



POLITÉCNICA



**UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
AERONÁUTICA Y DEL ESPACIO
GRADO EN INGENIERÍA AEROESPACIAL**

TRABAJO FIN DE GRADO

**Algoritmos basados en redes neuronales convolucionales y
derivadas topológicas para la detección de daño en
estructuras a partir de termografías infrarrojas.**

AUTOR: Anlu CHEN PAN

ESPECIALIDAD: Propulsión Aeroespacial (PA)

TUTORES: María-Luisa RAPÚN BANZO y Manuel PENA RODRÍGUEZ

Febrero de 2020

Índice general

1	Introducción	4
2	Detección de daño en placas metálicas mediante pruebas termográficas	6
2.1	Inspección no destructiva mediante pruebas termográficas.....	6
2.2	Planteamiento teórico del problema térmico.....	8
2.3	Definición de derivada topológica.	11
2.4	Derivada topológica para el problema termográfico.....	11
2.5	Base de datos del experimento.....	17
3	Entorno de ejecución	18
3.1	Software, hardware y librerías.....	18
4	Redes Neuronales Artificiales.	20
4.1	Perceptrones multicapa, capas ocultas.....	21
4.2	Funciones de activación	22
4.3	Función de coste	24
4.4	Optimización, descenso del gradiente	24
4.5	Learning Rate (Ratio de aprendizaje)	26
4.6	Backpropagation	27
4.7	Batch y epoch (lote y época)	28
4.8	Batch Normalization.....	28
4.9	Tipos de redes neuronales profundas.....	29
5	Redes Neuronales Convolucionales (CNNs).....	32
5.1	Capa convolucional	32
5.2	Padding (rellenado)	34
5.3	Convolución de matrices de diferentes dimensiones.	35
5.4	Capa Pooling.....	35

5.5	Ejemplos de redes CNN complejas.....	36
6	Configuración de las Redes Neuronales.....	37
6.1	Características de las bases de datos consideradas.....	37
6.2	Estructura de la red neuronal y sus parámetros.....	38
6.3	Aplicación de las capas convolucionales.....	43
7	Resultados y discusiones.....	45
7.1	Termografías:	46
7.2	Derivadas topológicas:	50
8	Conclusiones y líneas futuras	57
	Bibliografía y referencias.....	59

Capítulo I

Introducción

Este trabajo fin de grado tiene la finalidad de profundizar en la aplicación de las redes neuronales para el análisis de imágenes termográficas obtenidas mediante métodos de inspección y monitoreo no destructivos para la detección de defectos en placas metálicas, que comenzaron con los trabajos de Sergio Colubi Domingo [1] y Ángel Mateo Arenas [2].

Las termografías son imágenes adquiridas con cámaras termográficas que indican la radiación que emiten las diferentes partes de los materiales. Esta radiación puede ser la que emita el propio material por sí mismo o por una excitación externa. Al albergar defectos, las termografías correspondientes a materiales sanos respecto a los defectuosos variarán en consecuencia y esta diferencia de información entre ambas muestras termográficas es la que utilizaremos para el posterior análisis.

La derivada topológica es una herramienta matemática muy potente que puede utilizarse para procesar imágenes termográficas con la finalidad de mejorar la interpretación de dichas imágenes. Mediante esta derivada se puede definir una función indicadora de daño, capaz de clasificar cada punto de la placa inspeccionada como perteneciente o no a una zona dañada (es decir, no solo produce información sobre la cara de la placa que es fotografiada sino también sobre su interior).

Sin embargo en los artículos [3] y [4] se constató que uno de los problemas fundamentales en este tipo de proceso de inspección no destructivo, es su incapacidad de extraer información relativa a la muestra en la dirección de la profundidad, hecho que es claro cuando se trata imágenes termográficas (puesto que solo proporcionan una imagen de la cara de la muestra que es fotografiada), pero no tan obvio en las derivadas topológicas (que proporcionan una imagen correspondiente a toda la muestra inspeccionada, indicando las regiones que podrían estar dañadas). Es por ello por lo que en este trabajo fin de grado, se aplicarán las redes neuronales artificiales para solventar dicho problema de la identificación de la profundidad de los defectos de las muestras.

En la actualidad, las redes neuronales se han convertido en una herramienta muy potente para resolver multitud de problemas de diferentes campos. Este TFG se centrará en concreto en las redes neuronales convolucionales, porque se ha visto que son muy

útiles resolviendo problemas tan complejos como la identificación de enfermedades mediante imágenes biomédicas, identificación de objetos para la conducción autónoma o incluso recientemente, el plegamiento de proteínas [5]. Uno de los factores de este éxito, es debido a su gran capacidad de reconocer patrones en los datos, haciéndolos muy polivalentes para muchas áreas de investigación.

El objetivo de este proyecto es ver el comportamiento de dichas redes convolucionales aplicadas al problema de identificar la profundidad del defecto en el interior de una placa metálica dañada, mediante imágenes termográficas e imágenes termográficas que han sido previamente procesadas utilizando derivadas topológicas.

Este trabajo se estructurará de la siguiente manera:

- Desarrollo y realización de ensayos no destructivos para la detección de daño en materiales mediante pruebas de termografía infrarroja junto a su planteamiento matemático
- Explicación de las Redes Neuronales Artificiales junto a todos los parámetros que incurren en su construcción.
- Profundización de las Redes Neuronales Convolucionales y la función de sus diferentes capas.
- Aplicaciones de las redes construidas a bases de datos tanto de imágenes termográficas como de los valores resultantes tras la derivada topológica.
- Resultados de las predicciones de las redes neuronales y su respectiva discusión.
- Conclusiones y líneas futuras.

Capítulo II

Detección de daño en placas metálicas mediante pruebas termográficas

2.1 Inspección no destructiva mediante pruebas termográficas

Los ensayos con técnicas termográficas consisten en métodos de inspección no destructivos, que miden la radiación que emiten los materiales por sí mismos (método pasivo) o cómo ésta se propaga a lo largo de la muestra cuando se la excita de manera externa (método activo). Esto permite detectar ciertos defectos del material como grietas, discontinuidades, vacíos internos...etc. Este tipo de ensayos no destructivos son esenciales en la industria para garantizar una calidad y seguridad a la hora de la fabricación, algo de vital importancia en el sector aeroespacial.

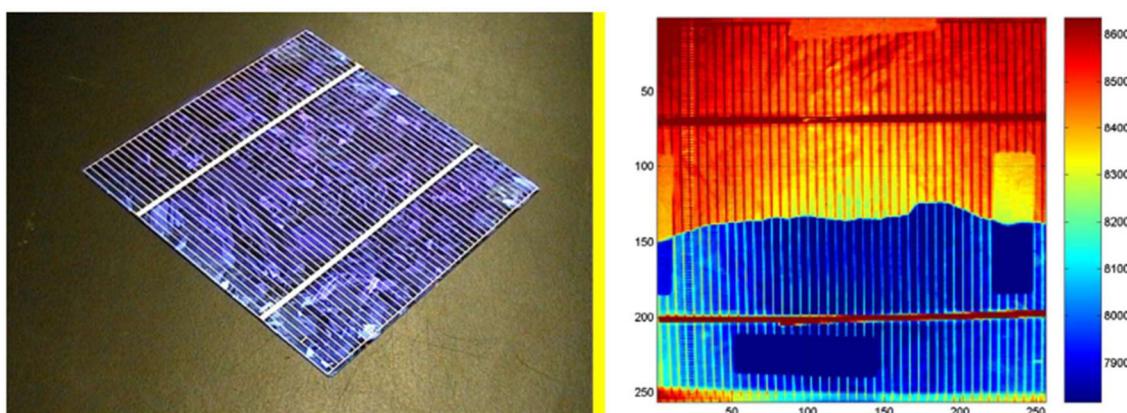


Figura 1: Panel solar agrietado y su imagen termográfica. Fuente: [6]

En la Figura 1 se muestra una placa solar agrietada junto a su imagen termográfica. En la imagen de la derecha, se puede apreciar cómo la técnica termográfica ilustra la fractura interna del material debido a la diferencia de temperaturas.

En este TFG nos centraremos en un método de inspección que consiste en excitar una placa de aluminio bidimensional de manera externa con lámparas térmicas, mientras

que un detector térmico capta los diferentes niveles de radiación de la muestra, véase un esquema del mismo en la Figura 2.

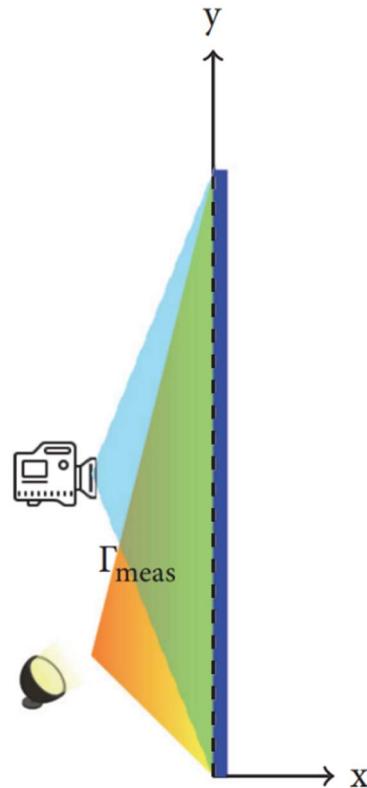


Figura 2: Configuración del experimento. Fuente: [3]

Para conseguir información a partir de estas imágenes termográficas, se procede a realizar el mismo experimento con la placa (la placa que se quiere analizar y que posiblemente tenga algún tipo de defecto) y otro con una placa sana. La diferencia de temperatura captada por estas dos imágenes termográficas es la información que se procesará con las redes neuronales para saber a qué profundidad se encuentran los defectos. También se aplicará la derivada topológica a las termografías como método complementario para la resolución del problema en cuestión. Aunque, como se ha mencionado al principio, esta derivada no muestra la forma y posición de los defectos en el eje de la profundidad, sí puede dar una información extra para encontrar la solución mediante las redes neuronales.

Ante la falta de imágenes termográficas reales, capturadas mediante una cámara termográfica en un experimento real, en este trabajo utilizaremos termografías sintéticas, es decir, generadas numéricamente utilizando un ordenador. Estas termografías sintéticas se han realizado mediante software, con el programa de cálculo FreeFem++, que es un código de software libre [7]. A éstas se les ha añadido un ruido aleatorio para simular la imprecisión que surgiría a la hora de realizar el ensayo en la realidad. Por lo tanto, la red neuronal que se va a construir debe tener la capacidad de distinguir el ruido de la información relevante, para el resolver el problema.

A continuación, se plantea la base teórica para la simulación de este tipo de pruebas termográficas.

2.2 Planteamiento teórico del problema térmico

Como se ha mencionado anteriormente, el problema, extraído de los artículos de los tutores de este proyecto, [3] y [4], consiste en excitar térmicamente una placa bidimensional de aluminio con un defecto en una posición específica en el interior con lámparas modelizadas como radiadores isotrópicos ideales. Se trata de una placa esbelta rectangular bidimensional, donde una de sus longitudes es mucho mayor que la otra (véase la Figura 2). Además, está aislada térmicamente por sus extremos más estrechos mientras el calor se transmite a través de sus extremos más anchos. Se realizan dos tipos experimentos: en el primero se calienta la placa de manera continua y se captura la imagen termográfica; en el segundo, la placa se excita de manera armónica, se captura la imagen termográfica y se realiza un postproceso mediante la derivada topológica.

El núcleo de este trabajo consiste en analizar termografías, por lo tanto, el primer problema a resolver consiste en calcular las temperaturas de la placa sabiendo todos los parámetros involucrados, la posición de las lámparas y el lugar de los defectos. A este problema vamos a denominarlo el problema directo, descrito mediante la ecuación del calor en la placa:

$$\rho c \partial_t T - \nabla \cdot (\kappa \nabla T) = 0 , \quad (2.1)$$

donde ρ , c y κ corresponden a la densidad mísica, el calor específico y la conductividad térmica. El espacio que ocupa la placa está definido mediante la región $\Omega = [0, L_x] \times [0, L_y]$, donde $L_y \gg L_x$. La región que ocupa el defecto está definida mediante Ω_i mientras que la región de la placa sin el defecto se define mediante Ω_e .

Para excitar térmicamente la placa, se usa una lámpara que se modela como un radiador isotrópico localizado en $s = (s_x, s_y)$ y se describe matemáticamente mediante la función

$$q_s(x, y) = \frac{P}{2\pi} \frac{x - s_x}{(x - s_x)^2 + (y - s_y)^2}, \quad (2.2)$$

donde $P > 0$ corresponde a la potencia continua de la lámpara en el caso del experimento de excitación continua. En el caso de la excitación armónica, la lámpara se modela mediante la función:

$$\begin{aligned} q_s(x, y, t) &= \tilde{q}_s(x, y) + \operatorname{Re}(Q_s(x, y)e^{-i\omega t}) = \\ &= \frac{\tilde{P}}{2\pi} \frac{x - s_x}{(x - s_x)^2 + (y - s_y)^2} + \operatorname{Re} \left(\frac{Pe^{-i\omega t}}{2\pi} \frac{x - s_x}{(x - s_x)^2 + (y - s_y)^2} \right) \end{aligned} \quad (2.3)$$

donde $\omega > 0$ es la frecuencia de la excitación, \tilde{q}_s es la distribución estacionaria de la lámpara que tiene una potencia $\tilde{P} > 0$ y P corresponde a la amplitud del término complejo de la potencia.

En el caso estacionario, la distribución de la temperatura solo depende de la variable espacial, es decir, $T_s(\mathbf{x}, t) = T_s(\mathbf{x})$, donde el subíndice "s" enfatiza la dependencia de la temperatura según la posición de la lámpara. Por lo tanto, considerando las excitaciones

térmicas modeladas por radiadores isotrópicos de la fórmula (2.2), el problema térmico se define mediante las siguientes ecuaciones:

$$\begin{aligned}
 \nabla \cdot (k_e \nabla T_s) &= 0, && \text{en } \Omega_e \\
 \nabla \cdot (k_i \nabla T_s) &= 0, && \text{en } \Omega_i \\
 T_s^+ - T_s^- &= 0, && \text{en } \partial\Omega_i \\
 \kappa_e \partial_n T_s^+ - k_i \partial_n T_s^- &= 0, && \text{en } \partial\Omega_i \\
 \partial_y T_s &= 0, && \text{en } y = 0 \\
 \partial_y T_s &= 0, && \text{en } y = L_y \\
 \kappa_e \partial_x T_s - (h + 4\varepsilon\sigma T_{air}^3) T_s &= -(h T_{air} + (\alpha + 3\varepsilon)\sigma T_{air}^4) - \alpha q_s, && \text{en } x = 0 \\
 \kappa_e \partial_x T_s + (h + 4\varepsilon\sigma T_{air}^3) T_s &= h T_{air} + (\alpha + 3\varepsilon)\sigma T_{air}^4, && \text{en } x = L_x
 \end{aligned} \tag{2.4}$$

donde h corresponde al coeficiente convectivo de transferencia de calor entre la placa y el aire y α , la absorbancia de la superficie de la placa. T_s^+ y T_s^- corresponden a las temperaturas en el límite exterior e interior de la región Ω_i . Los parámetros ε y σ se tratan de la emisividad de la superficie y el constante de Stefan-Boltzmann respectivamente.

