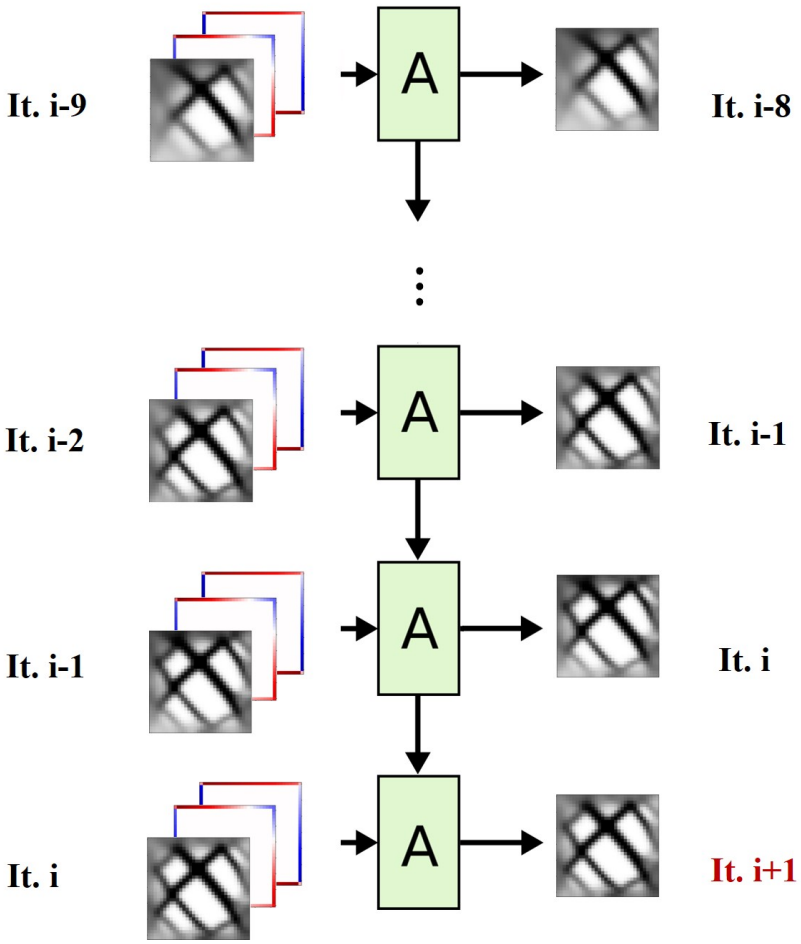


Figura 3.6: Representación datos de entrada y de salida modelo LSTM.

Dimensionamiento las redes neuronales

Faltan por conocer los datos concretos de los modelos de redes neuronales utilizados en este trabajo.

Los dos modelos neuronales de arquitectura U-NET están construidos a nivel interno como indica la Figura 3.7, a la vista de esta figura se puede afirmar que:

- Las operaciones que se producen entre dos barras contiguas son operaciones de filtrado con núcleos 3×3, con el número de canales de salida que marca la parte superior de la barra. Por ejemplo, la barra naranja representa la entrada, y la barra contigua de color azul es el resultado de aplicar un proceso de filtrado con un núcleo de 3×3 para obtener 64 mapas de características, tienen asociadas la función de activación ReLU.
- Las operaciones que representan las flechas rojas son de *max pooling*. Estas reducen a la mitad el tamaño, por ejemplo, en la primera operación la imagen pasa de 32×32 píxeles a 16×16 píxeles.
- Las operaciones que representan las flechas de color morado son operaciones de deconvolución.
- Las operaciones que representan las flechas de color gris concatenan dos imágenes de las mismas dimensiones y con los mismos canales.
- La barra morada representa la imagen de salida, que solo tiene un canal, tiene asociada la función de activación σ.

El modelo en concreto tiene aproximadamente 23.18 millones de parámetros, y su tamaño en disco es 265.5 MB. Este número de parámetros podría variar en caso de asignar más filtros a cada capa de convolución.

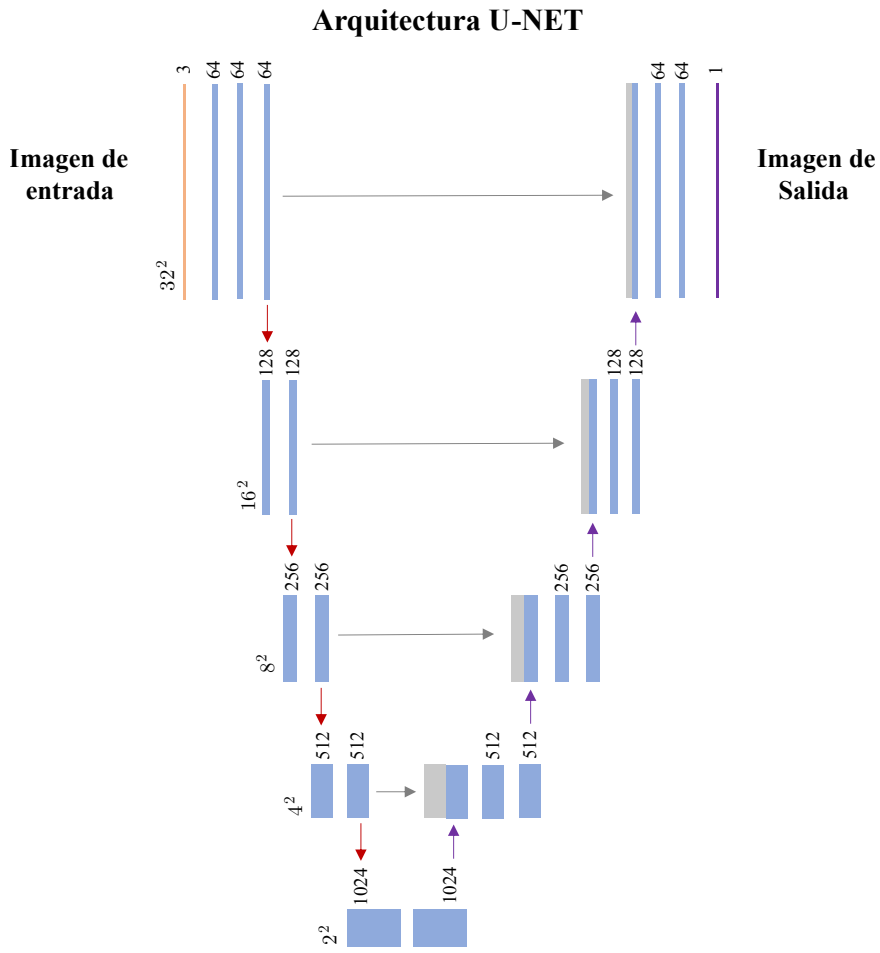


Figura 3.7: Diagrama arquitectura U-NET.

Por otra parte, la arquitectura CNN-LSTM es más sencilla de explicar, puesto que solo se compone de 6 capas:

1. La primera capa tiene 16 filtros, y un núcleo de 1×1.
2. La segunda capa tiene 32 filtros y un núcleo de 3×3.
3. La tercera capa tiene 64 filtros, y un núcleo de 5×5.
4. La cuarta capa tiene 128 filtros, y un tamaño de núcleo de 3×3.
5. La quinta capa tiene 256 filtros, y un tamaño de núcleo de 1×1.
6. La sexta capa simplemente es una operación de convolución en tres dimensiones, con un núcleo de $3 \times 3 \times 3$, de manera totalmente análoga a la convolución en dos dimensiones. Tiene asociada la función de activación σ

Tal y como se había mencionado, los filtros indican que las operaciones de multiplicación de matrices en las celdas LSTM se sustituyen por operaciones de convolución a las imágenes de entrada, pero las operaciones en las puertas de la célula siguen siendo las mismas. El número de parámetros a entrenar es 6.7 millones, y el modelo tiene un tamaño de memoria en disco de 77 MB.

3.2. Preparación *dataset*

Finalmente, se presenta en este apartado el procedimiento para generar las muestras de entrenamiento (*dataset*), esto es, los datos a partir de los cuales las NN calcularán el valor de los parámetros que reduzcan lo máximo posible la *loss function*. En este caso esta acción se corresponde con disminuir el error definido como MSLE entre la predicción de topología realizada por el modelo y la topología obtenida mediante SIMP. Es decir, la preparación del *dataset* no es más que organizar los "ejemplos" a partir de los cuales estos modelos de ML puedan ajustar sus parámetros.

En la Figura 3.5 ya se ha mostrado cuál es la información de entrada y cuál es la información de salida de un modelo U-NET. Por otra parte, en la Figura 3.6 se ha hecho lo propio con el modelo LSTM. Tanto para un modelo como para el otro se han añadido las condiciones de carga en forma de imagen.

Es conveniente recordar que las condiciones de carga de la escala fina provienen del análisis de la escala macro. Por lo tanto, se han de cumplir las condiciones de equilibrio estático

$$\sum F_x = 0, \tag{3.2}$$

$$\sum F_y = 0, \tag{3.3}$$

$$\sum M = 0. \tag{3.4}$$

Para cumplir con el equilibrio, se simplificará el problema suponiendo tensiones lineales en cada cada l de la celda, siguiendo las siguientes ecuaciones

$$t_{x_l} = a_l\, s_l + b_l, \tag{3.5}$$

$$t_{y_l} = c_l\, s_l + d_l \tag{3.6}$$

$$\tag{3.7}$$

donde a_l, b_l, c_l y d_l son coeficientes de las ecuaciones lineales (según eje cartesiano x, e y) de cada cara de la celda, y s_l es la coordenada local de cada cara, que sigue sentido anti-horario.

Por lo tanto, el problema se encuentra definido por un total de 16 parámetros, cuatro por cada cara de la retícula. Sin embargo, teniendo en cuenta las ecuaciones de equilibrio (Ec. 3.2, Ec. 3.3 y Ec. 3.4) los posibles casos de carga están definidos por solo 13 grados de libertad.

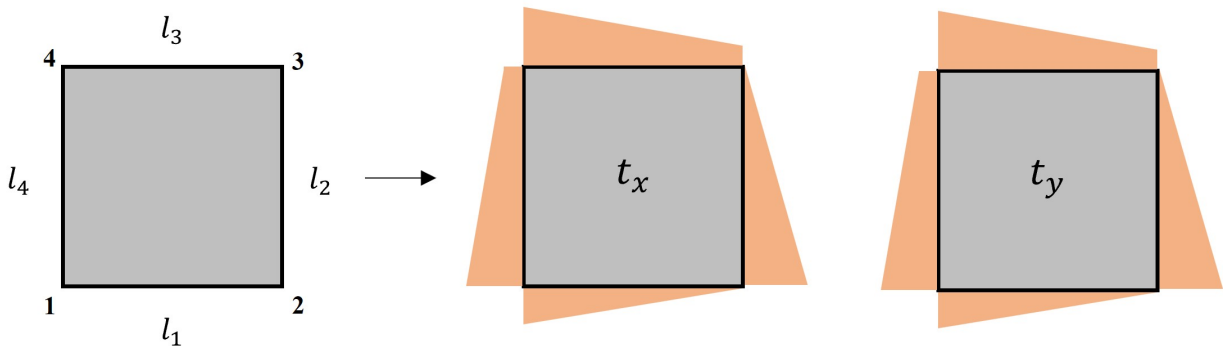


Figura 3.8: Tensiones cartesianas sobre caras de la celda.

En la Figura 3.8 se muestra, por una parte, el orden de las caras del criterio que siguen las Ec. 3.5 y Ec. 3.6; también se marca el origen de coordenadas de la coordenada local en cada cara, de tal manera que, la coordenada local s_1 comienza en 1 y termina en 2, s_2 comienza en 2 y termina en 3, y así sucesivamente. Además, todos los lados tienen la misma longitud.

Las tensiones cartesianas sobre las caras de la celda se transforman en imágenes, colocando en los píxeles del borde el valor de las tensiones. Una representación se encuentra en la Figura 3.9. Es interesante recalcar que el valor de las tensiones en las caras está siempre acotado.

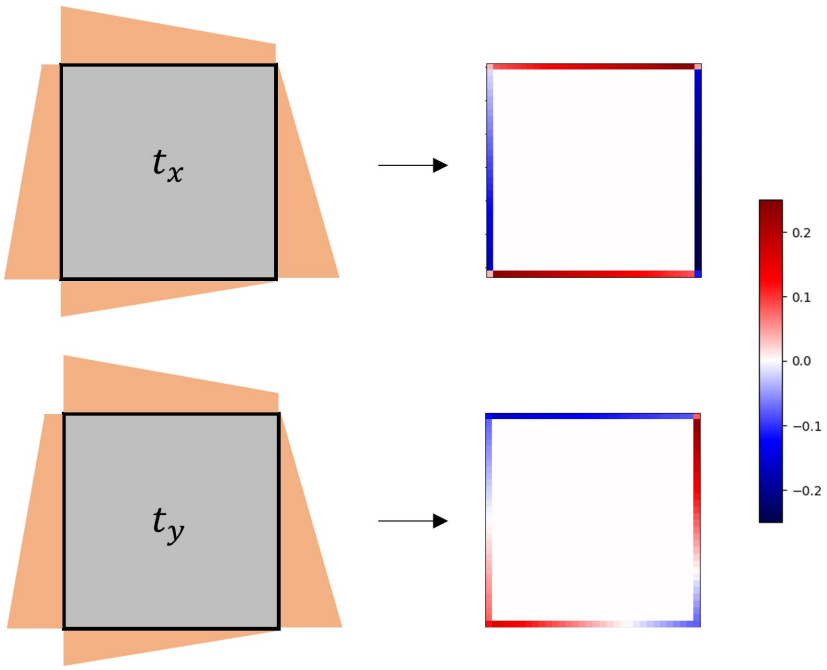


Figura 3.9: Transformación tensiones en imágenes.

En la Figura 3.10 se presentan las topologías de algunas iteraciones del proceso de optimización de un caso de carga de ejemplo. Tal y como se aprecia, la estructura queda más definida a medida que aumenta el número de iteración.

Es importante recordar que la iteración 0 (no mostrada en la Figura 3.10) es un domino donde todos los elementos tienen como densidad relativa la fracción de volumen; en el caso de este trabajo 0.5.

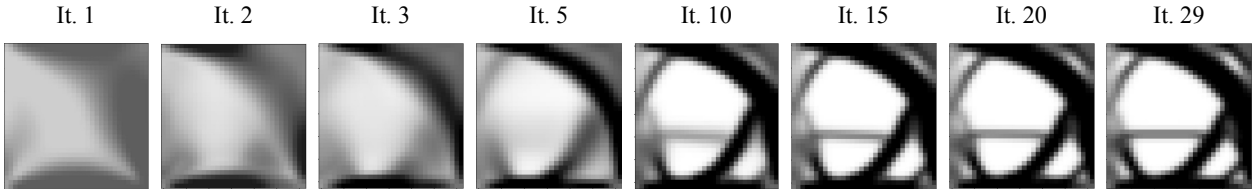


Figura 3.10: Estructuras caso de carga de ejemplo.

Se generan en total 8500 casos de carga, cada uno de ellos con 30 iteraciones, (de la iteración 0 a la iteración 29). A partir de estos casos de carga se crean las muestras tanto para la red U-NET como LSTM.

Se representa de manera visual en la Figura 3.11 la confección de muestras para el entrenamiento de los modelos de U-NET. Cada una de las muestras se encuentra recuadrada. Es fácil ver que, entre la iteración 0 y la iteración 29, se pueden crear 29 muestras.

Las primeras 9 muestras cada caso de carga se dedicarán a entrenar exclusivamente al primer modelo de U-NET, y el resto de muestras se emplearán para entrenar el segundo modelo de U-NET, que genera topologías entre la iteración 9 y la 19.

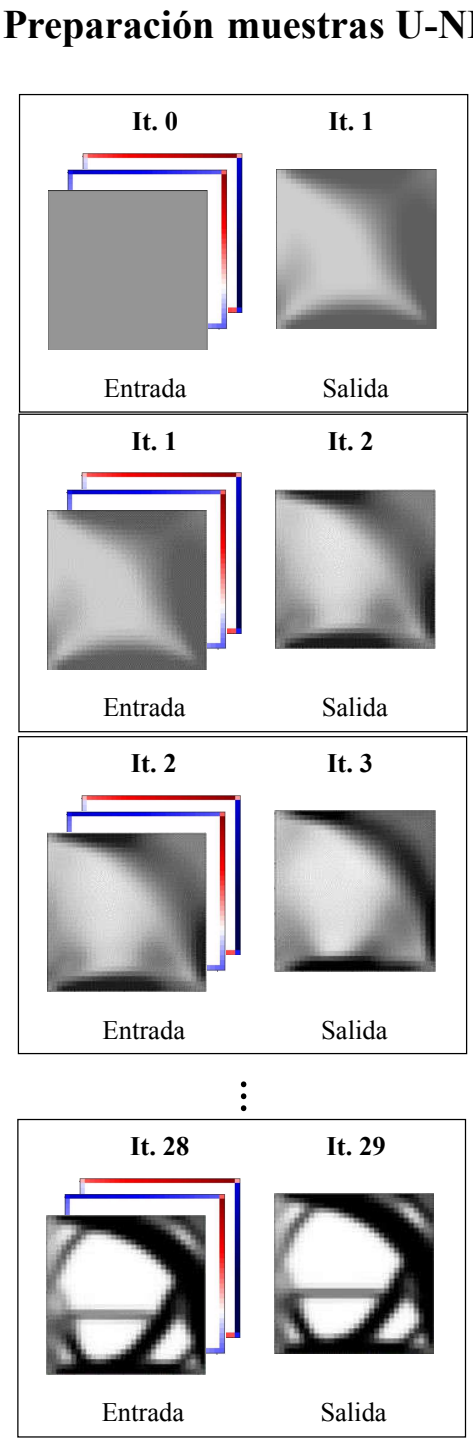


Figura 3.11: Confección muestras para arquitectura U-NET.

Por otra parte, también se representa de manera visual, en la Figura 3.11 la confección de muestras para el entrenamiento del modelo de LSTM. Las muestras se encuentran recuadradas. Para entrenar el modelo LSTM no se toman como muestras las topologías de las iteraciones iniciales, sino que se comienza en la iteración 15. De esta manera, se pueden obtener cinco muestras para entrenar la red LSTM de cada caso de carga.