En el caso armónico, en el que la excitación se modela por la función $q_s(x, y, t)$ definida en (2.3), la temperatura se comportará también de manera análoga, siendo de la forma $T_{s,\omega}(x, t) = \tilde{T}_s(x) + \mathcal{R}e(T_{s,\omega}(x) e^{-i\omega t})$, donde $\tilde{T}_s > 0$ corresponde a la media y $T_{s,\omega}(x)$ es la amplitud del término complejo. Por lo tanto, las ecuaciones que modelan el problema en el caso armónico son las siguientes:

$$\begin{aligned}
 \nabla \cdot (\kappa_e \nabla T_{s,\omega}) + i\omega \rho_e c_e T_{s,\omega} &= 0, && \text{en } \Omega_e \\
 \nabla \cdot (\kappa_i \nabla T_{s,\omega}) + i\omega \rho_i c_i T_{s,\omega} &= 0, && \text{en } \Omega_i \\
 T_{s,\omega}^+ - T_{s,\omega}^- &= 0, && \text{en } \partial\Omega_i \\
 \kappa_e \partial_n T_{s,\omega}^+ - \kappa_i \partial_n T_{s,\omega}^- &= 0, && \text{en } \partial\Omega_i \\
 \partial_y T_{s,\omega} &= 0, && \text{en } y = 0 \\
 \partial_y T_{s,\omega} &= 0, && \text{en } y = L_y \\
 \kappa_e \partial_x T_{s,\omega} - (h + 4\varepsilon\sigma T_{air}^3) T_{s,\omega} &= -\alpha Q_s, && \text{en } x = 0 \\
 \kappa_e \partial_x T_{s,\omega} + (h + 4\varepsilon\sigma T_{air}^3) T_{s,\omega} &= 0, && \text{en } x = L_x.
 \end{aligned} \tag{2.5}$$

En este caso, la temperatura $T_{s,\omega}$, depende tanto de la posición de la lámpara $s = (s_x, s_y)$ como de su frecuencia ω .

La distribución de la temperatura o amplitud compleja de ésta que se obtiene, mediante las ecuaciones (2.4) y (2.5), en la cara exterior $x = 0$, se denominará como T^{med} y se corresponde con la temperatura que captaría la cámara termográfica. En caso de disponer de termografías experimentales, no sería necesario resolver el problema directo, ya que T^{med} sería un dato. Sin embargo, dado que para este TFG no se dispone de datos experimentales, las termografías utilizadas se han generado resolviendo numéricamente el problema (2.4) en la situación estacionaria, o el problema (2.5) en el caso armónico.

El problema que interesa resolver en el presente TFG es el problema inverso. En este problema se conoce la temperatura medida, T^{med} , en $x = 0$ y se busca conocer a partir de esta temperatura la presencia de defectos, y en su caso, determinar su posición, tamaño y forma. Se trataría entonces de encontrar los defectos Ω_i tales que al resolver el problema directo (2.4) o (2.5) se verificara que $T_s = T^{med}$ en $x = 0$ (en el caso estacionario) o $T_{s,\omega} = T^{med}$ en $x = 0$ (en el caso armónico).

En este trabajo se estudian dos posibles alternativas. Una de ellas consiste simplemente en comparar la termografía T^{med} correspondiente a la placa inspeccionada con la termografía que se obtendría en una placa sana. Se trata, por tanto, de comparar dos funciones en la superficie $x = 0$ de la placa (véase la imagen de la Figura 3). Obviamente, a partir de esta información, sin utilizar de manera combinada con otro tipo de estrategia, lo único que se podría determinar es la altura a la que se encuentra el defecto, pero no su profundidad.

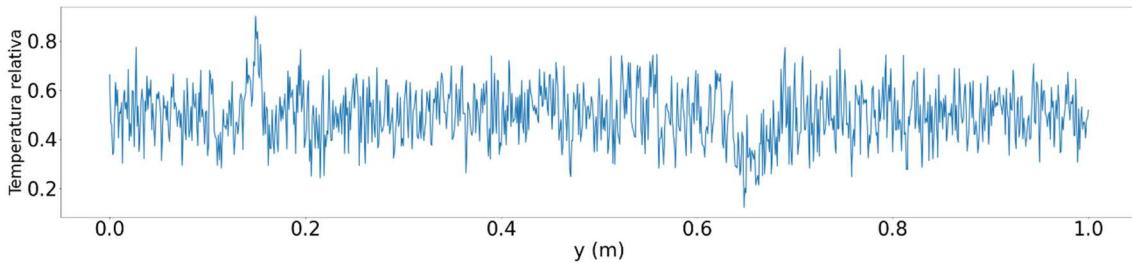


Figura 3: Termografía de la placa de ensayo excitada con una lámpara.

La otra alternativa, consiste en utilizar una herramienta adicional, la derivada topológica, que se explica en el siguiente apartado. Esta herramienta se utilizó en los artículos [3] y [4] para procesar las termografías con la finalidad de mejorar la interpretación de las mismas, consiguiendo detectar la posición del defecto de manera precisa en cuanto a su altura, pero no en cuanto a la profundidad. En las figuras 4 y 5, se puede ver esta apreciación.

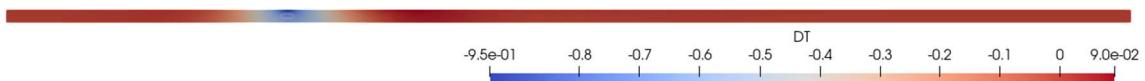


Figura 4: Valores de la derivada topológica en toda la placa.

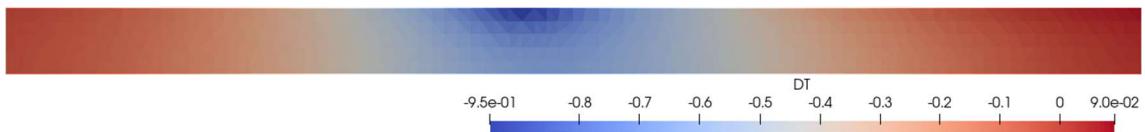


Figura 5: Ampliación de la derivada topológica de la Figura 4 en la zona del defecto.

En la Figura 4, se ha utilizado la derivada topológica para procesar los datos recogidos en las termografías. Se trata de un método numérico que sirve para procesar los datos termográficos proporcionando un mapa de color donde las zonas en las que se alcanzan los valores negativos más pronunciados (colores azules en la Figura 4) indican las zonas dañadas. Tras procesar las termografías con esta herramienta, se puede apreciar

claramente una zona anómala debido al defecto. La Figura 5 corresponde una ampliación de dicha zona, pero a simple vista, no es posible apreciar a qué profundidad se encuentra el defecto.

A continuación, se presentará la base teórica de dicha derivada topológica.

2.3 Definición de derivada topológica.

La derivada topológica, introducida en 1999 en el artículo [8] y explicada en el anterior trabajo [1] es una de las herramientas que va a ayudar a detectar los defectos en las muestras. Ésta representa las variaciones de un funcional cuando el dominio de éste se ve perturbado en su topología. Por lo tanto, es capaz de identificar las alteraciones de la muestra respecto al original, resaltando así, la localización de los defectos en el material.

La derivada topológica de un funcional $J(\mathcal{R})$ es un campo escalar D_T que mide la sensibilidad del funcional cuando una bola de tamaño infinitesimal $B_\epsilon(x)$ de radio ϵ se sitúa en cada punto $x \in \mathcal{R}$. Esto proporciona una expansión asintótica en cada punto x según la siguiente formulación:

$$J\left(\mathcal{R} \setminus \overline{B_\epsilon(x)}\right) = J(\mathcal{R}) + D_T(x)\mathcal{V}(\epsilon) + o(\mathcal{V}(\epsilon)), \quad \text{con } \epsilon \rightarrow 0, \quad (2.10)$$

donde $\mathcal{V}(\epsilon)$ es una función monótona creciente cuyo límite, cuando $\epsilon \rightarrow 0^+$, es igual a cero. Para el problema que nos ocupa, como se indica en los artículos [3,4], esta función es $\mathcal{V}(\epsilon) = \pi\epsilon^2$, es decir, coincide con el área de la bola. El objetivo es agrupar los puntos x donde $D_T(x)$ obtenga los valores negativos más pronunciados, ya que son los puntos que hacen que el decrecimiento del funcional sea máximo. Por lo tanto, el conjunto de defectos se aproximará como:

$$\Omega_{apr} := \left\{ x \in \Omega; D_T(x) < (1 - C) \min_{y \in \Omega} D_T(y) \right\} \quad (2.11)$$

donde el parámetro C , comprendido entre 0 y 1, se puede calibrar en función de la sensibilidad del problema.

2.4 Derivada topológica para el problema termográfico.

Al tener unos resultados con ruido, es muy posible que no exista ningún objeto Ω_i para el que se verifiquen las condiciones anteriores $T_s = T^{med}$ (ni siquiera al considerar los defectos reales se verifican las igualdades). Por ello, se va a relajar el planteamiento de manera que se minimice el cuadrado de la diferencia entre la temperatura T^{med} y la obtenida en el problema, es decir se va a buscar los objetos Ω_i que minimicen las siguientes funciones:

$$J_s(\Omega \setminus \overline{\Omega}_i) = \frac{1}{2} \int_{\{x=0\}} |T_s - T_s^{med}|^2 dl \quad (2.6)$$

en el caso estacionario, o

$$J_{s,\omega}(\Omega \setminus \overline{\Omega}_i) = \frac{1}{2} \int_{\{x=0\}} |T_{s,\omega} - T_{s,\omega}^{med}|^2 dl \quad (2.7)$$

para el caso armónico. Se buscará entonces los objetos que minimicen estas funciones utilizando la derivada topológica.

Cuando se realizan varios experimentos con más de una lámpara o frecuencia, es conveniente promediar la contribución individual de cada experimento, para lo que consideraremos funciones de la forma:

$$J(\Omega \setminus \overline{\Omega}_i) = \sum_{j=1}^{N_{lamp}} \beta_j J_{s_j}(\Omega \setminus \overline{\Omega}_i) \quad (2.8)$$

en el caso estacionario, o

$$J(\Omega \setminus \overline{\Omega}_i) = \sum_{j=1}^{N_{lamp}} \sum_{k=1}^{N_{freq}} \beta_{jk} J_{s_j, \omega_k}(\Omega \setminus \overline{\Omega}_i) \quad (2.9)$$

para el caso armónico, siendo β_j y β_{jk} los factores que ponderan la contribución de cada experimento y que dependerá de la derivada topológica de cada una de las funciones J_{s_j} o J_{s_j, ω_k} .

Para evaluar la derivada topológica D_T únicamente tendremos que resolver dos problemas: uno directo y otro adjunto, que tienen lugar en una placa sin defectos. Es decir, solamente con la distribución de la temperatura de la superficie, es posible la resolución del problema y no es necesaria la información sobre el número, ubicación o tamaño de los defectos desconocidos.

- **Caso estacionario:**

La derivada topológica D_T^S de la función de coste J_S , definida en la ecuación (2.6), para el caso estacionario puede calcularse mediante la siguiente expresión:

$$D_T^S(x) = \frac{2\kappa_e(\kappa_e - \kappa_i)}{\kappa_e + \kappa_i} \nabla T_S^0(x) \cdot \nabla V_S^0(x), \quad x \in \Omega \quad (2.12)$$

donde T_S^0 corresponde a la solución del problema de la placa sin defectos:

$$\begin{aligned} \nabla \cdot (\kappa_e \nabla T_S^0) &= 0, && \text{en } \Omega, \\ \partial_y T_S^0 &= 0, && \text{en } y = 0, \\ \partial_y T_S^0 &= 0, && \text{en } y = L_y, \end{aligned} \quad (2.13)$$

$$\kappa_e \partial_x T_S^0 - (h + 4\varepsilon\sigma T_{air}^3) T_S^0 = -(h T_{air} + (\alpha + 3\varepsilon)\sigma T_{air}^4) - \alpha q_s, \quad \text{en } x = 0,$$

$$\kappa_e \partial_x T_S^0 + (h + 4\varepsilon\sigma T_{air}^3) T_S^0 = h T_{air} + (\alpha + 3\varepsilon)\sigma T_{air}^4, \quad \text{en } x = L_x,$$

y V_S^0 corresponde a la solución del problema adjunto asociado, que de nuevo tiene lugar en una placa sin defectos:

$$\nabla \cdot (\kappa_e \nabla V_S^0) = 0, \quad \text{en } \Omega,$$

$$\begin{aligned}
 \partial_y V_s^0 &= 0, & \text{en } y = 0, \\
 \partial_y V_s^0 &= 0, & \text{en } y = L_y, \\
 \kappa_e \partial_x V_s^0 - (h + 4\epsilon\sigma T_{air}^3) V_s^0 &= T_s^{med} - T_s^0, & \text{en } x = 0, \\
 \kappa_e \partial_x V_s^0 + (h + 4\epsilon\sigma T_{air}^3) V_s^0 &= 0, & \text{en } x = L_x.
 \end{aligned} \tag{2.14}$$

- **Caso armónico:**

La derivada topológica $D_T^{s,\omega}$ de la función de coste $J_{s,\omega}$, definida en la ecuación (2.7), para el caso armónico viene dada por la siguiente fórmula:

$$\begin{aligned}
 D_T^{s,\omega}(x) = & \left(\frac{2\kappa_e(\kappa_e - \kappa_i)}{\kappa_e + \kappa_i} \nabla T_{s,\omega}^0(x) \cdot \overline{\nabla V_{s,\omega}^0(x)} \right. \\
 & \left. - i\omega(\rho_e c_e - \rho_i c_i) T_{s,\omega}^0(x) \overline{V_{s,\omega}^0(x)} \right), \quad x \in \Omega,
 \end{aligned} \tag{2.15}$$

donde $T_{s,\omega}^0$ corresponde a la solución del problema de la placa sin defectos:

$$\begin{aligned}
 \nabla \cdot (\kappa_e \nabla T_{s,\omega}^0) + i\omega \rho_e c_e T_{s,\omega}^0 &= 0, & \text{en } \Omega, \\
 \partial_y T_{s,\omega}^0 &= 0, & \text{en } y = 0, \\
 \partial_y T_{s,\omega}^0 &= 0, & \text{en } y = L_y, \\
 \kappa_e \partial_x T_{s,\omega}^0 - (h + 4\epsilon\sigma T_{air}^3) T_{s,\omega}^0 &= -\alpha Q_s, & \text{en } x = 0, \\
 \kappa_e \partial_x T_{s,\omega}^0 + (h + 4\epsilon\sigma T_{air}^3) T_{s,\omega}^0 &= 0, & \text{en } x = L_x.
 \end{aligned} \tag{2.16}$$

y $V_{s,\omega}^0$ corresponde a la solución del problema adjunto:

$$\begin{aligned}
 \nabla \cdot (\kappa_e \nabla V_{s,\omega}^0) - i\omega \rho_e c_e V_{s,\omega}^0 &= 0, & \text{en } \Omega, \\
 \partial_y V_{s,\omega}^0 &= 0, & \text{en } y = 0, \\
 \partial_y V_{s,\omega}^0 &= 0, & \text{en } y = L_y, \\
 \kappa_e \partial_x V_{s,\omega}^0 - (h + 4\epsilon\sigma T_{air}^3) V_{s,\omega}^0 &= T_s^{med} - T_s^0, & \text{en } x = 0, \\
 \kappa_e \partial_x V_{s,\omega}^0 + (h + 4\epsilon\sigma T_{air}^3) V_{s,\omega}^0 &= 0, & \text{en } x = L_x.
 \end{aligned} \tag{2.17}$$

En el caso estacionario, se puede apreciar que la ecuación (2.12) únicamente depende de la conductividad interior κ_i mediante el factor multiplicador $2\kappa_e(\kappa_e - \kappa_i)/(\kappa_e + \kappa_i)$. Por lo tanto, lo único relevante es si $\kappa_e < \kappa_i$ o si $\kappa_e > \kappa_i$, ya que solamente interesa saber los valores más negativos de D_T^s .

En el caso armónico, la dependencia de los parámetros es más compleja, puesto que la expresión (2.15) corresponde a una combinación lineal de dos términos, $\nabla T_{s,\omega}^0 \cdot \overline{\nabla V_{s,\omega}^0}$ y $T_{s,\omega}^0 \overline{V_{s,\omega}^0}$, que dependen de los parámetros interiores. Si se desconocen dichos parámetros, se puede calcular la sensibilidad considerando a estos términos como funciones indicadoras de daño.