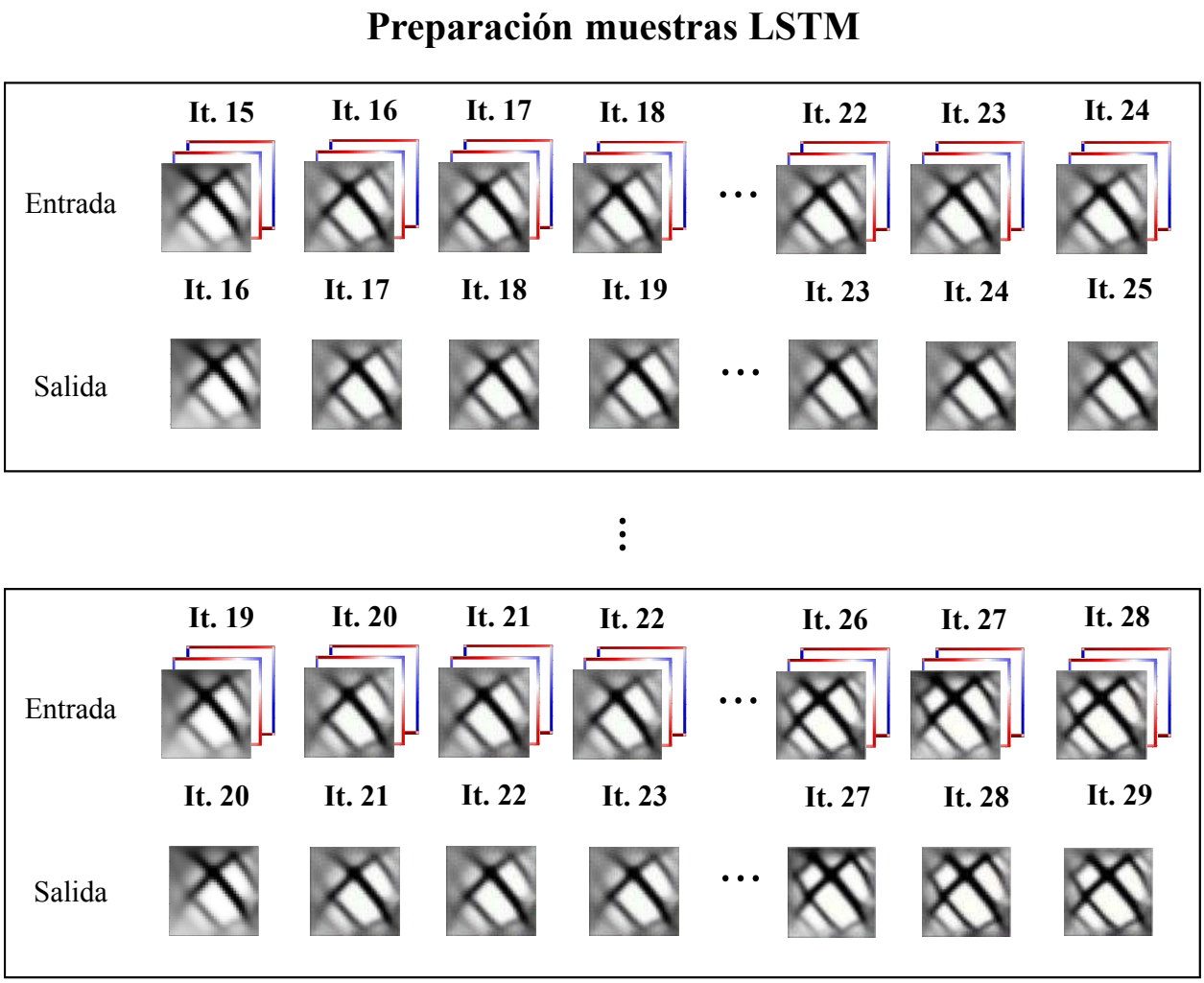


Figura 3.12: Confección muestra para arquitectura LSTM.

Es muy importante apuntar que tanto para el caso de los modelos U-NET como para el modelo LSTM, los casos de carga que generan las muestras de entrenamiento, son diferentes a los casos de carga a partir de los cuales se evalúa la calidad del modelo.

4 | Resultados y discusión

Es conveniente definir un conjunto de criterios que permitan evaluar la calidad de los resultados arrojados por las redes neuronales entrenadas. En este sentido Bernhard *et al.* [39] propusieron cuatro criterios para llevar a cabo este análisis:

- **Análisis de diferencia de densidades entre SIMP y NN**: este criterio compara el resultado obtenido usando el método de redes neuronales con el obtenido mediante la técnica tradicional SIMP, para ello se obtiene la diferencia de las densidades elemento a elemento entre los dos métodos y se calcula el error absoluto medio (MAE). Por consiguiente, si las topologías obtenidas se parecen, el error calculado es pequeño. Este criterio tiene el inconveniente de que, si las topologías no se parecen pero sí que tienen el mismo comportamiento mecánico, puede arrojar un error elevado que no se corresponde con la calidad de la solución.
- **Cálculo *compliance***: este indicador compara el valor de la función objetivo, *compliance* de las topologías obtenidas mediante NN con la *compliance* de las topologías obtenidas mediante SIMP. Esto es interesante puesto que minimizar el valor de esta función es el objetivo del algoritmo de optimización del método SIMP ya que es inversamente proporcional a la rigidez.
- **Comprobación fracción de volumen**: una restricción impuesta en el proceso de optimización topológica es que la fracción de volumen, particularmente en este trabajo, está fijada en 0.5. Por lo tanto es conveniente calcular la densidad media de los resultados obtenidos mediante modelos de NN para comprobar si se respeta, o no, la restricción de volumen.
- **Comparación de tipo cualitativo**: es conveniente realizar una inspección visual de los resultados obtenidos del cual llegar a conclusiones adicionales.

4.1. Análisis casos de carga concretos

En este apartado se analizan los resultados obtenidos mediante los modelos de NN siguiendo los cuatro criterios propuestos. Para ello se generan 50 casos de carga adicionales, y tras una inspección previa, se seleccionan los casos más representativos.

Caso de carga 4

Es conveniente presentar conjuntamente el resultado de la optimización topológica obtenida mediante SIMP con los resultados de los modelos de NN de manera visual. Para ello se confecciona la Figura 4.1 donde se muestra por una parte, los resultados de ciertas iteraciones obtenidos mediante SIMP (en la primera fila), y por otra parte los resultados de las mismas iteraciones obtenidas mediante los modelos entrenados en este proyecto (en la segunda fila).

A simple vista se puede apreciar que los dos métodos dan un resultado casi idéntico para todas las iteraciones, menos la última. En la última iteración se puede apreciar que las topologías son ligeramente distintas, tal y como puede dar cuenta la barra que no está definida en la parte superior izquierda de la topología de NN frente a la topología final generada por SIMP donde sí que se encuentra definida. Si se obtiene la diferencia de densidades

elemento a elemento, y se calcula el error medio entre todos los elementos, se obtiene el error MAE = 10.09 %.

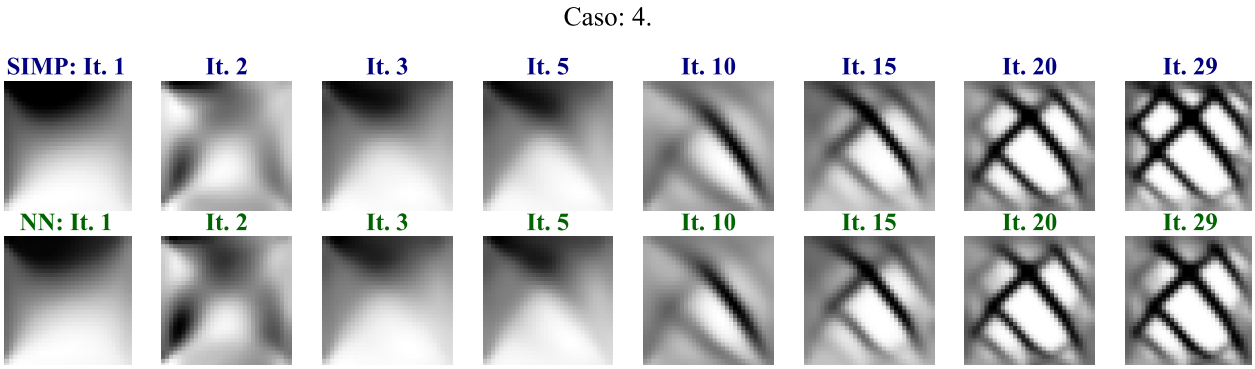


Figura 4.1: Comparación de resultados: SIMP (primera fila) y NN (segunda fila), caso 4.

Por otra parte, se puede calcular el valor de la *compliance* relativa, c/c_0 , y fracción de volumen de la topología de la última iteración comparando los dos métodos. Los resultados obtenidos se muestran en la Tabla 4.1. El error relativo presentado en esta tabla se calcula como

$$\text{Error relativo} = \frac{\text{NN} - \text{SIMP}}{\text{SIMP}} \cdot 100 \tag{4.1}$$

A la vista de la tabla se puede señalar que el error relativo de c/c_0 es muy bajo, dando cuenta que la calidad de la topología obtenida mediante NN es casi igual de buena que la obtenida mediante las técnicas tradicionales. Por otra parte, se aprecia que, dentro de un margen de tolerancia de un 5 %, la técnica NN cumple la imposición de fracción de volumen máxima 0.5.

	NN	**SIMP**	**Error relativo (%)**
c/c_0 [-]	0.525	0.521	0.77 %
Fracción de volumen [-]	0.522	0.508	2.76 %

Tabla 4.1: Resumen valores y error para el caso 4.

A la vista de los resultados, se puede apuntar que este caso es representativo de situaciones donde el modelo NN obtiene una solución muy similar en topología y con buenos resultados en relación con la *compliance*.

Caso de carga 27

Se presenta en la Figura 4.2 las topologías obtenidas mediante SIMP conjuntamente con las obtenidas mediante NN.

Para este caso de carga, se puede apreciar por simple inspección que existe una diferencia muy sustancial entre las topologías obtenidas mediante las dos técnicas. No solo es que la topología obtenida mediante NN se encuentra mucho más definida que la alternativa; también se puede intuir que, de haber continuado, los dos métodos habrían arrojado topologías sustancialmente diferentes. Tanto es así, que exceptuando la barra de la parte izquierda de sendas topologías, el resto de barras tienen una orientación y disposición diferente.

Al calcular el error de diferencia de densidades, elemento a elemento, se obtiene MAE = 30.85 %. Este error de densidades es elevado, como ya cabía esperar.

Caso: 27.

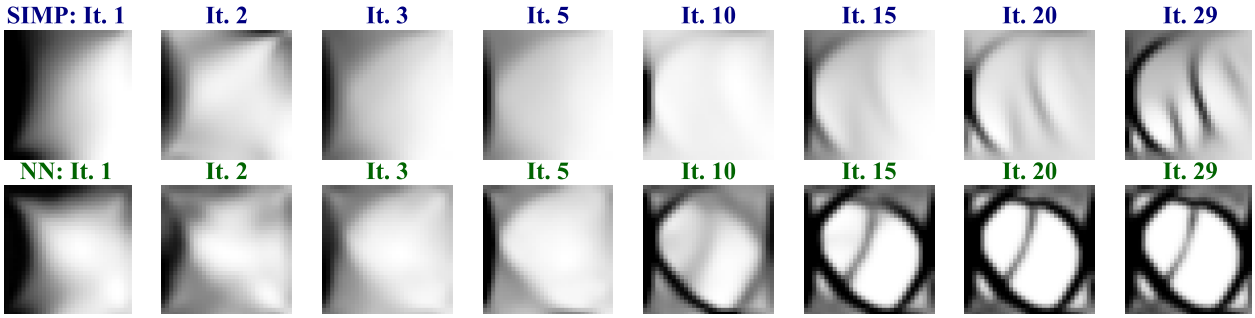


Figura 4.2: Comparación de resultados: SIMP (primera fila) y NN (segunda fila), caso 27.

De nuevo, se puede calcular el valor de *compliance* y fracción de volumen de la topología de la última iteración comparando los dos métodos. Los resultados se muestran en la Tabla 4.2.

	NN	**SIMP**	**Error relativo (%)**
c/c_0 [-]	0.749	0.786	-4.63 %
Fracción de volumen [-]	0.518	0.507	2.17 %

Tabla 4.2: Resumen valores y error para el caso 27.

A la vista de los resultados se puede apuntar que, por una parte, se cumple la restricción de la fracción de volumen, si se tiene en cuenta el margen de tolerancia. Por otra parte, se aprecia un error relativo negativo; esto significa que la *compliance* de la topología obtenida mediante NN es menor que la alternativa tradicional. Esto es muy significativo teniendo en cuenta que el error de diferencia de densidades se encuentra entorno al 30 %.

Si bien es cierto que el método NN arroja un resultado ligeramente mejor, ninguna de los dos consigue minimizar de manera sustancial la función objetivo.

Los resultados obtenidos en este caso de carga dan un indicio de que el proceso de NN sigue un proceso de optimización propio; no se limita a «replicar» los resultados obtenidos mediante el algoritmo tradicional, memorizando los resultados.

Caso de carga 8

Se presentan en la Figura 4.3 las topologías obtenidas mediante SIMP conjuntamente con las obtenidas mediante NN para el caso de carga 8.

Es posible apreciar, comparando de manera visual los resultados que no existe una diferencia significativa en la topología de la iteración final de ambos métodos. De hecho, se pueden encontrar las mismas barras con la misma disposición en las topologías de ambos casos. Si se calcula el error de diferencia de densidades se obtiene MAE = 12.66 %.

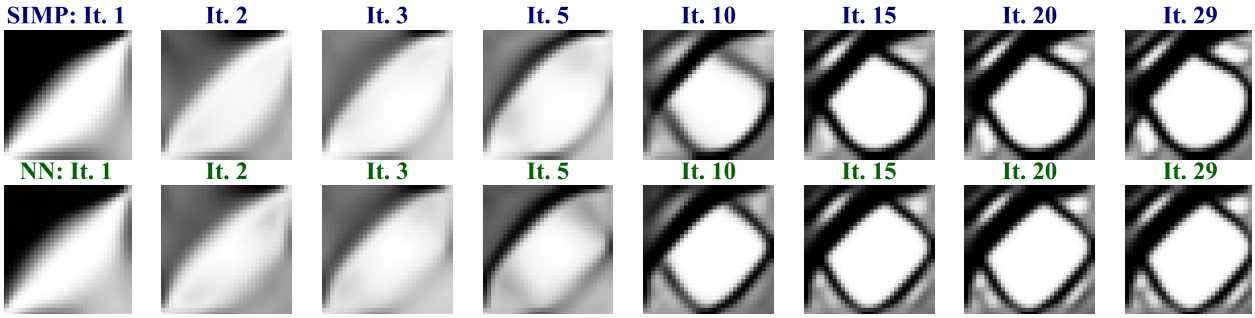


Figura 4.3: Comparación de resultados: SIMP (primera fila) y NN (segunda fila), caso 8.

Por otra parte, se puede calcular el valor de la *compliance* y fracción de volumen de la topología de la última iteración, y se consiguen los resultados que se muestran en la Tabla 4.3.

	NN	**SIMP**	**Error relativo (%)**
c/c_0 [-]	0.403	0.294	37 %
Fracción de volumen [-]	0.517	0.497	4.02 %

Tabla 4.3: Resumen valores y error para el caso 8.

A la vista de los resultados, se cumple de nuevo la restricción de fracción de volumen. Por otra parte, el error relativo de la c/c_0 es elevado. Por consiguiente, la calidad del resultado obtenido mediante NN, en comparación al método tradicional, no es buena. Esto resulta notable teniendo en cuenta que el resultado, tal como respalda el valor de MAE es similar desde un punto de vista topológico. Sin embargo, no se ha de perder de vista que ambos métodos logran reducir sustancialmente la función objetivo, consiguiendo rigidizar la estructura.

Del análisis de este caso de carga se puede señalar una deficiencia del método de optimización mediante NN y es que, durante el proceso de entrenamiento no se penaliza al modelo por la obtención de resultados de topologías similares pero con rigideces menores.

Caso de carga 9

Finalmente, se presentan en la Figura 4.4 los resultados de las topologías obtenidas mediante los dos métodos estudiados para el caso de carga 9.

En este caso de carga se aprecia que los resultados *procedentes* del el método NN son diferentes, a nivel visual, de los obtenidos mediante el otro método. Si se calcula el error de diferencia de densidades se obtiene MAE MAE = 17.66 %.

Caso: 9.

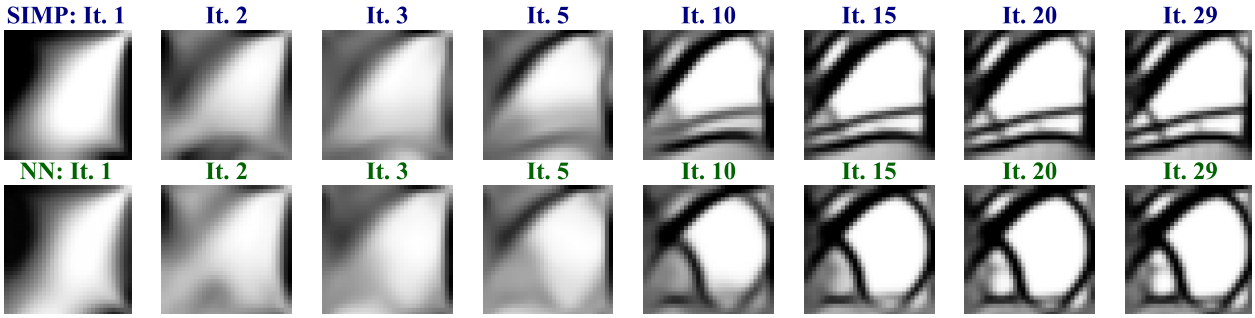


Figura 4.4: Comparación de resultados: SIMP (primera fila) y NN (segunda fila), caso 9.

Se calcula el valor de la *compliance* y la fracción de volumen de la topología de la última iteración, obteniendo los resultados que se muestran en la Tabla 4.4.

	NN	**SIMP**	**Error relativo (%)**
c/c_0 [-]	0.633	0.324	95.50 %
Fracción de volumen [-]	0.520	0.507	4.02 %

Tabla 4.4: Resumen valores y error para el caso 9.

A la vista de los resultados, se cumple de nuevo la restricción de fracción de volumen. Por otra parte, el error relativo de la *compliance* adimensional es muy elevado. Por consiguiente, la calidad del resultado obtenido mediante NN no es bueno. Esto resulta notable teniendo en cuenta que el resultado, tal como respalda el valor de MAE es similar, elemento a elemento.

A pesar de que el resultado no es bueno cabe señalar que, de nuevo, el método NN logra disminuir la función objetivo, rigidizando la estructura.

4.2. Análisis global de la función objetivo

El análisis de casos de carga concretos es útil para comprender el funcionamiento de los modelos de NN empleados, y para evaluar la naturaleza de cada topología a nivel particular. No obstante, es necesario hacer un examen desde una perspectiva global de la calidad de los modelos utilizados, que además, pueda dar criterios sólidos a partir de los cuales se puedan tomar decisiones en lo referente a las arquitecturas de las redes implicadas. Para ello se generan 500 casos de carga adicionales, sobre los cuales realizar diferentes análisis.

Se realiza un análisis de carácter global que da cuenta de la evolución de c/c_0 en función del número de iteración, tal y como ya se hizo en la Figura 3.4. En esta ocasión se añadirá al análisis la evolución del método de redes neuronales implementado. Además, con el objetivo de enriquecer el análisis, se calculará también la *compliance* relativa de métodos de redes neuronales con otras configuraciones probadas pero que no se han mostrado en la Memoria.

Para este primer análisis, además de la configuración de redes neuronales defendida en este trabajo, denominada de ahora en adelante como (23M+23M+LSTM G), se analizará la evolución de c/c_0 de las topologías obtenidas con una única red U-NET de 23 millones de parámetros, y por otra parte se repetirá el análisis con otra red de U-NET de 7 millones de parámetros.

Se calcula la *compliance* relativa asociada a cada iteración para cada una de las diferentes técnicas de optimización. Esto se hace de manera promediada para los 500 casos de carga. Se representa la evolución de este indicador promedio frente al número de iteración en la Figura 4.5.

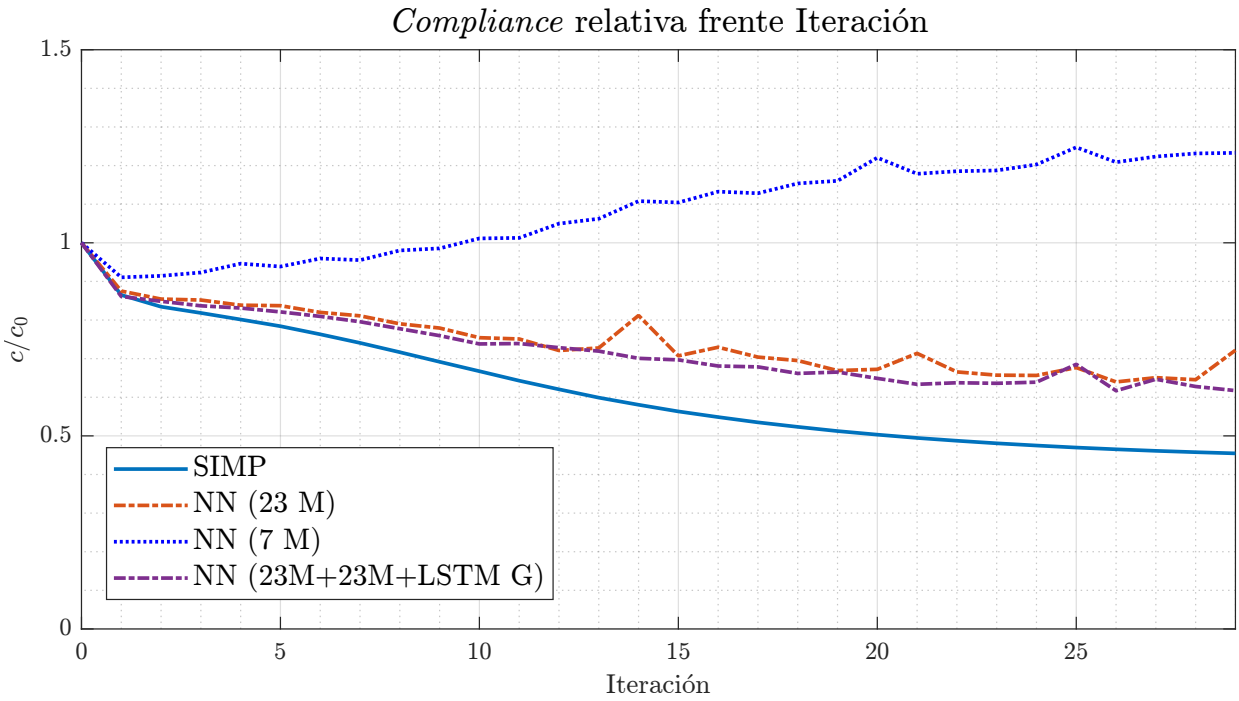


Figura 4.5: *Compliance* relativa frente número de Iteración para cada técnica.