Finalmente, puesto que las derivadas topológicas de los funcionales J_s y $J_{s,\omega}$ consisten en combinaciones lineales de derivadas topológicas de funcionales individuales, en el artículo [3] se propone utilizar los siguientes pesos en las definiciones (2.8) y (2.9):

$$\beta_j = \frac{1}{|\min_{y \in \Omega} D_T^{s_j}(y)|} \quad (2.18)$$

$$\beta_{jk} = \frac{1}{|\min_{y \in \Omega} D_T^{s_j, \omega_k}(y)|}$$

Esta elección de pesos evita que se pierda información de frecuencias o posiciones de lámparas (como en el caso $\beta_j = 1$ o $\beta_{jk} = 1$), puesto que de esta manera se garantiza que $\min_{y \in \Omega} \beta_j D_T^{s_j}(y) = -1$ y $\min_{y \in \Omega} \beta_{jk} D_T^{s_j, \omega_k}(y) = -1$.

En la Figura 7, extraída del artículo [9], se ilustra la aplicación de las técnicas termográficas, descritas anteriormente, a una placa con los defectos situados de la forma que se indica en la Figura 6.

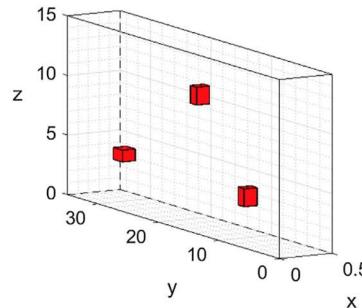


Figura 6: Ubicación de los defectos de una placa. Fuente: [9]

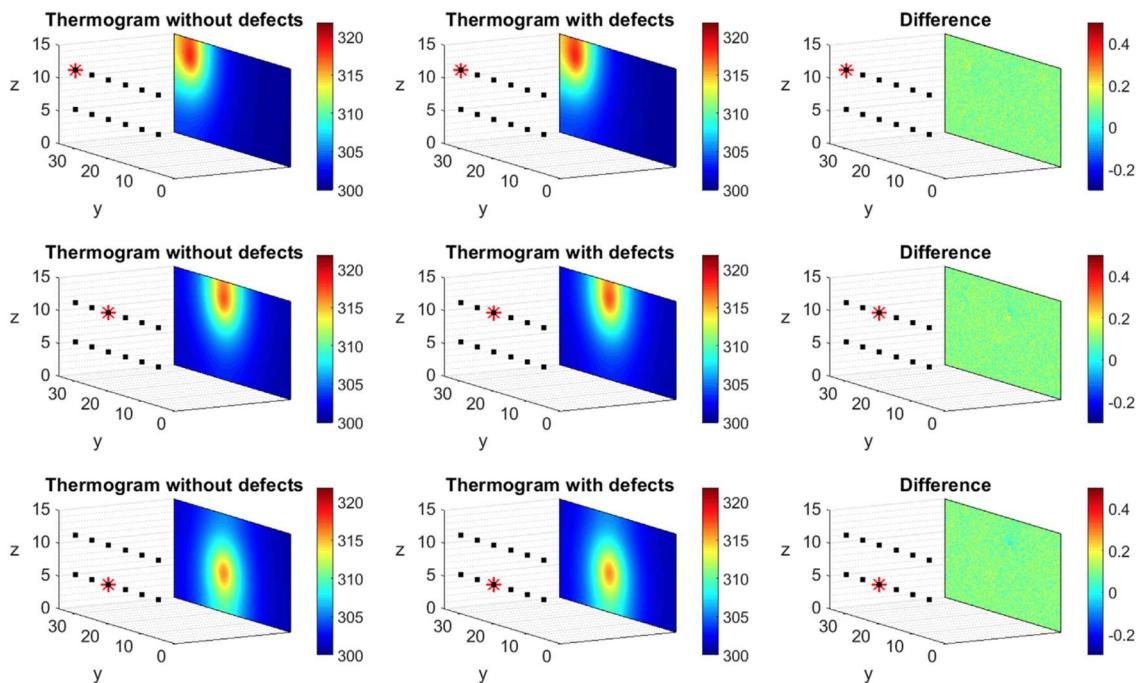


Figura 7: Aplicación de técnicas termográficas en una placa con defectos. Fuente: [9]

En la Figura 7 se puede observar en la primera columna, la medición de las temperaturas de una placa sin defectos con lámparas en distintas posiciones; en la segunda columna se realiza la misma medición con una placa con defectos y finalmente en la tercera columna se muestra la diferencia de temperatura que aparecen entre ambas mediciones. Como se puede observar, comparando ambas termografías, no es posible identificar con certeza la posición de los defectos mostrados en la Figura 6. Por ello, con el fin de llegar a un resultado, en el mismo artículo [9], se utiliza la derivada topológica para procesar estas imágenes termográficas, obteniendo los resultados que se muestran en la Figura 8.

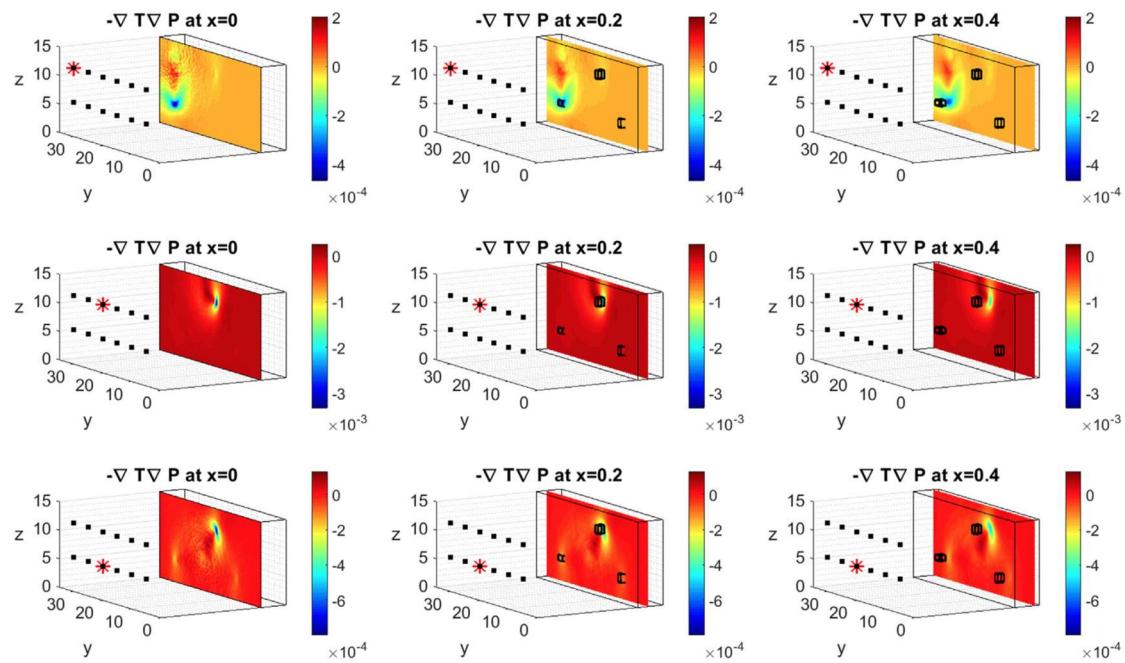


Figura 8: Aplicación de derivadas topológicas en termografías de placas con defectos. Fuente: [9]

En la Figura 8 se representan los valores de la derivada topológica en los planos $x = 0$, $x = 0.2$ y $x = 0.4$ obtenidos al procesar las termografías mostradas en la Figura 7, de forma que las imágenes de las filas 1, 2 y 3 de la Figura 8 muestran la derivada topológica correspondiente a las termografías de las filas 1, 2 y 3 de la Figura 7. Puede observarse que la derivada topológica en termografías ayuda mucho a resaltar la posición de alguno de los defectos. Observamos también que dependiendo de la posición de la lámpara considerada se identifica un objeto u otro (en particular, ninguna las tres posiciones consideradas es idónea para detectar la presencia del defecto situado a la derecha de la placa). Para mejorar el resultado obtenido, en el mismo trabajo [9], se combina varias derivadas para distintas lámparas considerando la función promedio J definida en (2.8) con los pesos β_j definidos en (2.18).

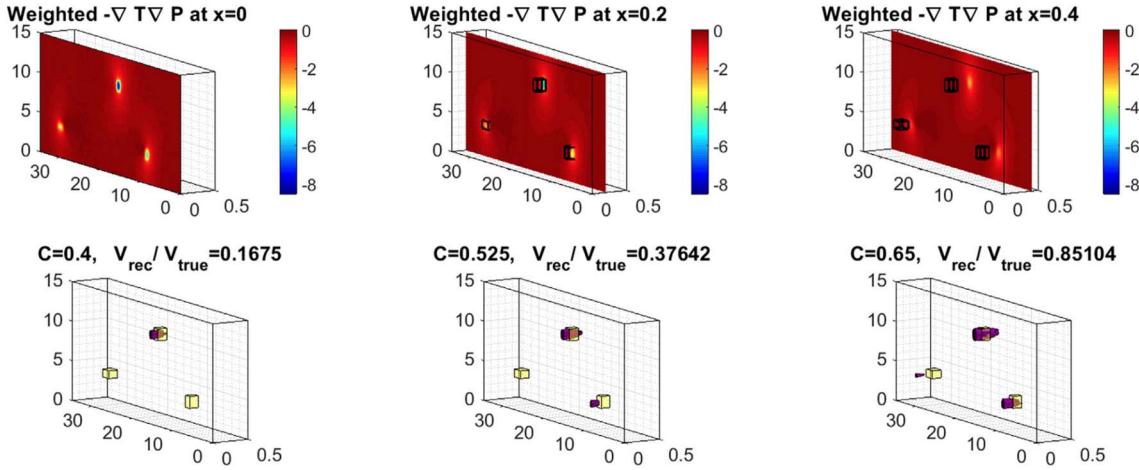


Figura 9: Valores en los planos $x = 0$, $x = 0.2$ y $x = 0.4$ de la derivada topológica de la función (2.8) con pesos definidos en (2.18). En la segunda fila se muestran los objetos reales (en color amarillo) y los reconstruidos (en magenta) al considerar la aproximación indicada en (2.11) para tres valores distintos de la constante C. Fuente: [9]

En la Figura 9, después de combinar varias derivadas topológicas para distintas lámparas, se pueden identificar perfectamente los defectos de la placa, pero no es posible saber la profundidad a la que se encuentran estos ya que se observa que independientemente del valor seleccionado de la constante C, los defectos parecen estar localizados en la cara frontal de la placa.

Esta imprecisión de las derivadas topológicas aplicadas a termografías, no se encuentra cuando aplicamos el método en otras técnicas de inspección no destructiva, como las radiográficas, ultrasónicas o electromagnéticas. En la Figura 10, extraída del artículo [10], muestra los resultados correspondientes a la aplicación de la derivada topológica en una técnica de inspección electromagnética. Y se puede observar que el resultado del método (imagen central) es capaz de dar información muy precisa, si la comparamos con los defectos originales (imagen de la derecha), sobre la profundidad de los defectos y la forma de estos.

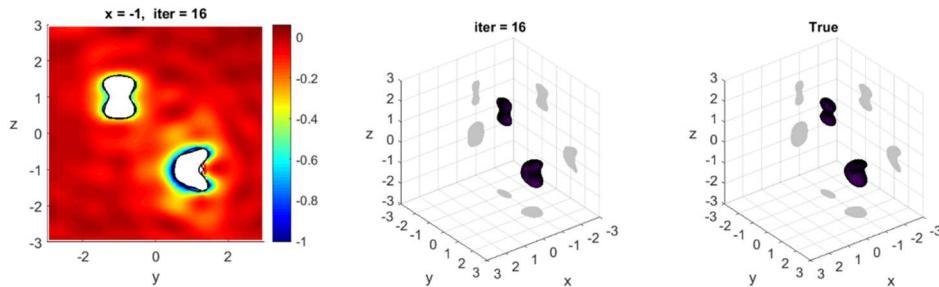


Figura 10: Aplicación de la derivada topológica en método de inspección electromagnética. [10]

Por lo tanto, este proyecto tratará de ver el comportamiento de las redes neuronales a la hora de predecir la profundidad de los defectos a dos conjuntos de datos: los valores resultantes tras contrarrestar la termografía de una placa sana y una dañada, y los valores que se obtienen tras la aplicación de la derivada topológica a dichas termografías.

2.5 Base de datos del experimento

Para las termografías, se ha seleccionado la base de datos “Base 7.1” realizada mediante el método de excitación estacionaria en el TFG de Á.Mateo [2]. Y para las derivadas topológicas se ha utilizado la base de datos “Harmonic DTt 30” realizadas mediante el método de excitación armónico en el TFM de S.Colubi [1].

En la Tabla 1 se muestra las características tanto de las muestras como del propio entorno:

Conductividad térmica del metal	$\kappa_{Al} = 200 \frac{W}{m \cdot K}$
Conductividad térmica del aire	$\kappa_{air} = 0,025 \frac{W}{m \cdot K}$
Emisividad del metal	$\epsilon = 0,08$
Absortancia del metal	$\alpha = 0,4$
Coeficiente de convección natural	$h = 15 \frac{W}{m^2 \cdot K}$
Constante de Stefan-Boltzmann	$\sigma = 5,67 \cdot 10^{-8} \frac{W}{m^2 \cdot K^4}$
Temperatura ambiente del aire	$T_{air} = 290 K$
Calor específico del aluminio	$c_{Al} = 900 \frac{J}{kg \cdot K}$
Calor específico del aire	$c_{air} = 1000 \frac{J}{kg \cdot K}$
Intensidad de radiación	$I = 6000 \frac{W}{m^2}$
Posición horizontal de la lámpara	$x_s = -0,15 m$

Tabla 1: Características de las muestras y del entorno

Para ampliar la información sobre las técnicas termográficas, la creación del experimento, los ensayos correspondientes o el proceso de parametrización de las variables, se puede consultar en el TFM de Sergio Colubi [1] y en el TFG de Ángel Mateo [2]. Este trabajo va a estar enfocado en el postproceso de los datos adquiridos en dichos experimentos.

Capítulo III

Entorno de ejecución

3.1 Software, hardware y librerías

Para realizar este trabajo, han sido necesarias las siguientes herramientas de software:

- **FreeFem++:** es un código de software libre que sirve para resolver problemas de ecuaciones en derivadas parciales mediante el uso de elementos finitos. Se ha utilizado para generar las bases de datos correspondientes a las termografías y para implementar la derivada topológica y generar las bases de datos correspondientes a las derivadas topológicas.
- **Paraview 5.8.0:** es una aplicación de código libre para interacción y visualización científica. Se ha utilizado para ilustrar los resultados de los experimentos.
- **Spyder:** es un entorno de desarrollo integrado (IDE) de código abierto para programación en el lenguaje Python. Se ha utilizado para crear las redes neuronales artificiales y las gráficas de los datos resultantes.
- **CeSVima:** consiste en un centro de supercomputación situado en Madrid que alberga el superordenador Magerit. En los trabajos [1] y [2], se ha utilizado tanto para la realización del experimento termográfico, en sus diferentes configuraciones, como para las derivadas topológicas. Y en este TFG se ha utilizado como nube para almacenar dichos datos.
- **Google Colaboratory:** se trata de un entorno de desarrollo integrado basado en Jupyter que sirve para escribir y ejecutar código de Python en la nube. Como recurso computacional, utiliza los servidores de Google, incluidas GPUs. En este trabajo se ha utilizado como herramienta fundamental para el entrenamiento de las redes neuronales.

Uno de los principales problemas del proyecto ha sido la carencia de potencia computacional cuando se utiliza CPUs para el entrenamiento de las redes, por ello, se ha decidido trabajar con GPUs en este proyecto. Las GPUs son procesadores compuestos por muchos núcleos pequeños que resultan muy prácticos a la hora de realizar multiplicaciones matriciales como el caso del entrenamiento de redes neuronales. Por lo

tanto, las GPUs tienen la capacidad de mejorar el tiempo de entrenamiento de las redes neuronales hasta en dos órdenes de magnitud. Es por ello que Google Colaboratory ha sido fundamental en este TFG, puesto que ha solventado esta carencia computacional.