En la Figura 4.5 se aprecia la existencia de cuatro evoluciones de *compliance* relativa diferentes.

Por un lado, en color azul sólido se aprecia la evolución de c/c_0 de la metodología SIMP, que es la más baja entre las cuatro. Por otro lado, se aprecia que la estrategia que peores resultados arroja es el método que emplea una red U-NET con 7 millones de parámetros, con color azul punteado. De hecho, en lugar de rigidizar la estructura, en promedio, hace lo contrario.

Ahora, se puede centrar el análisis sobre los dos métodos restantes; por una parte, el método que implementa una sola red U-NET de 23 millones de parámetros, representado en naranja, demuestra buenos resultados durante las primeras 13 iteraciones, sin embargo, a partir de ahí se aprecia la existencia de inestabilidades. El método defendido en este proyecto, representado en color morado punteado, tiene un buen comportamiento a lo largo de las 25 primeras iteraciones, y a partir de ahí, también se aprecia un pequeño rizado. Así y todo, el método defendido es el que crea topologías de mayor calidad, tal y como indica la *compliance*, entre todas las redes probadas.

Para mantener la consistencia con el apartado anterior, se calcula el error relativo promedio entre la *compliance* relativa de una topología generada por NN frente a SIMP. El error relativo obtenido para la iteración 9 es 14.15 %, para la iteración 19 un 36 % y para la iteración 29 un 39.04 %. Este indicador permite comparar rápidamente la calidad entre modelos, pero no da una idea de la calidad de un modelo concreto, como se verá a continuación.

El análisis promedio de la *compliance* relativa ha permitido observar patrones de carácter global entre las diferentes propuestas de métodos basados en redes y proporciona un indicador simple para evaluar la calidad de los modelos utilizados. Sin embargo, es preciso recordar que realizar promediados destruye gran cantidad de información de carácter estadístico esencial para delimitar el ámbito de aplicabilidad de los modelos.

Se propone generar un gráfico de dispersión en el que cada punto representa un caso de carga. El valor en el eje x del punto es el valor de *compliance* relativa de la topología generada por el método SIMP, mientras que el valor del eje y es el valor de su *compliance*

relativa de la topología generada mediante el modelo (23M+23M+LSTM G). Esta gráfica se genera en la iteración 9, (Figura 4.6), para la iteración 19, (Figura 4.7), y para la iteración 29, (Figura 4.8).

En la Figura 4.6 se muestra el gráfico de dispersión de 500 casos de carga en la iteración 9. También se muestra una línea con pendiente 1 y que pasa por el origen de coordenadas. Esta línea es útil, puesto que permite delimitar de manera visual si la topología obtenida mediante NN es de calidad o no; esto es, si el marcador se encuentra por encima de la línea significa que la topología tiene asociada una mayor *compliance* que la topología obtenida con SIMP. Si al contrario, el marcador se encuentra por debajo de la línea, significa que la topología obtenida mediante el método NN es de mayor calidad que la asociada al método SIMP.

A la vista de la Figura 4.6 se puede apreciar que la *compliance* de las topologías de la iteración 9 obtenidas mediante los dos métodos son similares, puesto que todos los marcadores están agrupados.

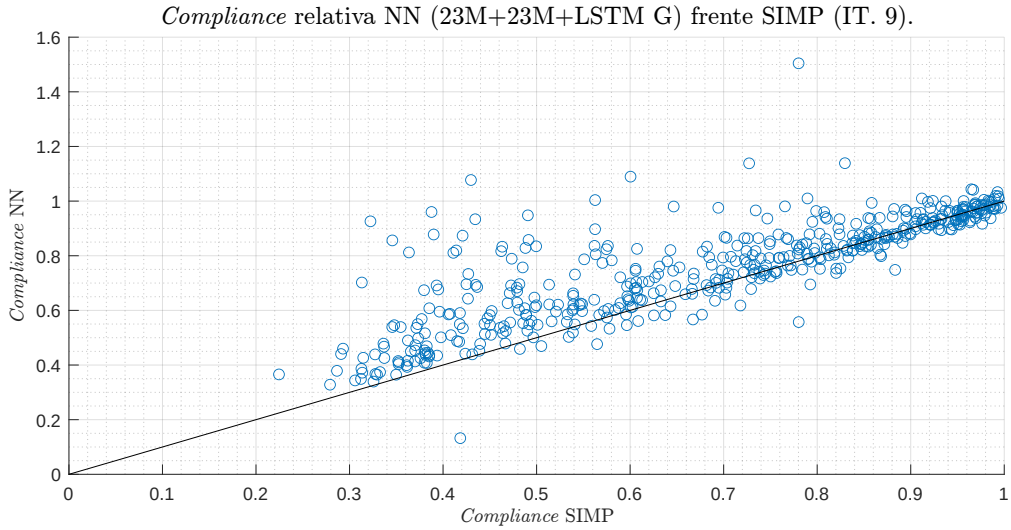


Figura 4.6: Gráfico de dispersión en iteración 9.

Se puede repetir el análisis para la iteración 19 a la vista de la Figura 4.7. En esta ocasión los marcadores se encuentran alejados de la línea de pendiente 1.

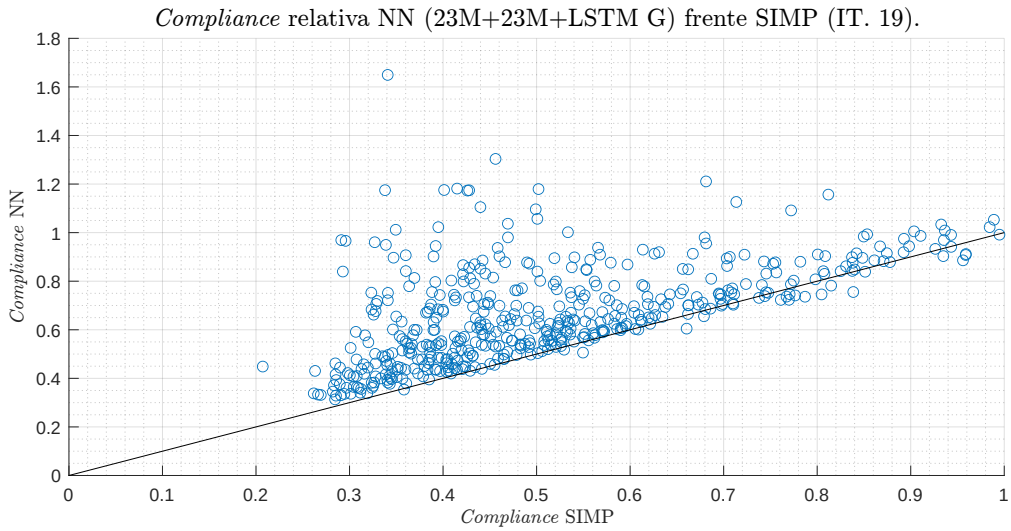


Figura 4.7: Gráfico de dispersión en iteración 19.

En la Figura 4.8 se calcula la gráfica de dispersión de este indicador para la última iteración.

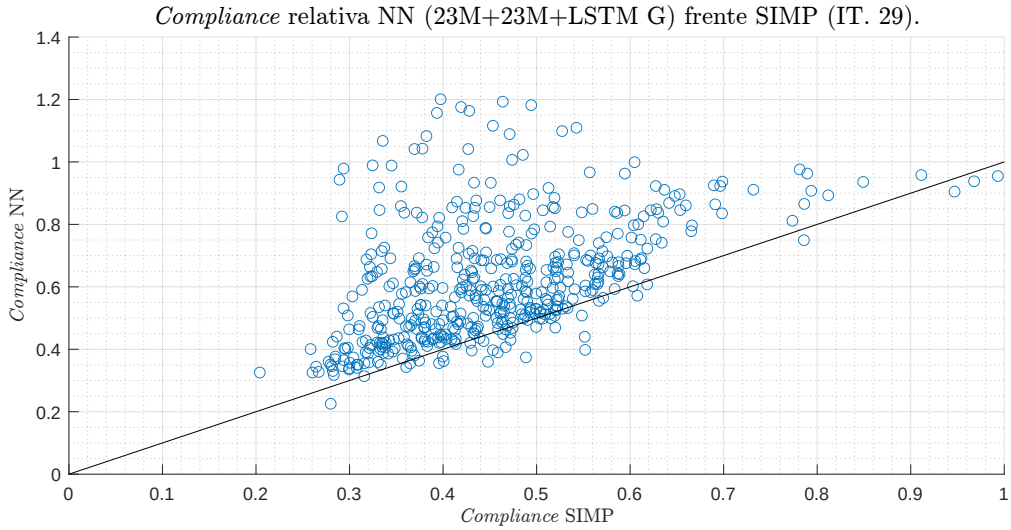


Figura 4.8: Gráfico de dispersión en iteración 29.

A simple vista, las gráficas de dispersión mostradas son un herramienta para estudiar una gran cantidad de datos y observar tendencias;

- La dispersión de los resultados de las iteraciones altas es mayor que que la dispersión de la iteración 9.
- Hay pocos casos de carga en el que el desempeño de NN sea mejor que la técnica SIMP.
- Hay pocos casos de carga en el que c/c_0 calculado aumente más que el valor 1, esto es, que el modelo neuronal obtenga una topología peor que la topología inicial (donde todo el dominio está formado por elementos con densidad 0.5).

Para concretar estas ideas se calculan valores estadísticos de interés, mostrados en la Tabla 4.5.