El entorno de ejecución de Google Colaboratory no tiene incorporada una GPU por defecto y ésta necesita ser activada de manera manual en la pestaña “Entorno de ejecución”, “Cambiar entorno de ejecución” y elegir “GPU”. La GPU asignada es la Nvidia Tesla K80 con 4992 núcleos CUDA y con un rendimiento de 2,91 teraflops en operaciones de precisión doble [11]. Hay que tener en cuenta que la librería utilizada para el entrenamiento de nuestra red neuronal es Tensorflow, por lo que las únicas GPU compatibles son las de Nvidia con núcleos CUDA [12].

En cuanto a las librerías de Python utilizadas, se encuentran las siguientes:

- Numpy 1.16.5
- Pandas 0.25.1
- Matplotlib 3.1.1.
- Time
- Math
- Pickle
- Os
- Tensorflow 2.1.
- Keras 2.3.1.
- Sklearn

Capítulo IV

Redes Neuronales Artificiales.

Las redes neuronales artificiales, actualmente muy extendidas en todos los ámbitos de nuestra sociedad, tienen su origen en 1958 en un Laboratorio Aeronáutico de la Universidad de Cornell por Frank Rosenblatt, con la invención de su algoritmo *el perceptrón* [13], que simula la siguiente función:

$$f(x) = \begin{cases} 1 & \text{si } \sum_{i=1}^N x_i w_i + b_i > 0 \\ 0 & \text{resto de casos} \end{cases} \quad (4.1)$$

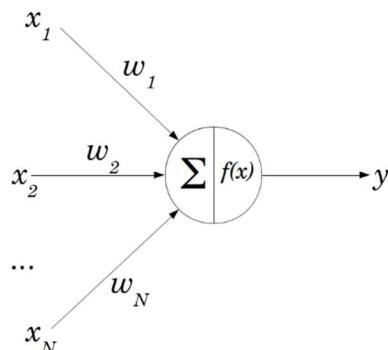


Figura 11: Concepto simple del algoritmo del perceptrón.

En la Figura 11 y en la definición (4.1), $x = (x_1, \dots, x_N)$ representa el conjunto de valores de entrada a la neurona, w_i corresponde a los pesos relacionados de cada input, b_i (bias) es el sesgo de cada valor de entrada y N el número de inputs de la neurona. Se trata de un clasificado binario que decide el grupo al que pertenece cada uno de los valores de entrada.

Este algoritmo, basado en la teoría del aprendizaje hebbiano de Donald O. Hebb [14], simula el comportamiento de las conexiones entre las neuronas cerebrales.

El perceptrón de una sola capa tiene el inconveniente de ser únicamente capaz de aprender patrones lineales. Sumado a la falta de capacidad computacional de la época, hizo que la teoría empezara a perder fuerza. Posteriormente, en 1985, fue con la

implementación del algoritmo de *backpropagation* [15] a las redes multicapa, lo que cambió el paradigma de las redes neuronales, haciendo éstas mucho más eficientes a la hora de predecir patrones. En las últimas décadas, el gran avance tecnológico, sobre todo en el campo de los semiconductores, ha convertido a las redes neuronales en un recurso fundamental en diferentes ámbitos de la sociedad.

En esta sección se va a hacer una breve explicación de los conceptos fundamentales de las redes neuronales profundas y sus parámetros, por lo que, si el lector está interesado en ampliar su conocimiento sobre la materia, en el libro *Deep Learning* [16] (Goodfellow et al.; 2016; MIT) puede encontrar información mucho más detallada.

4.1 Perceptrones multicapa, capas ocultas

Uno de los problemas principales del perceptrón, como explicamos anteriormente, es su linealidad, lo cual limita mucho las posibilidades de usarlo como herramienta matemática. Las capas ocultas han sido un método muy efectivo para solventar este problema, ya que esta sucesión de capas permite que la red simule comportamientos no lineales. Todas las neuronas de las capas están “completamente conectadas” (*fully-connected*) con las neuronas de sus capas vecinas como se muestra en la Figura 12.

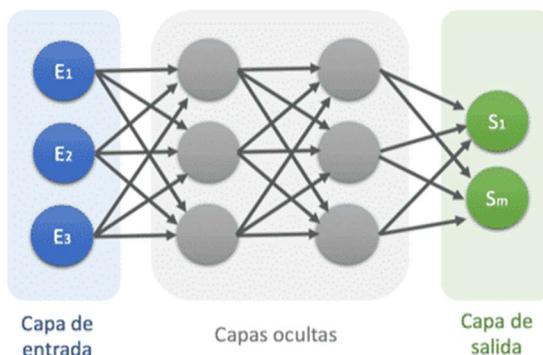


Figura 12: Concepto simple del perceptrón multicapa.

Como se puede ver en la Figura 13, este perceptrón multicapa se asemeja mucho a cómo las neuronas biológicas reciben la información mediante sus dendritas y la envían a través del axón, el cual representa las capas ocultas, hasta llegar a las terminales del axón. Estas terminales simulan los valores de salida del perceptrón multicapa.

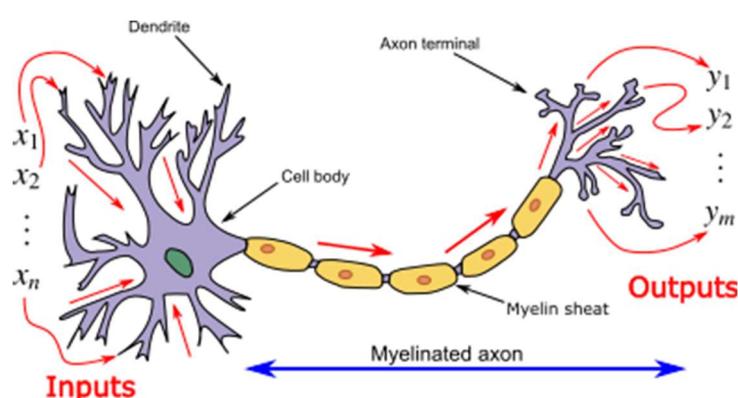


Figura 13: Símil entre neurona biológica y perceptrón multicapa.

A continuación, se describe matemáticamente la operativa necesaria en el caso de una red con una única capa oculta.

$$\begin{aligned} z_j &= \sum_{j=1}^m \sum_{i=1}^n w_{j,i} x_i + b_j, \\ y_j &= \sum_{j=1}^m \sum_{i=1}^n w_{j,i} z_i + b_j, \end{aligned} \quad (4.2)$$

donde x_i representa los valores de entrada de la red, z_j el valor de las neuronas de la capa oculta, $w_{j,i}$ y b_j representan los “pesos” y el “sesgo” que se atribuye a cada neurona de la capa oculta, por ejemplo, $z_1 = (w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3}x_3 + \dots + w_{1,n}x_n) + b_1$. Por último, y_j corresponde a los valores resultado, situados en la capa final, que estamos buscando.

4.2 Funciones de activación

Para aprovechar el potencial de la arquitectura multicapa, necesitamos de un componente fundamental, una *función de activación* ($f(\cdot)$), la cual aplicaremos a cada neurona. Estas funciones pueden decidir si una neurona debe activarse o no, y, en caso de activarse, el valor ponderado que ésta debe tener para nuestra red neuronal. Las funciones de activación se distinguen en lineales y no lineales.

ReLU (Rectified Linear Unit)

La activación *ReLU* es la más popular a la hora de construir redes neuronales, tanto por su simplicidad como por su buen desempeño en multitud de tareas [17]. Dado un elemento x , la función se define como el máximo entre éste y 0.

$$ReLU(x) = \max(x, 0). \quad (4.3)$$

ELU (Exponencial Linear Unit)

La diferencia entre la activación *ReLU* y *ELU* es que, en esta última, obtenemos valores diferentes a cero cuando la entrada es negativa. La *ELU* es una función que tiende a minimizar el error de forma más rápida y a producir resultados más precisos [18]. En cambio, tiene el problema de ser computacionalmente más pesada que la *ReLU*.

$$ELU(x) = \begin{cases} x & \text{si } x > 0 \\ \alpha(e^x - 1) & \text{si } x < 0. \end{cases} \quad (4.4)$$

Sigmoide / Logística

La función sigmoide transforma los datos de entrada en un intervalo de valores entre 0 y 1.

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}. \quad (4.5)$$

Se trata de una de las primeras activaciones utilizadas en redes neuronales, por ser una función diferenciable y por su similitud con el “potencial de acción” de las neuronas biológicas [19]. Pero, debido a que tiene una derivada cercana a cero en los extremos provocando problemas de desvanecimiento de gradiente, con el tiempo se ha visto reemplazada por la función *ReLU*. Ésta, además de evitar el problema anterior, es más fácil de entrenar por su simplicidad. Aun así, la función sigmoide es muy útil a la hora de construir redes neuronales, colocándose en la última capa para servir como función resultado del modelo.

TanH / Tangente-hiperbólica

Al igual que la función sigmoide, la función tangente-hiperbólica transforma los datos de entrada a un resultado acotado. En este caso, el intervalo es (-1,1).

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}. \quad (4.6)$$

La tangente-hiperbólica nos permite trabajar con valores de entrada negativos, aunque sigue teniendo el problema del desvanecimiento de gradiente cuando nos alejamos del cero, puesto que lejos del eje central la derivada tiende a cero.

Existen multitud de funciones de activación, cada una de ellas con sus respectivas ventajas e inconvenientes. En la Figura 14 se muestran algunas de las funciones de activación anteriormente mencionadas y otras que también están disponibles en Tensorflow2.1 (véase [20] para mayor detalle).

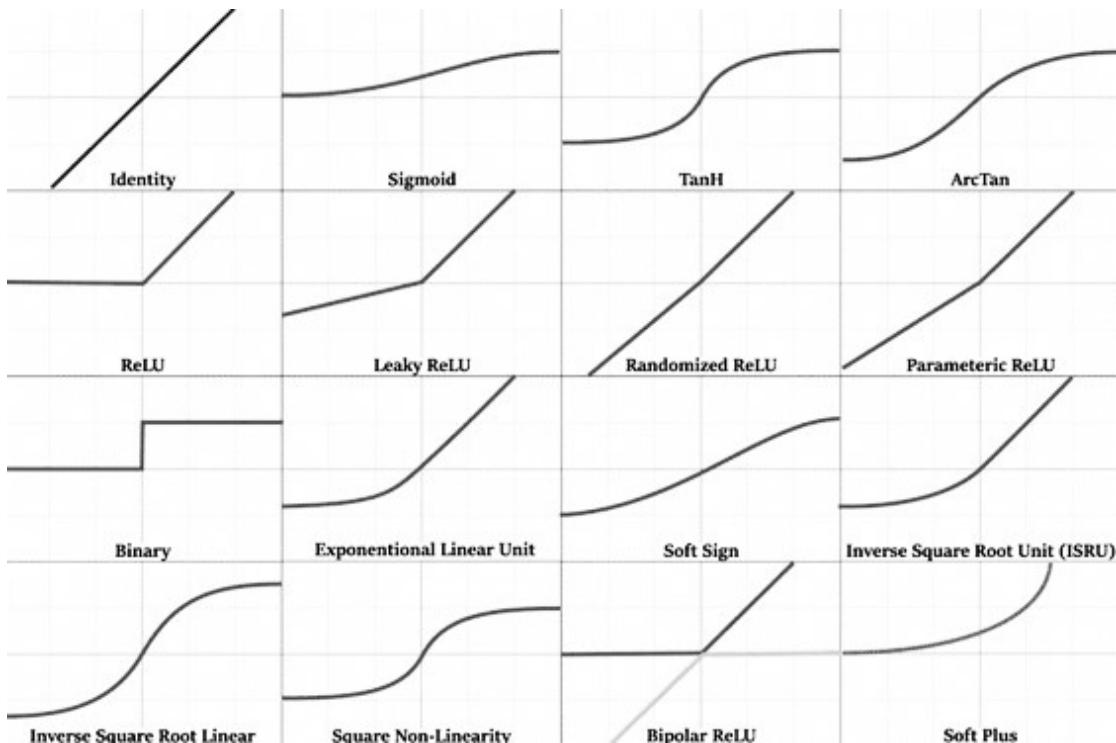


Figura 14: Algunas de las funciones de activación más usadas en el campo del Deep Learning.

El valor obtenido después de pasar por cualquier función de activación f , normalmente recibe el nombre de “activación” (a), por lo tanto, la formulación de las interconexiones neuronales de la red queda de la siguiente manera:

$$z_j^l = \sum_{j=1}^m \sum_{i=1}^n w_{j,i}^l a_i^{l-1} + b_j^l$$

$$a_j^l = f(z_j^l) \quad (4.7)$$

siendo l el superíndice que indica la capa correspondiente a cada parámetro.

4.3 Función de coste

La función de coste, también conocida como función de error, es la herramienta que se usará para determinar la distancia entre el valor real y el predicho. La función de coste más usada en problemas de regresión es el error cuadrático medio:

$$c^{(i)}(w, b) = \frac{1}{2} (\tilde{y}^{(i)} - y^{(i)})^2 \quad (4.8)$$

donde $\tilde{y}^{(i)}$ es el valor predicho e $y^{(i)}$ el valor real.

Para medir el error cometido para todo el conjunto de datos, debemos realizar el promedio de los errores del entrenamiento:

$$C(w, b) = \frac{1}{n} \sum_{i=1}^n c^{(i)}(w, b) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} ((w^T x^{(i)} + b) - y^{(i)})^2. \quad (4.9)$$

En el campo del Deep Learning, cuando se trata de métricas de regresión, también se trabaja frecuentemente con otras funciones de coste como el error absoluto medio (MAE), el error logarítmico cuadrático medio (MSLE) o el error porcentual absoluto medio (MAPE). En la página web [21], se pueden encontrar todas las funciones de coste que están disponibles en la librería de Tensorflow.

El objetivo final a la hora de entrenar una red neuronal es minimizar el error de nuestra predicción respecto a la solución real. Al tener muchas capas con muchas neuronas, encontrar una solución analítica al problema es prácticamente inviable. Esto conlleva que, en las redes neuronales profundas, se utilicen otros métodos para acercarnos a la solución, que se muestran a continuación.

4.4 Optimización, descenso del gradiente

El optimizador más usado en el campo del aprendizaje profundo es, sin lugar a duda, el descenso del gradiente. Éste consiste en un algoritmo de optimización iterativo de primer orden [22], que busca encontrar un mínimo local en una función diferenciable. La idea básica del algoritmo consiste en dar pasos en dirección opuesta al gradiente de la función en un punto, de manera iterativa, hasta encontrar un mínimo. Por lo que el punto siguiente a $x^{(i)}$, se define como:

$$x^{(i+1)} = x^{(i)} - \alpha \cdot \nabla f(x^{(i)}) \quad (4.10)$$

siendo $\alpha > 0$ el tamaño del paso, en la dirección opuesta al gradiente. En la Figura 15 se muestra gráficamente una sucesión de puntos obtenida al aplicar este método.

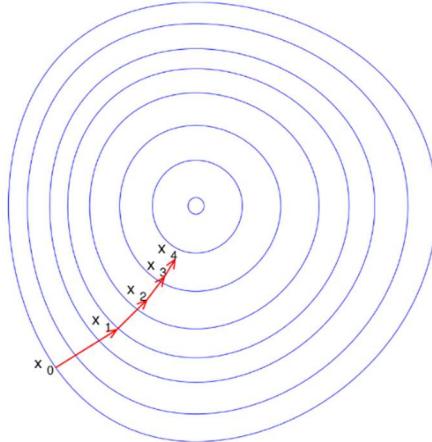


Figura 15: Imagen conceptual del descenso del gradiente. Fuente: pngegg.com

En el campo del Deep Learning, la función objetivo consiste en una suma de todos los errores individuales de cada neurona ponderados por sus respectivos pesos y como estos pesos se inicializan de manera aleatoria, el método se le denomina descenso de gradiente estocástico (SGD) [23]. La desventaja de éste se encuentra en los *outliers* de la muestra, que afectan de manera muy notable al método. Además, a la hora de resolver problemas, dependemos mucho del tamaño del paso, pudiendo estancarnos en un mínimo local no deseado.

Descenso de Gradiente con Momento (Inercia)

La diferencia de este método con el anterior [24], es que la velocidad del descenso se va acumulando a medida que avanzan los pasos, es decir, el momento le proporciona un descenso acelerado del gradiente que se define del siguiente modo:

$$\begin{aligned} p^{(i)} &= \nabla f(x^{(i)}) + \eta p^{(i-1)} \\ x^{(i+1)} &= x^{(i)} - \alpha p^{(i)} \end{aligned} \quad (4.11)$$

siendo $p^{(i)}$ la dirección del descenso en cada iteración. El parámetro $\eta > 0$ actúa como un “coeficiente de rozamiento”, para que el modelo tenga una velocidad límite y se pueda controlar la suavidad del descenso.

Aunque este método mejora el tiempo de convergencia del SGD y arregla el problema de los mínimos locales no deseados, también tiene el inconveniente de sobrepasar los resultados que se buscan (*overshooting*).

Descenso acelerado de Nesterov (NAG)

El descenso acelerado de Nesterov [25] consiste en un método de dos pasos que trata de mejorar el método anterior. En el primer paso, éste extrapola linealmente la trayectoria actual; y en el segundo paso, calculando el gradiente en el punto predicho, realiza una corrección de la trayectoria. De esta manera, se consigue una aproximación de segundo orden de la trayectoria con el coste computacional del método anterior, incrementando así la velocidad de convergencia. Sus ecuaciones son las siguientes:

$$\begin{aligned}\tilde{x}^{(i)} &= x^{(i)} - \alpha p^{(i-1)} \\ p^{(i)} &= \nabla f(\tilde{x}^{(i)}) + \eta p^{(i-1)} \\ x^{(i+1)} &= x^{(i)} - \alpha p^{(i)}\end{aligned}\tag{4.12}$$

Debido a la gran cantidad de datos de entrenamiento en este trabajo, se ha tenido que optar por el último método. Además, en las pruebas iniciales, se ha visto que también es el método más certero.

4.5 Learning Rate (Ratio de aprendizaje)

Como se ha mencionado anteriormente, el tamaño de paso, α , es uno de los parámetros fundamentales en el proceso del descenso de gradiente. En el campo del aprendizaje profundo se le conoce como ratio de aprendizaje, que define la velocidad a la que la red neuronal va a aprender.

Uno de los problemas que se encuentra a la hora de configurar este parámetro, es que, si se quiere ser preciso, la red necesita de una ratio de aprendizaje pequeña, haciendo que el aprendizaje sea muy lento y tenga la posibilidad de acabar en un mínimo local no deseado (*underfitting*). En cambio, si se quiere tener un aprendizaje rápido, se necesita de una ratio de aprendizaje grande, lo que puede producir una perdida sustancial de precisión a causa del *overfitting*.

Por ello, en este TFG, a la hora de buscar cómo definir correctamente este parámetro, se ha optado por aplicar una ratio que descienda a lo largo del proceso, de manera que se consigan las ventajas de ambos modelos. Para ello, se ha decidido aplicar una *ratio de aprendizaje exponencial decreciente*, que viene dada por la siguiente fórmula:

$$Lr = Lr_{inicial} * e^{-k*epoch}\tag{4.13}$$

siendo:

- Lr : ratio de aprendizaje
- $Lr_{inicial}$: ratio de aprendizaje inicial
- k : parámetro de velocidad de descenso
- $epoch$: iteración

Hay que tener en cuenta, que cuando se utilizan este tipo de ratios de aprendizaje, pueden surgir dos tipos de problemas:

1. Si k es muy alta, la ratio de aprendizaje descenderá muy rápido y es posible que ocurra *underfitting*, es decir, que el descenso acabe antes de llegar a la zona del mínimo deseado.
2. Si k es muy baja, es posible que estemos mucho tiempo trabajando a ratios de aprendizajes demasiado altos y no consigamos la precisión que se busca.

Por lo tanto, este tipo de *Learning Rate* tiene una dificultad añadida, donde el usuario debe probar y estimar de manera oportuna, cuál es la ratio de aprendizaje que más le conviene en cada situación, y a qué velocidad quiere que ésta descienda en el problema.

En este TFG se ha probado con muchas combinaciones de diferentes $Lr_{inicial}$ y k para cada problema, hasta encontrar el punto óptimo donde los modelos convergen al mínimo global de manera precisa y en el mínimo tiempo posible.

4.6 Backpropagation

Anteriormente, uno de los problemas fundamentales en el campo del aprendizaje profundo consistía en la dificultad de minimizar la función cuando la red neuronal se hacía muy grande. Esto cambió completamente desde la publicación del artículo del descenso de gradiente mediante *backpropagation* [15]. En este apartado, se expone el método matemático utilizado en dicho método.

Para calcular el error de cada neurona en una capa se utiliza la siguiente formulación:

$$\begin{aligned}\delta_j^L &= \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L} \\ \frac{\partial C}{\partial b_j^L} &= \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L} \cdot \frac{\partial z_j^L}{\partial b_j^L} = \delta_j^L \cdot 1 \\ \frac{\partial C}{\partial w_j^L} &= \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L} \cdot \frac{\partial z_j^L}{\partial w_j^L} = \delta_j^L \cdot a_j^{L-1},\end{aligned}\tag{4.14}$$

donde C corresponde al coste final de la red, z_j^L es el valor asociado de cada neurona, a_j^L corresponde a la activación, w_j^L son los pesos correspondientes a cada neurona, b_j^L es el sesgo asociado y δ_j^L es el error que se atribuye a cada neurona en el proceso de *backpropagation*. El superíndice L corresponde a la capa en la que se encuentra cada término.

Como se puede apreciar en la formulación anterior, utilizando la regla de la cadena, se puede conocer cómo varía el coste respecto a los diferentes parámetros de cada neurona, y de esta manera, podemos variar los pesos para minimizar los errores δ_j^L correspondientes. Cuando necesitamos realizar esto para muchas capas, en el artículo [15], propone lo siguiente:

$$\delta_j^{L-1} = \frac{\partial C}{\partial z^{L-1}}.\tag{4.15}$$

La anterior ecuación facilita todo el proceso, porque se puede calcular el error de una neurona como la suma ponderada de todos los errores de las neuronas de la capa

posterior, ponderándolas con los pesos y funciones de activación. De esta manera, si se realiza este cálculo con todas las neuronas de esa capa, se puede saber el error de todas las neuronas de la capa anterior. Repitiendo este proceso hasta la capa inicial, podemos cuantificar cómo varía el error de todas las neuronas de la red variando sus pesos a consecuencia (*backpropagation*).

4.7 Batch y epoch (lote y época)

Tras calcular el error asociado a cada neurona de nuestra red, se podría realizar una actualización de los pesos de éstas, aunque en la práctica esto no es lo que se suele hacer. Un problema típico a la hora de entrenar redes neuronales es la variedad de los datos de entrada, esto puede ocasionar que, si se actualiza los pesos de las neuronas con cada muestra del conjunto de datos, el entrenamiento resulte muy errático. Una estrategia para evitar este problema es estimar, sin cambiar los pesos, el error de la red con varias muestras de datos. De esta manera, el error promedio que se consigue hace que el entrenamiento sea menos errático y disminuye sustancialmente el tiempo necesario para entrenar el conjunto de datos. A este conjunto de muestras se le denomina lote (batch).

Entrenar a la red con todos los datos una sola vez puede no ser suficiente. El número de veces que la red entrena a todo el conjunto de datos se denomina época (epoch).

Hay que tener en cuenta que, cuando se entrena con un tamaño de lote muy grande, el error que se consigue para calibrar la red es demasiado general y se pierden características importantes de los datos. Además, esto dispara el número de épocas necesarias para llegar al mínimo. Por ello, se suele utilizar un tamaño de lote comedido, evitando así el entrenamiento errático (sin lote) a la par que un entrenamiento generalista.

4.8 Batch Normalization

Una de las herramientas más populares utilizadas en los últimos proyectos de aprendizaje profundo es la normalización por lotes de entrenamiento [26]. En términos generales, esta técnica tiene como objetivo suavizar las distribuciones de activación, controlando la media y la desviación estándar de los outputs de cada capa en cada lote. Para una activación y_j de la capa y se define la normalización por lotes (BN de sus siglas en inglés Batch Norm) mediante la expresión:

$$BN(y_j^{(b)}) = \gamma \cdot \left(\frac{y_j^{(b)} - \mu(y_j)}{\sigma(y_j)} \right) + \beta, \quad (4.16)$$

donde $y_j^{(b)}$ corresponde al valor del output de y_j del número de lote b . $\gamma > 0$ y β son los parámetros encargados de controlar la media y la varianza del valor resultante.

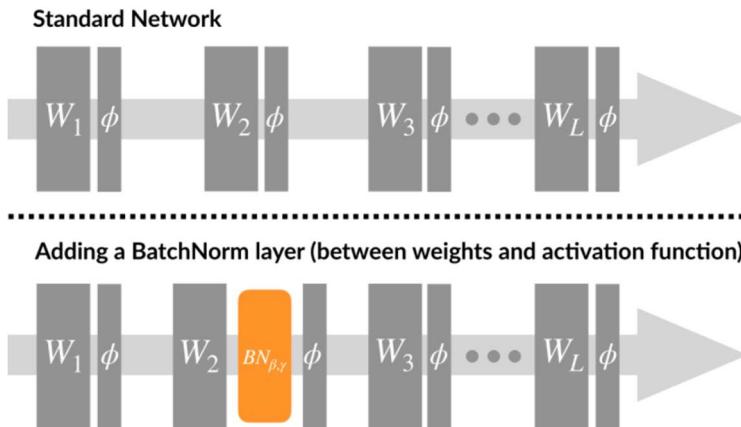


Figura 16: Estructura clásica de una arquitectura de red neuronal con Batch Normalization.

En la Figura 16, se puede apreciar que la normalización por lotes se aplica después de la primera fórmula de la ecuación (4.2) y antes de la ecuación (4.7). Es decir, después de aplicar los pesos y antes de calcular la activación.

En el artículo original, [26], los autores indican que la mejora de esta técnica viene dada por la “reducción del cambio interno de covariables”, puesto que, según su hipótesis, la inicialización de parámetros y los cambios en la distribución de las entradas de cada capa afectan a la tasa de aprendizaje de la red neuronal. Sin embargo, algunos académicos argumentan que es debido a la suavización de la *función objetivo*, mejorando así el rendimiento. También, hay quienes justifican que es debido a la *explosión de gradiente* que genera el método, o que es debido a que la normalización por lotes logra un *desacoplamiento de la dirección de entrenamiento*, entre otros [27].

La normalización por lotes de entrenamiento es un método que funciona muy bien para el entrenamiento de redes neuronales, aunque no está claro el porqué de su efectividad, como ocurre con gran parte del campo del aprendizaje profundo.

4.9 Tipos de redes neuronales profundas.

En el campo del aprendizaje profundo existen numerosas arquitecturas de redes neuronales, desde la simple red fully-connected (FC), hasta otras mucho más complejas como las que se mencionan a continuación:

- **Red Generativa Adversaria (GAN):** consiste en una arquitectura donde una red genera datos, mientras otra discrimina y evalúa los datos creados.
- **Red Neuronal Recurrente (RNN):** es una red que tiene en cuenta la secuencia de los datos de entrada, por lo que es muy utilizada para el procesamiento de lenguaje natural (NLP)
- **Red Neuronal Convolucional (CNN):** utilizada comúnmente con imágenes, videos e incluso reconocimiento de voces. Consiste en la aplicación de filtros a los datos iniciales, que se encargan de extraer información adicional de los datos de entrada. Este tipo de red será el que utilizaremos en este TFG, por lo que hemos dedicado el próximo capítulo 5 a una descripción más detallada de las mismas.

- Híbridas: cuando el problema a resolver es muy complejo, se utilizan redes formadas por mezclas de otras, como ocurre en el caso de la conducción autónoma.

En la siguiente página, la Figura 17 muestra cómo se estructuran algunas de las principales arquitecturas que se utilizan en el campo del aprendizaje profundo.

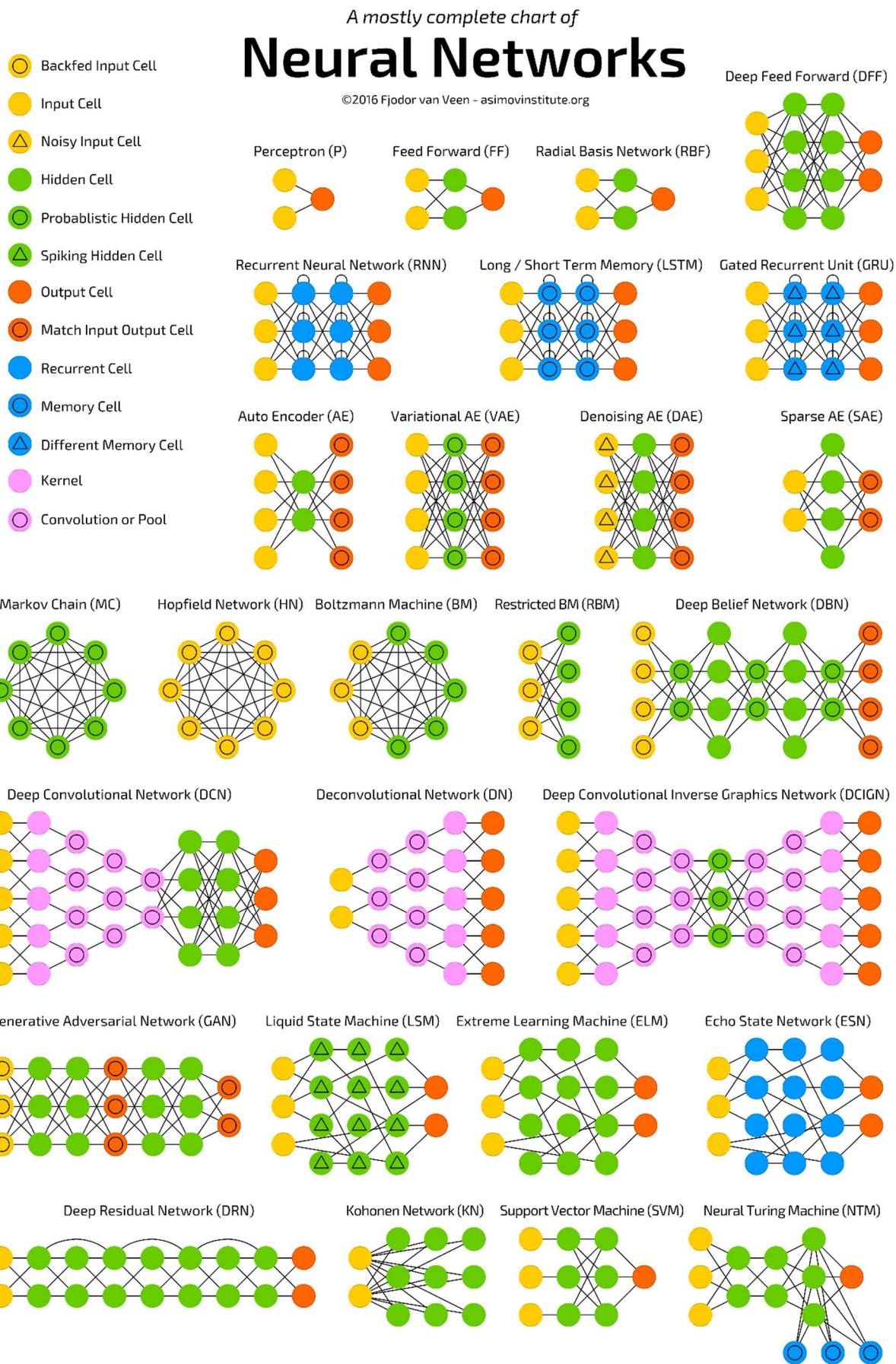


Figura 17: Arquitecturas de redes neuronales más usadas. Fuente: [28]

Capítulo V

Redes Neuronales Convolucionales (CNNs)

Como ya se ha mencionado, el objetivo del trabajo es encontrar una manera de predecir mejor la profundidad de los defectos presentes en el interior de una placa a partir de mediciones termográficas capturadas en una de sus superficies. Para ello, utilizaremos redes neuronales convolucionales. Este tipo de redes son muy eficaces a la hora de extraer información mediante patrones en los datos y en el presente trabajo se intentará sacar provecho de esta característica para mejorar la precisión de nuestras predicciones.