	It. 9	**It. 19**	**It. 29**
c/c_0 de NN es mayor que 1:	3.88 %	4.91 %	3.48 %
c/c_0 de NN es menor que c/c_0 SIMP:	28.42 %	5.72 %	**8.58** %
Media c/c_0:	0.759	0.665	0.617
Mediana c/c_0:	0.794	0.641	0.589
Desviación típica c/c_0:	0.186	0.197	0.189
Tercer cuartil:	0.917	0.788	0.724

Tabla 4.5: Resumen datos de carácter estadístico de *compliance* relativo.

A la vista de la Tabla 4.5 se pueden cuantificar datos concretos:

En primer lugar, se demuestra que el modelo de NN defendido es seguro, puesto que ocurre con poca frecuencia el hecho de que el modelo de NN aumente la *compliance* de la iteración final en vez de reducirla (3.48 %).

Por otra parte, también es poco habitual que del modelo NN arroje un resultado con menos *compliance*, que SIMP. Para la topología final eso solo sucede un 8.58 % de las ocasiones, sin embargo, este porcentaje aumenta respecto al 5.72 % de las ocasiones en que la *compliance* relativa de las topologías obtenidas mediante NN es menor que SIMP en la iteración 19. Esto da cuenta de la utilidad que tiene la utilización de la última red LSTM.

Además, el valor de la mediana de la *compliance* relativa es menor que el valor de la media para la iteración final. Esta situación está provocada por la presencia de valores *outliers* que tienen un impacto alto en el valor del media, pero que representan pocos casos.

En este sentido, se muestra la Figura 4.9, donde se representa el diagrama de cajas que da cuenta de la *compliance* relativa de las topologías obtenidas mediante NN.

En este diagrama se aprecia cómo, para la iteración final, la mitad de las topologías tienen una c/c_0 menor que 0.589, mientras que la otra mitad de casos tienen una c/c_0 de entre 0.589 y 1.2. Por otra parte, para la iteración final, el tercer cuartil se encuentra en 0.724; es decir, el 75 % de los casos se optimizarán por debajo de ese valor.

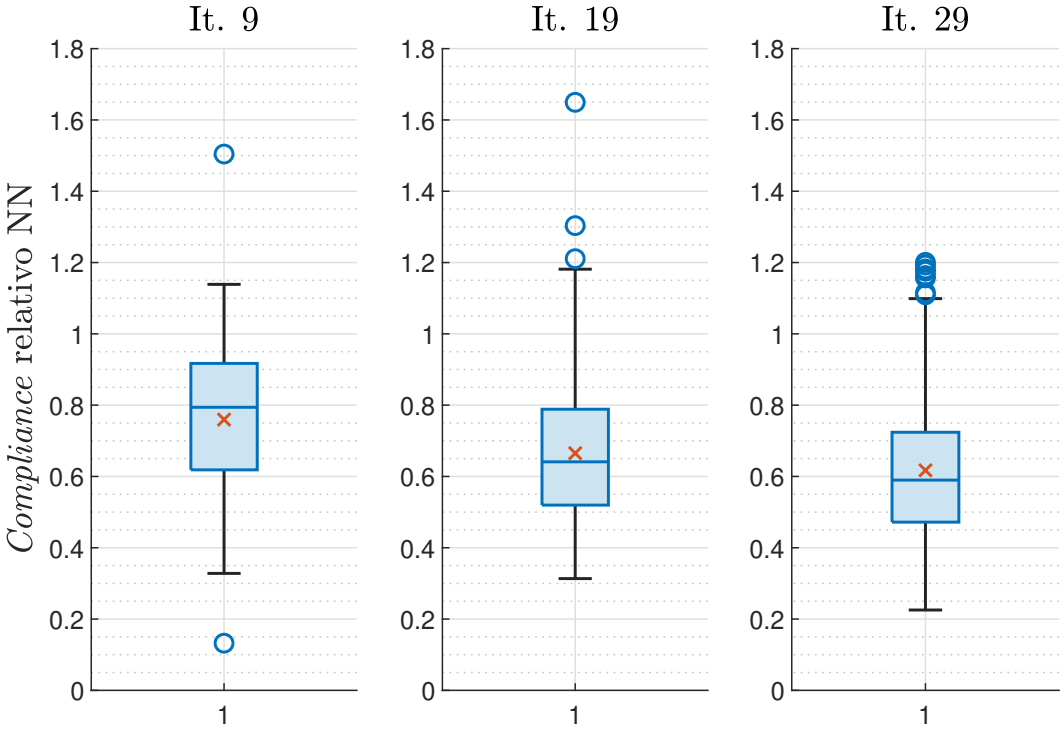


Figura 4.9: Diagrama de caja y bigotes de *compliance* relativo de NN.

En síntesis, parece haberse detectado una deficiencia propia del proceso de optimización topológica mediante NN, esta es que existen casos que se pueden denominar «conflictivos» que, a pesar de ser reducidos en cantidad (ver *outliers* de Figura 4.9), tienen un gran impacto en los indicadores de calidad del modelo.

5 | Conclusiones

Tras el desarrollo del proyecto de optimización topológica de celdas mediante IA se ha llegado a una serie de conclusiones, que están relacionadas con la metodología, con la calidad de los resultados, la aplicabilidad, y las posibles extensiones.

Se ha logrado crear un modelo de IA que supone la disminución de coste computacional en la optimización del problema multinivel presentado.

Para ello, el método de optimización creado se ha especializado en la optimización de la escala fina al haberse conseguido desacoplar eficazmente el problema macro del problema fino mediante el uso de una aproximación a tracciones lineales en las caras del dominio de cada celda.

Se ha acotado el problema a resolver ya que tanto los mapas de densidades, como las condiciones de carga se han tipificado, lo que ha hecho posible abordar el problema mediante el uso de NN. En relación con estas NN, se ha realizado su elección y se ha determinado su ámbito de aplicación con base en las características concretas del problema y en la evolución de la función *compliance* para multitud de casos de carga.

Por otra parte, se ha diseñado de manera correcta el conjunto de muestras de entrenamiento para las NN, seleccionando de manera apropiada la información de multitud de casos de carga.

Y se han definido varios criterios para evaluar la calidad del modelo obtenido. De estos análisis se ha desprendido que el modelo defendido es robusto, puesto que da como resultado topologías optimizadas para la mayor parte de casos de carga, y en muy pocas ocasiones empeoran el resultado.

Esto posibilita la utilización de la herramienta como una solución aproximada en escenarios donde prima la obtención rápida de soluciones sobre la precisión exacta de dichas soluciones.

5.1. Trabajo futuro

El proyecto llevado a cabo ha demostrado que la utilización de herramientas de IA en el proceso de optimización topológico es un campo con mucho potencial por desarrollar. Tanto es así que la literatura disponible acerca de la utilización de IA en el campo de la optimización, como los propios avances publicados acerca de nuevas herramientas de IA aumenta de manera veloz.

Esto hace que este campo sea muy fértil, por consiguiente, existan numerosos caminos por explorar. Tras la realización de este proyecto se proponen las siguientes cuestiones como posibles extensiones a este trabajo:

- Búsqueda parámetros óptimos de las NN. Tal y como se ha mostrado, un cambio entre 7 y 23 millones de parámetros en la red U-NET ha sido suficiente como para mejorar extraordinariamente la calidad del modelo. Este ejemplo abre la posibilidad de hacer estudios de carácter paramétrico que relacionen el número de parámetros y la forma en que se interrelacionan esos parámetros con el rendimiento del modelo.

- Proceso de re-entrenamiento de redes. La sección de análisis de resultados ha mostrado la existencia de casos de carga que podrían denominarse «conflictivos». Esto abre una oportunidad de realizar un estudio detallado de por qué algunos casos de carga son conflictivos y otros no, y qué acciones se pueden tomar para reducir el número de casos conflictivos, por ejemplo, realizar un proceso de re-entrenamiento de las redes empleadas suministrando muchos de estos datos conflictivos.
- Optimización de redes con información de rigidez. En el transcurso de este proyecto se han entrenado las NN mediante minimizando una *loss function*, en este caso, una función de error. Esta estrategia de entrenamiento es sencilla, pero solo da cuenta de la información de las topologías, y no sobre información adicional de carácter físico. Por lo que se propone realizar un estudio acerca de la posibilidad de crear funciones de error personalizadas que permitan dar cuenta de información adicional en el proceso de entrenamiento.
- Generalización del problema a tres dimensiones. El presente proyecto se ha dedicado a estudiar el caso bidimensional, que tiene un interés académico elevado. Sin embargo, desde el punto de vista de la aplicabilidad y el diseño de componentes reales, es necesario resolver el problema en tres dimensiones.

Bibliografía

[1] Jesús Mares, Griselda Stephany Abarca y Enrique Ruby Becerra Montero. *La optimización estructural y sus aplicaciones*. www.boletin.upiita.ipn.mx, ene. de 2021. URL: https://www.boletin.upiita.ipn.mx/index.php/ciencia/916-cyt-numero-82/1889-la-optimizacion-estructural-y-sus-aplicaciones (visitado 04-06-2023).

[2] Behrooz Hassani y Ernest Hinton. *Homogenization and Structural Topology Optimization*. Springer Science & Business Media, dic. de 2012.

[3] Dan M Frangopol y Franklin Y Cheng. *Advances in Structural Optimization*. American Society of Civil Engineers, 1997.

[4] Niels Olhoff y James Taylor. «On Structural Optimization». En: *Journal of Applied Mechanics* 50 (dic. de 1983), págs. 1139-1151. DOI: 10.1115/1.3167196. (Visitado 04-06-2023).

[5] APWorks. *Journey of the Lightrider*. Apworks, 2016. URL: https://www.apworks.de/blog-lightrider (visitado 09-06-2023).

[6] D Brackett, I Ashcroft y R Hague. *TOPOLOGY OPTIMIZATION FOR ADDITIVE MANUFACTURING*. Ago. de 2011. URL: http://utw10945.utweb.utexas.edu/Manuscripts/2011/2011-27-Brackett.pdf.

[7] Qatar National Convention Center. *Qatar National Convention Centre [QNCC Qatar]*. QNCC, 2023. URL: https://www.qncc.qa/.

[8] Jihong Zhu, Weihong Zhang y Liang Xia. «Topology Optimization in Aircraft and Aerospace Structures Design». En: *Archives of Computational Methods in Engineering* (abr. de 2015). DOI: 10.1007/s11831-015-9151-2.