Las CNNs son redes neuronales jerarquizadas cuyas capas convolucionales se alternan con las capas de muestreo, recordando así a las células simples y complejas de la corteza visual primaria [29]. Por ello, el entrenamiento de las redes convolucionales difiere sustancialmente al de las redes convencionales. Algunos de los primeros trabajos que utilizan las CNNs son: Schmidhuber et al., 1996 [30]; Olshausen y Field, 1997 [31]; Hoyer y Hyvarinen, 2000 [32].

Una de las características que tienen las CNNs es la implementación de filtros localizados, que en nuestro caso servirán para detectar de manera más sensible la información de cada termografía y derivada topológica.

5.1 Capa convolucional

Según el artículo [33], dada una matriz $A_{m \times n}$ y una matriz $C_{k \times l}$ con $k < m$ y $l < n$ se define la convolución de las matrices A y C como una nueva matriz $D = A * C$ definida a partir de la expresión:

$$d_{ij} = \sum_{r=1}^k \sum_{s=1}^l a_{(i-1)+r, (j-1)+s} \cdot c_{r,s}, \quad (5.1)$$

donde d_{ij} está definida para $i = 1, \dots, m - k + 1$ y $j = 1, \dots, n - l + 1$. La matriz C se le denomina *kernel* o filtro. Para ilustrar este proceso de convolución se ha considerado el ejemplo de la Figura 18:

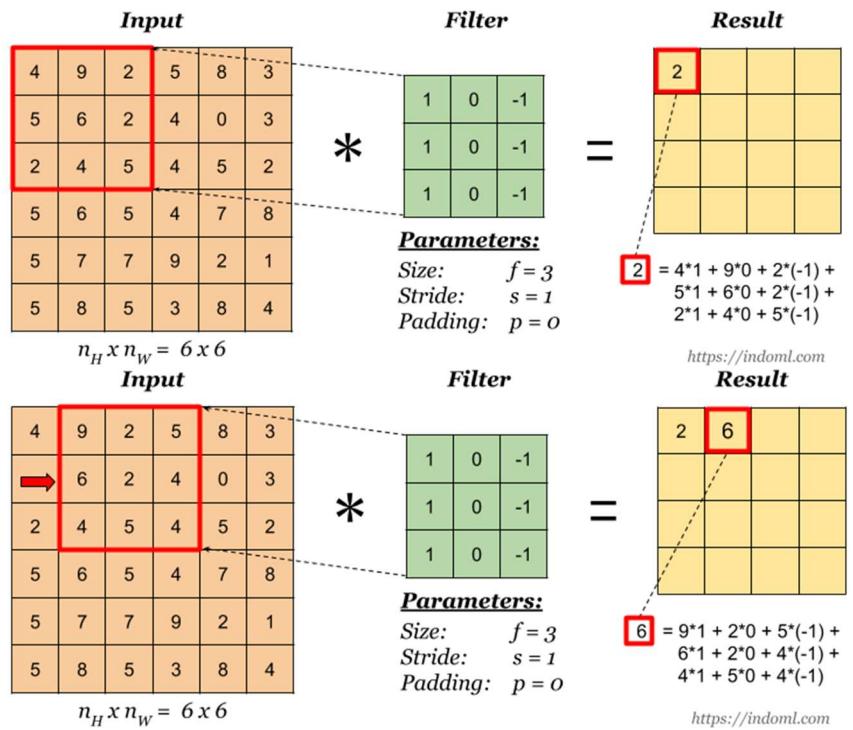


Figura 18: Operación de un filtro a la matriz de entrada. Fuente: indoml

Como se puede ver en la Figura 18, la convolución consiste en la multiplicación directa de la matriz inicial por una matriz filtro en cada zona, creando así una matriz final más pequeña que resalte ciertas características de la matriz inicial.

Nótese que según la igualdad (5.1) y la Figura 18, el tamaño de la matriz de salida es menor que el de entrada. Cuando el salto (*stride*) de la matriz filtro es la unidad, la correlación de las dimensiones viene dada por:

$$(A_x - C_x + 1) \times (A_y - C_y + 1), \quad (5.2)$$

donde x es la dimensión horizontal e y la dimensión vertical. Es decir, con la notación utilizada en la definición (5.1) tenemos $A_x = m$, $A_y = n$, $C_x = k$ y $C_y = l$

Otra de las opciones a la hora de aplicar un filtro es dar dos o más saltos, en vez de uno. De esta manera, la matriz resultante ocupa menos espacio, pero existe la posibilidad de perder información en el proceso. En la Figura 19, se puede observar cómo este proceso de convolución se ha realizado con un salto (*stride*) de la matriz filtro de dos unidades.

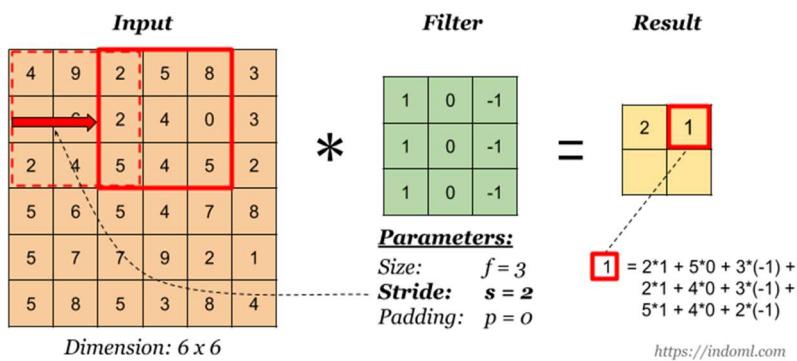


Figura 19: Aplicación de un filtro con doble paso. Fuente: indoml

La convolución al replicarse con diferentes filtros consigue tantas matrices finales como número de filtros se apliquen. Estos filtros pueden ser matrices fijas con funcionalidades determinadas como ilustra la Figura 20, pero en nuestro caso, el proceso para seleccionar los filtros óptimos pasa por la utilización de la red neuronal con el fin de encontrar los que minimicen el error de la red.

Enfoque	Desenfoque	Realce de bordes	Repujado
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$
Detección de bordes	Filtro de tipo Sobel	Filtro de tipo Sharpen	
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$	
Filtro Norte	Filtro Este	Filtro de tipo Gauss	
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 1 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 1 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$	

Figura 20: Tipos de filtros más usados. Fuente: [33]

5.2 Padding (rellenado)

Uno de los problemas clásicos que se encuentra a la hora de aplicar filtros es la pérdida de información en el perímetro de la matriz, que empeora con la sucesiva acumulación de filtros que se aplica. Una solución a este problema es el *padding*, que consiste en agregar valores adicionales de relleno alrededor del límite de nuestra matriz. En el ejemplo siguiente, mostrado en la Figura 21, se muestra cómo al aplicar el *padding* a la matriz, ésta aumenta su tamaño de 6x6 a 8x8, de manera que la matriz final conserva el tamaño de la matriz inicial. Por lo tanto, la correlación de las dimensiones tras el *padding* y la convolución viene dada por:

$$(A_x - C_x + p_x + 1) \times (A_y - C_y + p_y + 1), \quad (5.3)$$

donde p_x y p_y corresponden al número de filas y columnas adicionales que añade el *padding*.

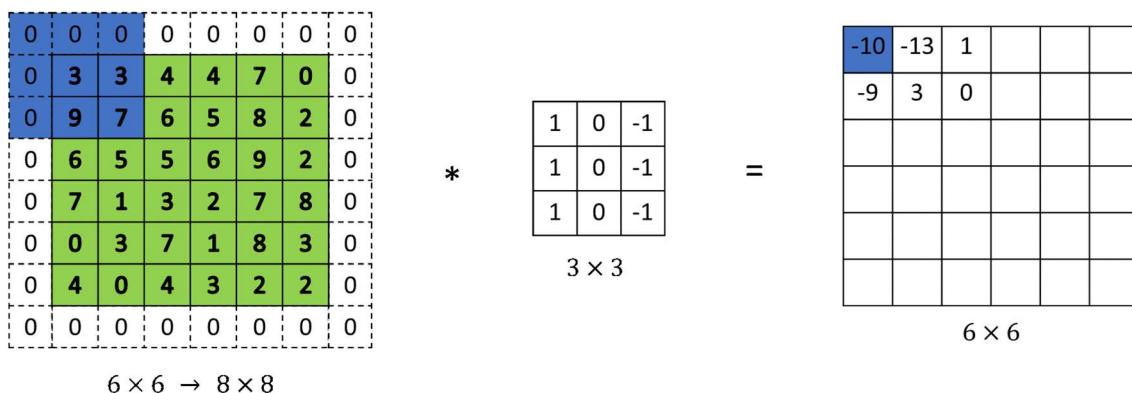


Figura 21: Aplicación de *padding* a una matriz y posterior convolución con un filtro 3x3. Fuente: <http://datahacker.rs>

5.3 Convolución de matrices de diferentes dimensiones.

Los datos de entrada en la red neuronal no tienen por qué ser bidimensionales. Por ejemplo, en el caso de las fotos a color, nos encontramos una matriz de 3 dimensiones, siendo esta última dimensión los tres colores RGB. En este caso, las dimensiones del filtro deben ser $C_x \times C_y \times 3$, como ilustra la Figura 22. En otros casos, tenemos un vector unidimensional, como es el caso de los datos de series temporales. En conclusión, la dimensión de los filtros se debe adaptar siempre a las dimensiones de los datos de entrada.

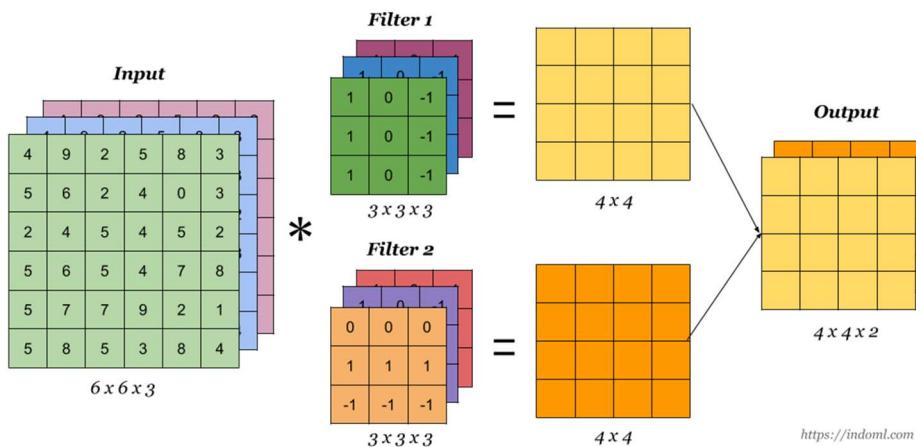


Figura 22: Convolución de matriz 3D con dos filtros 3D. Fuente: <https://indoml.com>

5.4 Capa Pooling

El concepto de agrupación (*pooling*), publicado en 2010 en el artículo [34], es uno de los pilares fundamentales en los sistemas actuales de redes neuronales a la hora de crear sistemas eficientes. Uno de los problemas fundamentales de crear muchos filtros, es el aumento de la cantidad de datos con los que se va a entrenar a la red, y, por ello, tras la convolución con filtros, generalmente se aplica la operación *pooling*. Este proceso consiste en la aplicación de otro filtro para disminuir el tamaño de la matriz. Los más usados son dos: *max-pooling* y *average pooling*. En el *max-pooling* se elige el valor más alto de la sección que aplica el filtro para el valor de la matriz final, mientras que el *average pooling* realiza la media de los valores seleccionados.

Para ilustrar el funcionamiento de este tipo de procesos, se muestra en la Figura 23 un ejemplo donde puede verse el resultado de aplicar ambos filtros a una matriz de tamaño 4x4. El método del *max-pooling* recoge los 4 valores más altos de cada sección 2x2 y el *average pooling* una media de todos los valores de cada sección. De esta manera, cada dimensión de la matriz final termina siendo la mitad de la original.

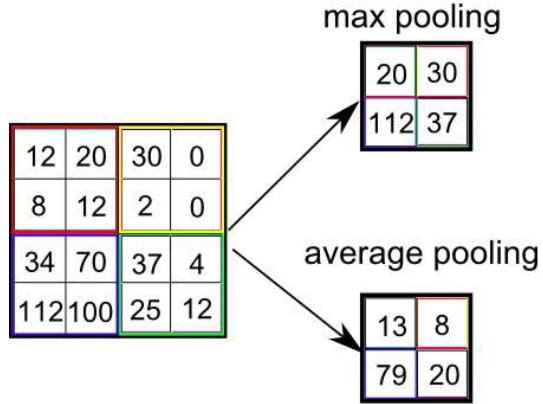


Figura 23: Aplicación de *max-pooling* y *average pooling* a una matriz de tamaño 4x4.

5.5 Ejemplos de redes CNN complejas

Una de las arquitecturas complejas de redes neuronales convolucionales más utilizadas es la llamada *LeNet* [35] (*LeNet-5*). Ésta consta de 2 fases claramente diferenciadas, como se puede apreciar en la Figura 24: una doble convolución con sus respectivos *poolings* y posteriormente, varias capas convergentes de neuronas *fully-connected*. La finalidad de esta red convolucional es sectorizar la información de la imagen inicial divergiéndola en múltiples imágenes para después entrenar la red neuronal con estos datos más específicos.

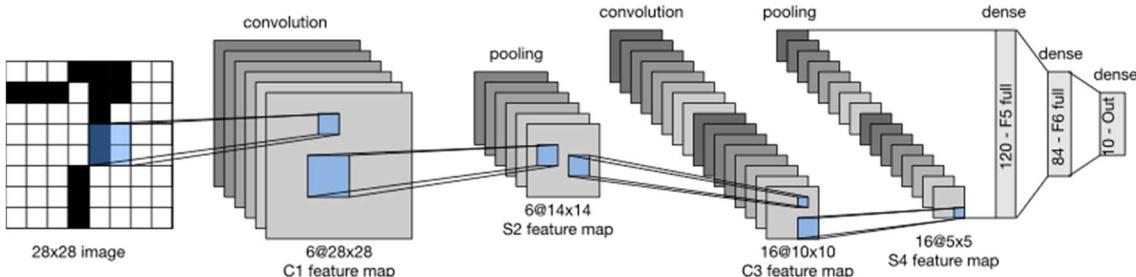


Figura 24: Aplicación de una red *LeNet-5* a una imagen 28x28. Fuente: [36]

Este trabajo se inspirará en la red *LeNet-5* para construir su propia red neuronal convolucional para lidiar con el problema planteado anteriormente.

Otras redes CNN complejas pueden ser:

- AlexNet [37]
- Red de bloques (VGG) [38]
- Red de redes (NiN) [39]
- Red con concatenaciones paralelas (GoogLeNet) [40]

En resumen, en este TFG se utilizarán las redes convolucionales, junto a las diferentes capas asociadas a éstas, para resaltar la información en forma de patrones de las muestras termográficas y derivadas topológicas, para mejorar así el entrenamiento de nuestras redes.

Capítulo VI

Configuración de las Redes Neuronales

En este capítulo comentaremos primero cómo son las bases de datos que se van a procesar y a continuación se describirán los aspectos más importantes que definen a las redes consideradas.

6.1 Características de las bases de datos consideradas.

Las bases de datos seleccionadas para realizar las pruebas son: “Base7.1” para las termografías y “FinalDatabase30” para las derivadas topológicas.

La base de datos “Base7.1”, consiste en 7000 termografías distintas con 15 tipos de ruido diferentes en cada una, dando un total de 105 mil muestras. Cada una de estas muestras tiene 1001 valores de temperaturas, es decir, más de 105 millones de valores termográficos. La base de datos “Harmonic_DTt_30”, subdividida en 5 bases de datos: Harmonic_DTt_301, 302, 303, 304, 305 consiste en 24900 derivadas topológicas, con una dimensión de sus valores de 10x10, es decir un total de 2,49 millones de valores para el entrenamiento.

Antes de aplicar las redes convolucionales a las bases de datos, se va a realizar un par de modificaciones a dichas bases de datos:

- En las pruebas de las termografías, se han reescalado los valores originales, que tienen un rango de [-1,1], a un rango de valores de [0,1]. Esto es debido a que, en las pruebas realizadas, la función de activación *ReLU*, era incapaz de aprender a causa de esos valores negativos y la función *ELU* se encontraba frecuentemente el problema de desvanecimiento de gradiente.

Como se puede apreciar en la Figura 25, los datos de las termografías tienen una gran cantidad de ruido, con la intención de simular el ruido que producen las cámaras termográficas.

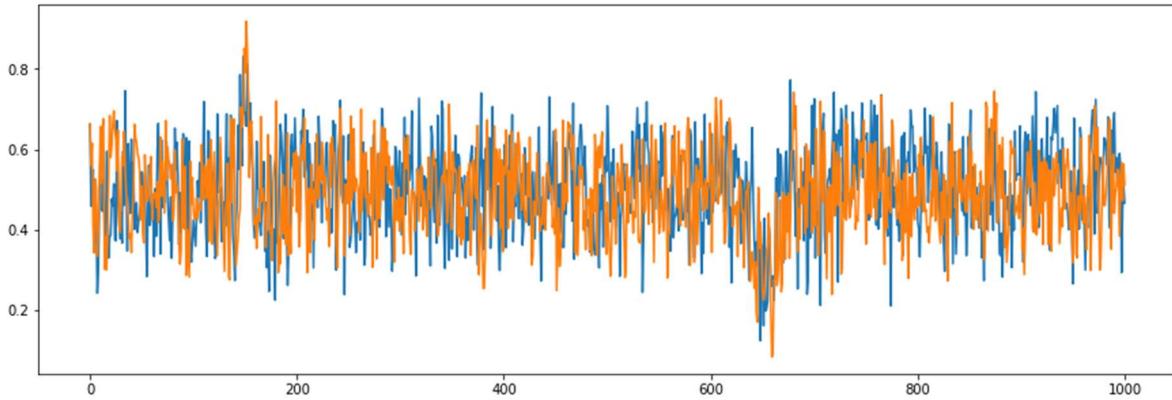


Figura 25: Datos de dos termografías reescaladas con el mismo defecto, pero diferente ruido.

- En cuanto a las derivadas topológicas se ha realizado el mismo procedimiento puesto que encontramos el mismo problema, es decir, el rango de valores original es $[-1,0]$ y tenemos que reescalarlo a $[0,1]$. En la Figura 26 se muestran varias derivadas topológicas correspondientes a placas con un mismo defecto situado a distintas profundidades. Éstas se han centrado en la zona anómala cercana al defecto como se ha visto en la Figura 5.

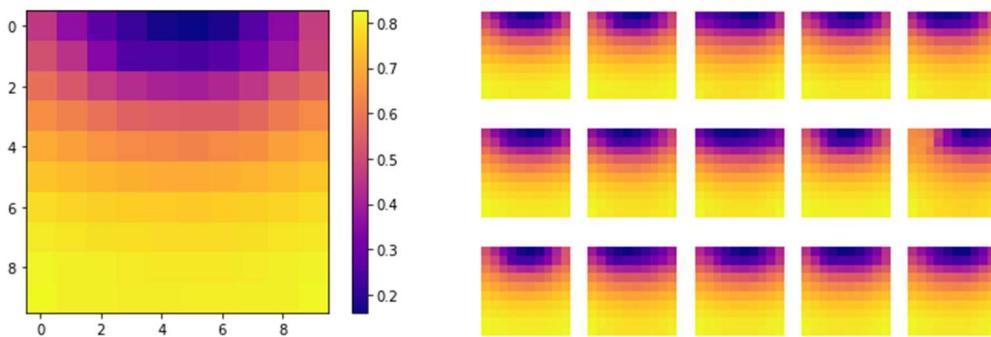


Figura 26: Datos reescalados de diferentes derivadas topológicas en la zona cercana al defecto.

6.2 Estructura de la red neuronal y sus parámetros

Estructura de la red neuronal

Respecto a la red neuronal “*fully-connected*” posterior a las capas convolucionales, se han probado 3 estructuras neuronales, inspiradas en las neuronas biológicas: divergente, paralela y convergente. Véase la Figura 27 para una ilustración gráfica de estas tres estructuras:

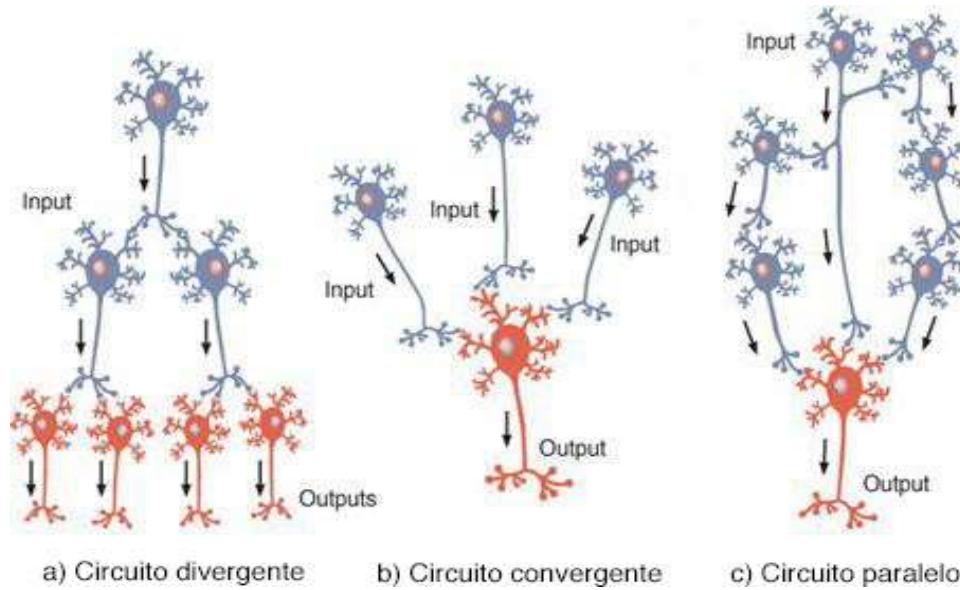


Figura 27: Estructuras neuronales biológicas.

En este TFG se han probado redes de los tres tipos mencionados anteriormente:

- En nuestro caso, el circuito divergente empieza con pocas neuronas por capa y se van incrementando estas a lo largo de las capas. La última capa está formada por una única neurona sigmoide conectada a 2048 neuronas *ELU* de la anterior capa. En las pruebas realizadas, se ha visto un mal comportamiento de esta estructura para entrenar la red neuronal, debido a que en las primeras capas hay muy pocas neuronas, perdiendo información en el proceso.
- El circuito paralelo consiste en una sucesión de capas con la misma cantidad de neuronas. En las pruebas se ha visto un buen comportamiento (similar a la convergente) pero que requiere mucha capacidad computacional.
- Finalmente, se ha elegido un circuito convergente para construir la red de este trabajo, por su buen comportamiento requiriendo menor capacidad computacional que el método paralelo. En la Tabla 2 se muestran las características de las neuronas de las capas *fully-connected*, posterior a las capas convolucionales, junto a sus respectivas funciones de activación.

Capa	Nº de neuronas	Función de activación
1	2048	Exponential Linear Unit (<i>ELU</i>)
2	1024	<i>ELU</i>
3	512	<i>ELU</i>
4	256	<i>ELU</i>
5	128	<i>ELU</i>
6	64	<i>ELU</i>
7	32	<i>ELU</i>
8	1	Sigmoide

Tabla 2: Características de las neuronas que forman la red *fully-connected* posterior a las capas convolucionales.

Función de activación

Como se puede observar en la Tabla 2, se ha utilizado la función de activación *ELU* para las neuronas de las capas intermedias, puesto que en las pruebas se ha visto que su comportamiento es ligeramente mejor que el de la función de activación *ReLU*. En la última capa, al tratarse de la capa resultado, con un valor fijo entre [0,1], se ha optado por la activación sigmoide, aunque otras funciones como el arco-tangente, softsign o tangente hiperbólica han dado resultados similares.

Optimización

En cuanto al optimizador, se ha elegido el método del *descenso acelerado de Nesterov (NAG)* junto a un *momento* con el valor $\eta = 0.9$. Como se ha explicado en apartado 4.4, en la ecuación (4.12), este método acelera la convergencia del descenso de gradiente acumulando el tamaño de paso (α) en cada iteración. De esta manera es capaz de llegar al mínimo global en muy pocas iteraciones, mientras la corrección de *Nesterov* minimiza de manera muy efectiva el *overfitting* que pueda surgir con este método.

Función de coste

Se ha elegido el *error cuadrático medio* como función de coste, puesto que es con la que se ha llegado al mínimo global de manera más precisa. En las pruebas, también se ha podido apreciar que el *error absoluto medio* o el *error cuadrático logarítmico medio* pueden dar resultados muy positivos, más concretamente el último mencionado, aunque ambos tienen el problema de ser imprecisos a la hora de identificar la posición de los defectos más profundos.

En cuanto a la medición del error en la red neuronal, se ha optado por el *error absoluto medio*, pues éste nos da un valor más interpretable para visualizar la precisión de nuestra red.

Learning Rate (ratio de aprendizaje)

Uno de los puntos importantes a la hora de realizar un aprendizaje rápido y eficaz es optar por una ratio de aprendizaje correcta. Como ya se ha mencionado, el objetivo de este estudio es encontrar la mejor manera de entrenar a la red neuronal y realizarlo de la forma más rápida posible. Por ello, se ha decidido optar por la ratio de aprendizaje exponencial decreciente, puesto que tiene tanto las ventajas en velocidad de aprendizaje de una ratio alta, como la precisión de una ratio baja. En los resultados se puede apreciar una mejora sustancial del tiempo de aprendizaje, llegando incluso a 2 órdenes de magnitud menos. En nuestros modelos usaremos la expresión (4.13), mencionada anteriormente:

$$Lr = Lr_{inicial} * e^{-k*epoch} \quad (4.13)$$

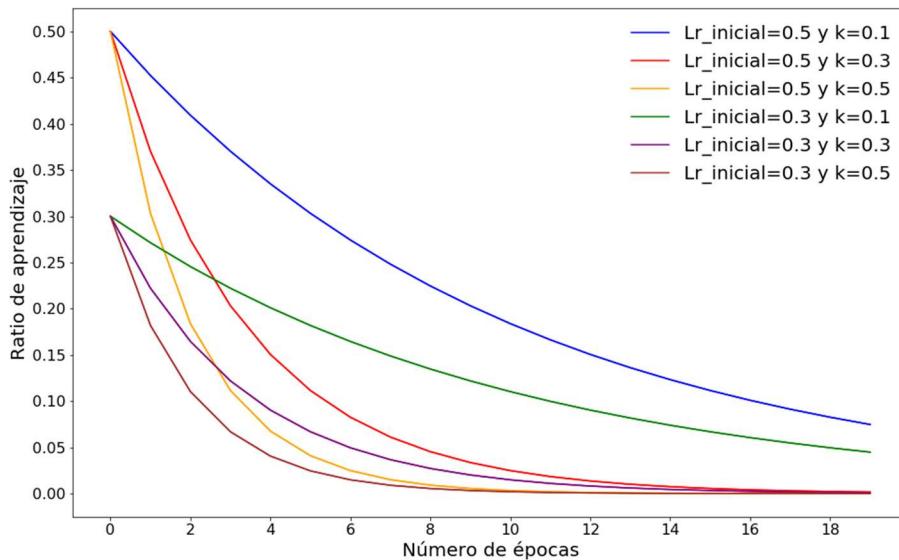


Figura 28: Diferentes modelos de ratios de aprendizaje exponencial decreciente.

En la Figura 28 se puede observar varias combinaciones de ratios de aprendizajes exponencialmente decrecientes. En nuestras pruebas, se ha escogido distintas ratios de aprendizaje para los diferentes modelos de red neuronal, puesto que el tamaño de lote o las capas convolucionales modifican la ratio de aprendizaje óptima. El parámetro $Lr_{inicial}$ permite controlar la ratio del aprendizaje en las primeras épocas, mientras que el parámetro k controla la rapidez a la que ésta desciende. Se podría poner una k muy pequeña para asegurar que la red entrene adecuadamente, pero haría al sistema poco eficiente. Por lo tanto, se ha tenido que buscar manualmente la óptima combinación de parámetros de la ratio de aprendizaje para cada modelo de red neuronal. En el apartado 8 se especificará los parámetros utilizados para cada uno.

Batch (lote) y Batch Normalization.

Utilizar la agrupación de experimentos para la realización del entrenamiento se debe a dos motivos esenciales: primero, con un lote adecuado de agrupación, se puede ahorrar mucho tiempo a la hora de entrenar la red, y segundo, porque de esta manera, se consigue alcanzar el mínimo de manera más directa lidiando con el *overfitting* que puede aparecer si usamos lotes muy pequeños. Como ilustración de este hecho, se muestra en la Figura 29 un ejemplo extraído del artículo [41], donde se puede observar claramente que una agrupación de experimentos pequeña ($B=16$) produce un entrenamiento muy errático, mientras que con tamaños de agrupaciones mayores ($B=1024$), aumenta tanto la precisión del entrenamiento como la estabilidad del mismo.

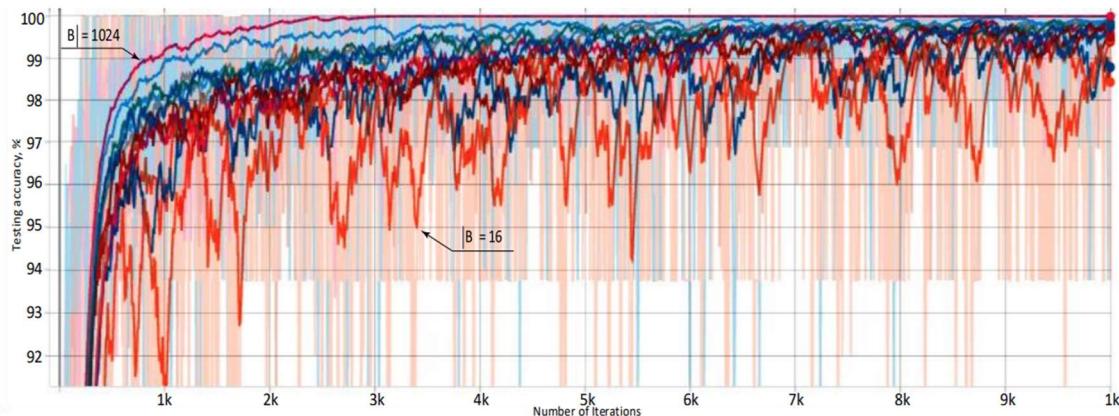


Figura 29: Ejemplo teórico de cómo una correcta agrupación de experimentos ayuda al entrenamiento. Fuente: [41]

En nuestro caso encontramos que un *Batch* demasiado alto nos obliga a entrenar la red durante muchas iteraciones para llegar al mínimo. Como el objetivo es disminuir el tiempo de nuestro entrenamiento, se va a optar por un tamaño de lote comedido.