[9] Grégoire ALLAIRE et al. «The Homogenization Method for Topology Optimization of Structures: Old and New». En: *Interdisciplinary Information Sciences* 25 (2019), págs. 75-146. DOI: 10.4036/iis.2019.b.01.

[10] Martin P. Bendsøe y Ole Sigmund. *Topology Optimization*. Springer Berlin Heidelberg, 2004. DOI: 10.1007/978-3-662-05086-6.

[11] Dassault systemes. *3DS.COM/SIMULIA Abaqus Topology Optimization Module Nonlinear Structural Optimization for Improved, Rapid Product Design Overview*. URL: https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/SIMULIA/RESOURCES/SIMULIA-Abaqus-Topology-Optimization-Module.pdf (visitado 06-07-2023).

[12] Altair Engineering. *Análisis estructural y optimización avanzada| Altair OptiStruct*. www.altair.com.es. URL: https://www.altair.com.es/optistruct/ (visitado 06-06-2023).

[13] Lars Krog, Alastair Tucker y Gerrit Rollema. *APPLICATION OF TOPOLOGY, SIZING AND SHAPE OPTIMIZATION METHODS TO OPTIMAL DESIGN OF AIRCRAFT COMPONENTS*. 2011. (Visitado 10-06-2023).

[14] Àlex Ferrer Ferré et al. «Vademecum-based approach to multi-scale topological material design». En: *Advanced modeling and simulation in engineering sciences* 3 (dic. de 2016), págs. 1-22. DOI: 10.1186/s40323-016-0078-4. URL: https://upcommons.upc.edu/handle/2117/89387 (visitado 10-07-2023).

[15] Mikel Barral Poveda. «Optimización topológica continua a 2 niveles en mecánica estructural». En: (2019).

[16] Jun Wu et al. «Infill Optimization for Additive Manufacturing – Approaching Bone-like Porous Structures». En: *IEEE Transactions on Visualization and Computer Graphics* 24 (feb. de 2018), págs. 1127-. DOI: 10.1109/TVCG.2017.2655523.

[17] Gobierno de España. *Qué es la Inteligencia Artificial*. planderecuperacion.gob.es. URL: https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr.

[18] IBM. *What is Artificial Intelligence (AI)?* www.ibm.com, 2023. URL: https://www.ibm.com/topics/artificial-intelligence.

[19] Andrés González. *¿Qué es Machine Learning? – Cleverdata*. Cleverdata.io, 2019. URL: https://cleverdata.io/que-es-machine-learning-big-data/.

[20] Simon J.D. Prince. *Understanding Deep Learning*. MIT Press, 2023. URL: https://udlbook.github.io/udlbook/.

[21] Rebekka V Woldseth et al. «On the use of artificial neural networks in topology optimisation». En: 65 (oct. de 2022). DOI: 10.1007/s00158-022-03347-1. (Visitado 22-11-2022).

[22] Klaus-Jrgen Bathe. «Finite Element Method». En: *Wiley Encyclopedia of Computer Science and Engineering* (jun. de 2008). DOI: 10.1002/9780470050118.ecse159.

[23] Virginia Williams. *Multiplying matrices in O(n 2.373) time*. 2014. URL: https://people.csail.mit.edu/virgi/matrixmult-f.pdf (visitado 28-06-2023).

[24] Solidworks. *Método SIMP para optimización de topología - 2023 - Ayuda de SOLIDWORKS*. help.solidworks.com, 2023. URL: https://help.solidworks.com/2023/spanish/SolidWorks/cworks/c_simp_method_topology.htm?verRedirect=1 (visitado 02-06-2023).

[25] Damla Ozkapici y Ulas Yaman. «Tiling of Microstructures According to the Density Values of SIMP Topology Optimization». En: *Procedia Manufacturing* 51 (ene. de 2020), págs. 1033-1037. DOI: 10.1016/j.promfg.2020.10.145.

[26] Jun-ichi Koga, Jiro Koga y Shunji Homma. *Checkerboard Problem to Topology Optimization of Continuum Structures*. 2013. arXiv: 1309.5677 [cs.CE].

[27] Bert Carremans. *Handling overfitting in deep learning models*. Medium, ene. de 2019. URL: https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e.

[28] Álvaro Artola Moreno. «Clasificación de imágenes usando redes neuronales convolucionales en Python». Tesis de mtría. 2019. URL: https://hdl.handle.net/11441/89506.

[29] Li Yin. *A Summary of Neural Network Layers*. Machine Learning for Li, ene. de 2019. URL: https://medium.com/machine-learning-for-li/different-convolutional-layers-43dc146f4d0e.

[30] Kaveh Shahhosseini. *In Keras what is the difference between Conv2DTranspose and Conv2D*. Stack Overflow, ago. de 2021. URL: https://stackoverflow.com/questions/68976745/in-keras-what-is-the-difference-between-conv2dtranspose-and-conv2d (visitado 30-07-2023).

[31] Christopher Olah. *Understanding LSTM Networks*. Github.io, ago. de 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[32] Google. *Keras | TensorFlow Core*. TensorFlow, ene. de 2022. URL: https://www.tensorflow.org/guide/keras?hl=es-419.

[33] Ivan Sosnovik e Ivan Oseledets. En: *Russian Journal of Numerical Analysis and Mathematical Modelling* 34.4 (2019), págs. 215-223. DOI: doi:10.1515/rnam-2019-0018. URL: https://doi.org/10.1515/rnam-2019-0018.

[34] Yiquan Zhang et al. *A deep Convolutional Neural Network for topology optimization with strong generalization ability*. arXiv.org, mar. de 2020. DOI: 10.48550/arXiv.1901.07761. URL: https://arxiv.org/abs/1901.07761.

[35] Qiyin Lin et al. «Investigation into the topology optimization for conductive heat transfer based on deep learning approach». En: *International Communications in Heat and Mass Transfer* 97 (2018), págs. 103-109. ISSN: 0735-1933. DOI: https://doi.org/10.1016/j.icheatmasstransfer.2018.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0735193318301593.

[36] Hunter T. Kollmann et al. «Deep learning for topology optimization of 2D metamaterials». En: *Materials & Design* 196 (2020), pág. 109098. ISSN: 0264-1275. DOI: https://doi.org/10.1016/j.matdes.2020.109098. URL: https://www.sciencedirect.com/science/article/pii/S026412752030633X.

[37] Diab W. Abueidda, Seid Koric y Nahil A. Sobh. «Topology optimization of 2D structures with nonlinearities using deep learning». En: *Computers & Structures* 237 (2020), pág. 106283. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2020.106283. URL: https://www.sciencedirect.com/science/article/pii/S0045794920300869.

[38] Cheng Qiu, Shanyi Du y Jinglei Yang. «A deep learning approach for efficient topology optimization based on the element removal strategy». En: *Materials & Design* 212 (2021), pág. 110179. ISSN: 0264-1275. DOI: https://doi.org/10.1016/j.matdes.2021.110179. URL: https://www.sciencedirect.com/science/article/pii/S0264127521007346.

[39] *TopoGAN topology optimization with generative adversarial networks*. Abr. de 2021.

Parte II

Planos, pliego de condiciones y presupuesto

A | Planos

En el presente proyecto no se han incluido planos puesto que el desarrollo del mismo implica la creación metodologías y programas informáticos, y no de un producto de carácter físico.

B | Pliego de condiciones

Alcance

La finalidad del presente escrito es recoger las exigencias técnicas, de ejecucion, económicas, y legales que son necesarias para llevar a cabo correctamente el Trabajo de Fin de Grado, de manera que por una parte se eviten interpretaciones distintas a las deseadas, y por otra se establezcan adecuadamente las relaciones entre el proyectista y las demás partes.

El proyecto "Generación de prediseños estructurales de retículas topológicamente optimizadas mediante el uso de inteligencia artificial" consta de tres documentos: La Memoria, el Pliego de Condiciones y el Presupuesto.

Condiciones generales

En esta sección se detallan las condiciones de carácter general que rigen las responsabilidades y derechos de las partes implicadas en el desarrollo del presente proyecto. Por otro lado, también se hará referencia a las condiciones técnicas en las que se determina el material con que se ha realizado el trabajo.

Condiciones facultativas

Este proyecto nace impulsado por el Instituto de Ingeniería Mecánica y Biomecánica (I2MB), y cuenta con la ayuda y supervisión de los docentes José Manuel Navarro Jiménez y Antolín Martínez Martínez. Ahmad Saleh Walie es un alumno de la Escuela Técnica Superior de Ingeniería del Diseño que asume llevar el proyecto a cabo, y se asegura abordar los siguientes puntos:

- Aplicar las competencias vinculadas al grado universitario, de forma que se exhiban los conocimientos y habilidades adquiridos a lo largo de los estudios.
- Garantizar el cumplimiento de la normativa establecida por la Universitat Politècnica de València y la Escuela Técnica Superior de Ingeniería del Diseño en lo referente a los Trabajos Finales de Grado.
- Asumir la autoría, y responder por la originalidad del proyecto, sometido a las leyes vigentes sobre propiedad intelectual o industrial.
- Asistir a tutorías destinadas a guiar al estudiante en los aspectos estructurales del trabajo, tales como objetivos, metodología, bibliografía, formato, y también las relacionadas con el propio proceso de elaboración y presentación del proyecto.
- Acatar los plazos establecidos de acuerdo con la normativa, así como notificar a los tutores sobre avances e implementar las correcciones sugeridas.

Por otra parte, el alumno accede a los siguientes puntos:

- La utilización de diversas herramientas, como uno de los servidor del departamento de Ingeniería Mecánica y Biomecánica para lanzar los casos.
- Utilización del software que sea oportuno.

Condiciones materiales

El material que emplea este proyecto puede ser entendido como los casos de carga que son generados por un programa externo y que han de reunir las siguiente características:

- Cada caso de carga ha de ser generado mediante un método de optimización topológico basado en densidad, esto es, con valores de densidad comprendidos entre 0 y 1.
- Este método de optimización ha de tener protección frente a materiales en disposición de «isla».
- Cada caso de garga ha de contener, como mínimo, treinta iteraciones, para poder entrenar las redes neuronales empleadas.
- La condición de carga de cada caso será generada de manera aleatoria variando las constantes de las ecuaciones de la tracciones lineales en cada cara del dominio. Estas constantes estarán comprendidas entre -1 y 1.

Condiciones ejecución

Para poder llevar a cabo los objetivos de este proyecto ha sido necesario hacer uso de varios equipos informáticos, en este caso se ha empleado un ordenador portátil con las características que se muestran en la Tabla 5.1, y se ha tenido acceso al servidor del departamento.

Pieza	Características técnicas
CPU	Intel(R) Core(TM) i7-1280P
RAM	32 GB
GPU	RTX 3050 Ti (4 GB)

Tabla 5.1: Ordenador portátil.

Las condiciones de ejecución implican que se ha de hacer uso de un ordenador de, por lo menos, características similares. Es importante mencionar que esas especificaciones constituyen la cota inferior de lo exigible. Esto es porque la parte crítica del proyecto es la del entrenamiento de los modelos neuronales, y esa parte es fundamental realizarla con una GPU con una gran cantidad de memoria.

Por otra parte es conveniente tener acceso a un servidor con características similares a las que aparece en la Tabla 5.2, en el que lanzar los cálculos sin ocupar el ordenador portátil de trabajo.

Pieza	Características técnicas
CPU	Intel(R) Xeon(TM)
RAM	256 GB
GPU	RTX 1060 Ti (4 GB)

Tabla 5.2: Ordenador servidor.

Condiciones contractuales

En la Memoria se ha presentado el problema del alto coste computacional de la optimización topológica de retículas. Para hacer frente a este problema se ha expuesto tanto el fundamento teórico como la metodología que permiten plantear una solución basada en Redes Neuronales. Finalmente, se han propuesto indicadores para evaluar el grado de eficacia de la solución. Queda reservada la Propiedad Intelectual de la obra escrita.

El cliente deberá abonar el importe especificado en el Presupuesto.

El cliente recibirá a cambio los modelos de Redes Neuronales debidamente entrenados.

Relación trabajo con ODS

El presente trabajo ha propuesto una metodología para reducir el coste computacional del proceso de optimización topológica de estructuras. En la Tabla 5.3 se apunta el grado de alineación que tiene el presente TFG con los Objetivos de Desarrollo sostenible. Como se puede apreciar, los ODS. con un grado de relación más alto son el ODS. 9 y el ODS. 12.

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

Tabla 5.3: Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

En relación con el **ODS. 9**, el presente trabajo hace una aportación a las técnicas de optimización topológica que crean estructuras ligeras y optimizadas, esto busca innovar en el diseño de infraestructuras, que se desea que sean sostenibles, resilientes, y de calidad (ODS. 9.1). Además, el presente trabajo es una muestra, a pequeña escala, de un proyecto que pretende presentar innovaciones de carácter científico. Esta es, en esencia la meta del ODS. 9.5; aumentar la capacidad tecnológica de los sectores industriales aumentando el número de personas que trabajan en desarrollo tecnológico.

Por otra parte, el presente trabajo también está alineado con el **ODS. 12**, puesto que la creación de estructuras más ligeras implica que emplean menos materiales en su fabricación; esto está estrechamente relacionado con el ODS. 12.2. Además, si la estructura es ligera, significa que la energía invertida en su transporte, e instalación también es más reducida. Incluso, el ahorro en el coste computacional está relacionado de manera positiva con el ahorro energético en las computadoras que ejecuten los programas de optimización topológica, puesto que consumen menos electricidad en su funcionamiento.

C | Presupuesto

En este apartado se presentará el presupuesto para la realización del trabajo *Generación de prediseños estructurales de retículas topológicamente optimizadas mediante el uso de inteligencia artificial* como si el proyecto se tratase de un trabajo encargado de un cliente a una consultoría. El presupuesto desempeña un papel fundamental en la planificación del trabajo y estimación económica desde el punto de vista del cliente. Es conveniente expresar el precio total del proyecto como la suma de los costes de los diferentes conceptos, obteniendo así una visión precisa de los recursos necesarios en cada área concreta.

Licencia de *software*

En la realización de este trabajo se han empleado programas informáticos y lenguajes de programación que, en algunos casos, tienen un coste económico asociado. Es importante apuntar que el precio de las licencias se corresponde con las versiones que permiten uso comercial, y no con la versión académica que se ha utilizado en realidad.

Por otra parte, se puede definir un porcentaje de utilización puesto que el ingeniero a cargo de este trabajo también hace uso de las licencias para otros proyectos. Si el tiempo de trabajo anual de un trabajador en España son 1736 horas[1], y el tiempo de trabajo dedicado a este proyecto son 335 horas, entonces la cota superior del tiempo que ocupa el ingeniero en las distintas licencias para este proyecto es aproximadamente un 20 %, donde se tiene en cuenta que las licencias son de uso intransferible.

En la Tabla 5.4 se muestra el precio de los diferentes programas informáticos que se han utilizado, junto con ciertos porcenajes de utilización.

Producto:	**Característica**	**Precio anual**	**Porcentaje utilizado**	**Coste parcial**
MATLAB	Licencia Standard	860 €	20 %	172 €
Office 365	Licencia Empresa Estándar	170 €	20 %	34 €
Python	Código Libre	0 €	20 %	0 €
LaTeX	Código Libre	0 €	20 %	0
			Subtotal	206 €

Tabla 5.4: Coste licencia programas informáticos.

Coste de *hardware*

Durante la realización de este trabajo se ha hecho uso, por una parte, de un ordenador portátil individual en el que se ha confeccionado el documento, y realizado la parte de programación, y por otra parte, un servidor en el que se han generado los casos de carga y se han entrenado los modelos neuronales que se explican en el documento.

Para estimar el coste de utilización de los equipos informáticos es necesario en primer lugar calcular la amortización. Se puede calcular teniendo en cuenta el valor de compra

[1] https://aclaramosdudas.wpcomstaging.com/horas-trabajo-anuales/

menos el valor de venta por un periodo de amortización de cinco años, tal y como se muestra en la Ec. 5.1. El precio del ordenador portátil en el momento de la compra es 1300 €, y el del servidor aproximadamente 4000 €. Por tanto se pueden calcular los precios por hora de utilización en la Tabla 5.5.

$$C_{amort} = \frac{V_i - V_f}{n} = \frac{V_i - 0.2V_i}{n} \tag{5.1}$$

Durante la fase de pruebas del proyecto se han entrenado aproximadamente treinta redes. Si el tiempo de entrenamiento de cada red es de media cuatro horas, significa que el tiempo de utilización del servidor asciende a 120 horas.

Equipo	Característica	Precio por hora	Horas	Coste parcial
Servidor	Servidor departamento I2MB	0.073 €/hora	120	8.76
Portátil	Ordenador auxiliar	0.027 €/hora	300	8.1
			Subtotal	16.86 €

Tabla 5.5: Coste uso de computadores.

Coste de recursos humanos

Se sigue con la analogía de que este trabajo es un producto realizado por una consultoría. En la realización de este trabajo ha intervenido un ingeniero técnico aeroespacial cuyo sueldo medio es de 27312 €[2] brutos anuales. Por otra parte, el ingeniero técnico ha requerido de la asistencia de tres profesores universitarios, cuyo sueldo medio es de 43400 €[3] brutos al año. También ha supervisado el trabajo otro ingeniero, que es estudiante predoctoral, pero se calculará su salario en calidad de ingeniero aeronáutico, 35250 €[4] brutos al año. El salario bruto se divide por las horas trabajadas al año para obtener el coste de trabajo por hora. El tiempo de trabajo, el coste de trabajo, y finalmente el coste parcial se indican en la Tabla 5.6.

Rango	Tiempo de trabajo (h)	Característica	Coste de trabajo	Coste parcial
Ingeniero técnico aeroespacial	335	Desarrollo producto	15.7 €/h	5259 €
Profesor universitario (3 personas)	30	Asesoramiento	25 €/h	2250 €
Ingeniero - Investigador predoctoral	30	Asesoramiento	20.30 €/h	609 €
			Subtotal	8118 €

Tabla 5.6: Coste total recursos humanos.

[2] https://es.talent.com/salary?job=ingeniero+tecnico
[3] https://www.jobted.es/salario/profesor-universidad
[4] https://es.talent.com/salary?job=ingeniero+aeronautico

Costes adicionales

Por un lado se tienen en cuenta multitud de costes indirectos que son difíciles de estimar; coste del alquiler del lugar de trabajo, suministros como luz y agua, servicios como internet u otros relacionados con la red informática, costes ocultos por mantenimiento predictivo y correctivo relacionados con el uso del hardware, gastos de generales asociados con los recursos humanos, impuestos, etc. Todos estos costes se asumirán como el 15 % de la suma de todos los costes anteriores. Así, se ve en la Tabla 5.7 el coste total de cada concepto, y el coste neto total del proyecto.

Suma costes parciales	
Concepto	**Importe**
Coste licencia programas informáticos	206.00 €
Coste uso de computadores	16.86 €
Coste total recursos humanos	8118.00 €
Subtotal:	8340.86 €
Cálculo coste neto proyecto	
Porcentaje coste indirecto (15 %)	1251.13 €
Total neto:	9591.99 €

Tabla 5.7: Coste neto proyecto.

Según la Tabla 5.7, el coste total neto del proyecto es #NUEVE MIL QUINIENTOS NOVENTA Y UN EUROS CON NOVENTA Y NUEVE CÉNTIMOS#

Costes totales

Finalmente, se tiene en cuenta un margen comercial del 10 % sobre el coste neto del proyecto. Sin embargo, el precio que pagará el cliente será el coste neto del proyecto más el margen comercial y se suma el Impuesto del Valor Añadido (IVA). La relación de costes se muestra en la Tabla 5.8.

Presupuesto proyecto	
Concepto	**Importe**
Coste neto proyecto	9591.99 €
Beneficio comercial	
Beneficio comercial sobre coste neto (10 %)	959.19 €
Total proyecto:	10551.2 €
Impuesto sobre el Valor Añadido (IVA)	
IVA (21 %)	2215.75 €
Total comercial:	12766.9 €

Tabla 5.8: Coste total comercial proyecto.

Según la Tabla 5.8, el coste comercial del proyecto asciende a: #DOCE MIL SETECIENTOS SETENTA Y SEIS EUROS CON NOVEINTA CÉNTIMOS#.