Tras realizar diferentes pruebas con diferentes tamaños de lotes, se ha decidido entrenar la red en lotes de 50 experimentos para ambos casos, es decir, tanto cuando los datos sean termografías como cuando correspondan a derivadas topológicas. En un primer escenario se puede pensar que es un número de experimentos por lote demasiado pequeño y que puede dificultar el aprendizaje de la red. Pero se busca finalizar el entrenamiento en pocas iteraciones, y para lidiar con el problema del *overfitting*, se va a hacer uso de un *Learning Rate* exponencial decreciente adecuado además del *Batch Normalization*. Este último, ha sido indispensable para tener una mejor precisión a la hora de entrenar y mantener un entrenamiento estable sin sobresaltos por correcciones excesivas, gracias a la optimización del cambio de covariables interno de la red [26]. También hay que mencionar que, gracias a esta técnica, desaparecen los problemas recurrentes de *desvanecimientos de gradiente* al utilizar un *Learning Rate* alto como ilustra la Figura 30.

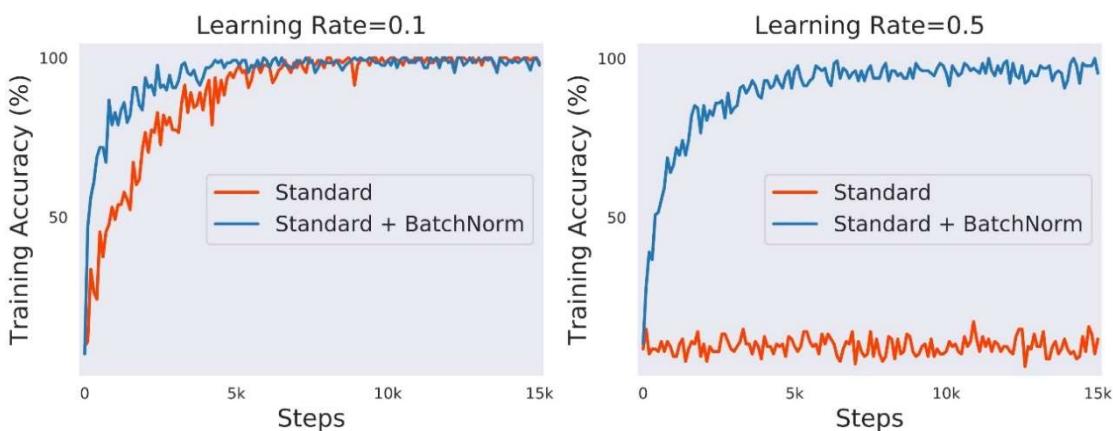


Figura 30: Ejemplo teórico del aporte del *Batch Normalization* al entrenamiento de redes neuronales. Fuente: [27]

Como se ha comentado en el apartado 4.8, pese a no saber claramente la causa, el *Batch Normalization* ayuda mucho al entrenamiento de las redes neuronales. En la anterior Figura 30, la gráfica de la izquierda ilustra claramente cómo el *Batch Normalization* mejora la precisión y la velocidad de entrenamiento cuando se trata de una

ratio de aprendizaje pequeña. En la imagen de la derecha, se muestra que, utilizando una ratio de aprendizaje grande, una red estándar se encuentra con el desvanecimiento de gradiente, mientras que esta técnica posibilita dicho entrenamiento.

En resumen, se ha optado por un *Learning Rate* exponencial decreciente, con un tamaño de lote (*Batch*) de 50 unidades junto al *Batch Normalization* y para la optimización, el método del *descenso acelerado de Nesterov (NAG)* junto a un *momento* con el valor $\eta = 0.9$. Todo ello, con el fin de conseguir que la red neuronal aprenda de manera muy rápida, eficaz y evitando el *overfitting*, *underfitting* y *desvanecimiento del gradiente* que se ha encontrado previamente.

6.3 Aplicación de las capas convolucionales.

En este apartado se explicará la aplicación de las capas convolucionales de la red neuronal junto a las capas que la acompañan para mejorar su comportamiento en las dos situaciones que analizaremos en este TFG: cuando el input sea una base de datos formada por termografías o cuando sea una base de datos formada por derivadas topológicas. También se analizará la repercusión de cada una de estas mejoras en la red convolucional.

Filtros convolucionales

Después de numerosas pruebas, se ha llegado a la conclusión de que, para lograr una mayor precisión con el tiempo limitado que se posee, una elección satisfactoria es aplicar **dos capas de 64 filtros** en cada capa para las dos situaciones (termografías o derivadas topológicas). El tamaño de filtro en las termografías, al ser unidimensional, es de 3x1 y en las derivadas topológicas es de 3x3. En cuanto a la selección de los filtros, en vez de utilizar unos filtros específicos, se ha optado por que la propia red neuronal busque los filtros óptimos que minimicen el error final, es decir, tratar a cada uno de los valores de las matrices filtro como una neurona de la red y utilizar la técnica del *backpropagation* para su optimización [42]. Para ello, se ha seleccionado la función de activación *ReLU* para ambas capas convolucionales. Finalmente, se ha establecido un salto (*stride*) de filtro de una unidad, para no perder precisión.

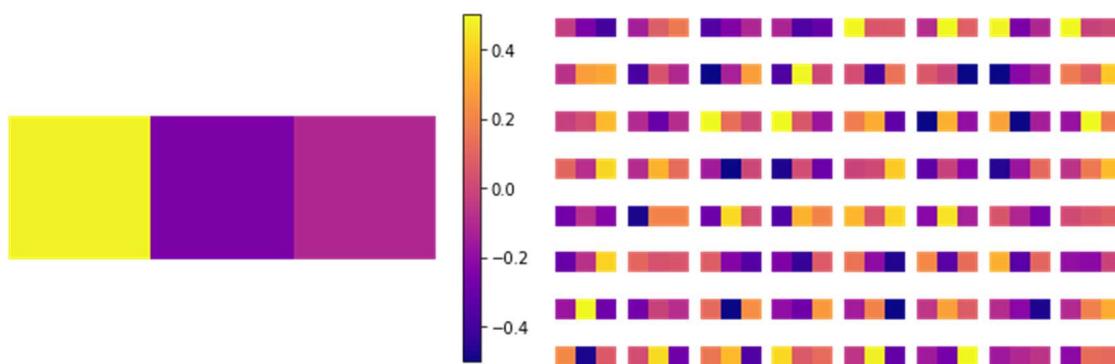


Figura 31: Filtros de la primera capa convolucional para la base de datos formada por termografías.

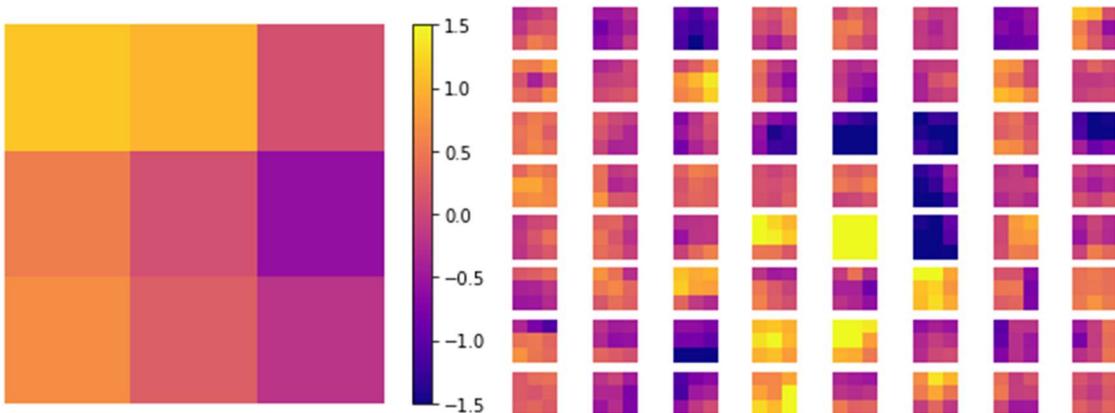


Figura 32: Filtros de la primera capa convolucional para la base de datos formada por las derivadas topológicas.

En las figuras 31 y 32 se muestra gráficamente los valores de los 64 filtros de la primera capa convolucional que ha seleccionado la red neuronal para optimizar el entrenamiento. Hay que tener en cuenta que este proceso de selección de filtros puede dar lugar a diferentes filtros en dos entrenamientos con los mismos parámetros. Esto es debido a que el entrenamiento inicializa de manera aleatoria, por lo que la optimización puede variar en cada caso.

Al ser la red neuronal la que, mediante el entrenamiento elige los filtros óptimos, estos pueden cambiar entre un entrenamiento y otro. Aun así, las imágenes anteriores pueden mostrar una idea sobre qué tipo de filtros decide aplicar la red neuronal a los datos en cada una de las dos situaciones consideradas.

Los 64 filtros que se aplican a las bases de datos suponen un gran incremento en el tamaño de los datos que procesa la red neuronal. Cada uno de los filtros tiene la misión de crear para cada imagen otra nueva. Y si extrapolamos esto a dos capas significa que, por cada imagen de entrada, se tendrá 4096 nuevas imágenes. Esto resulta en más de 400.000 millones de datos para las termografías y más de 10.000 millones de datos en las derivadas topológicas. Es por esta ingente cantidad de datos por la que gran parte de este trabajo se ha centrado en implementar técnicas para acelerar el entrenamiento.

Padding

Uno de los problemas fundamentales que aparecen con imágenes pequeñas, como es el caso de las derivadas topológicas, es que la aplicación de los filtros hace que se pierda cierta información de los bordes. Aunque, en las pruebas se ha visto una mejora poco sustancial de los resultados con la aplicación del *padding*. En cuanto a las termografías al tratarse de un vector tan largo, la repercusión del *padding* en su entrenamiento es prácticamente nulo.

Pooling

En las pruebas, se ha visto que el *average pooling* es muy dañino para la red neuronal empeorando drásticamente la precisión. Se trata de un resultado esperado, puesto que, con este método, al promediar los valores seleccionados, se difumina la información, haciendo que sea más difícil para la red neuronal distinguir posibles patrones.

El *max-pooling*, a cambio de una pequeña disminución de precisión, disminuye el tiempo de entrenamiento hasta en un orden de magnitud.

Capítulo VII

Resultados y discusiones

En este apartado se mostrarán los resultados obtenidos, tras entrenar diferentes modelos de redes neuronales, tanto para las imágenes termográficas como las derivadas topológicas. Los modelos se han escogido según las combinaciones de las técnicas mencionadas en los capítulos 5 y 6, siguiendo estructura que se presenta en la Figura 33.

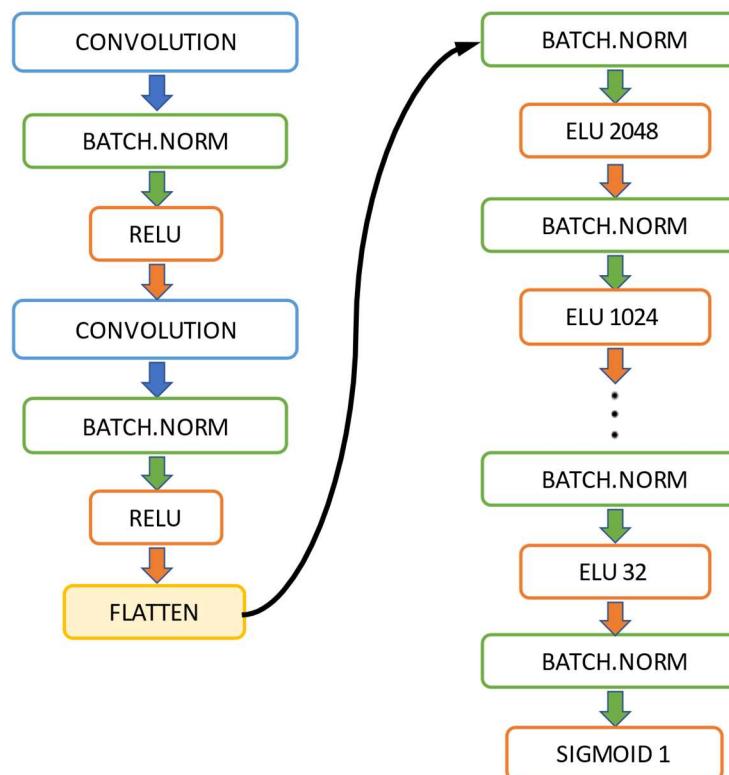


Figura 33: Estructura de la red neuronal convolucional que se ha utilizado en las pruebas.

En la Figura 33, en la columna de la izquierda, podemos ver dos convoluciones con sus respectivos *Batch Normalization* y función de activación. La columna de la izquierda consiste en una red *fully-connected* estructurada de la forma que se indica en la Tabla 2.

7.1 Termografías:

Teniendo en cuenta que todas las redes propuestas en este trabajo contienen una red *fully-connected* (segunda columna de la Figura 33), en la Tabla 3 se muestran las diferentes combinaciones que se han propuesto para modelar las redes neuronales dedicadas a las imágenes termográficas.

Modelo	L.R. inicial	k	Momento + Nesterov	Capas convolucionales	max-pooling	Batch	N. de épocas	Error promedio	Tiempo
1	0.01	0				50	50	5.57%	7min
2	0.3	0.5				50	3	5.35%	30seg
3	0.3	0.5	✓			50	3	5.30%	30seg
4	0.5	0.1		✓	✓	50	3	5.15%	90seg
5	0.5	0.1	✓	✓	✓	50	30	4.30%	15min
6	0.5	0.1	✓	✓		50	30	3.87%	33min

Tabla 3: Diferentes modelos de redes neuronales para las termografías.

En los resultados obtenidos de la Tabla 3 se puede apreciar varios detalles:

- El impacto al usar la ratio de aprendizaje exponencial decreciente ha sido muy significante. Se puede apreciar que el modelo 2, en solamente 3 épocas de entrenamiento, consigue mejores resultados que el modelo 1 usando 50 épocas.
- La aplicación de las capas convolucionales mejora la precisión respecto a los modelos que no las utilizan, aunque se puede apreciar que los tiempos necesarios para el entrenamiento de redes convolucionales es mucho mayor que sin ellas.
- La técnica del descenso acelerado de Nesterov (NAG) junto a un momento con $\eta = 0.9$, ha tenido un impacto muy positivo para las redes con capas convolucionales (modelo 5).
- En cuanto al *max-pooling*, pese a ser un método muy eficaz para disminuir el tiempo de entrenamiento, en el modelo 6 se puede apreciar una notable mejora en cuanto a la precisión de la red si eliminamos esta técnica.

Por lo tanto, solamente observando el error promedio obtenido, se puede apreciar que, utilizando un correcto optimizador, una adecuada ratio de aprendizaje y capas convolucionales, se consigue mejorar sustancialmente la precisión del sistema.

Para poder visualizar con más claridad las predicciones obtenidas, en las Figuras 30 al 35, se ha utilizado una media móvil 100 centrada. Esta técnica realiza, para cada punto, una media aritmética con sus 50 valores posteriores y 50 anteriores que se expresa de la siguiente manera:

$$l_i = \frac{1}{n+1} \sum_{j=0}^n p_{(j-\frac{n}{2})+i} \quad (7.1)$$

donde l_i corresponde a cada uno de los valores de esta media, p es el punto correspondiente en la serie de datos y n es en este caso 100. Nótese que la media de cada punto cuenta con 101 valores, puesto que el valor en esa misma posición también se contabiliza. Por lo tanto, n es siempre un número par, por lo que la media cuenta con un

número impar de valores. En este caso, el valor inicial de esta media tiene que empezar en el punto 51, es decir, $l_{initial} = l_{(n/2)+1}$ y el valor final debe acabar 50 puntos antes del valor final de la serie, $l_{final} = l_{m-(n/2)}$, siendo m el número total de valores de la serie. Las medias móviles son herramientas muy utilizadas en diversos campos como el análisis de datos, aprendizaje automático, series temporales, trading en bolsa de valores, etc.... Se ha escogido la media móvil 100, además de por ser una de las más frecuentes a la hora de analizar datos, por tener la capacidad de mostrar claramente cómo fluctúa la media sin perder demasiados matices por la excesiva suavización de esta media, como ocurre con medias móviles de mayor número de valores.

Termografía: Modelo 1

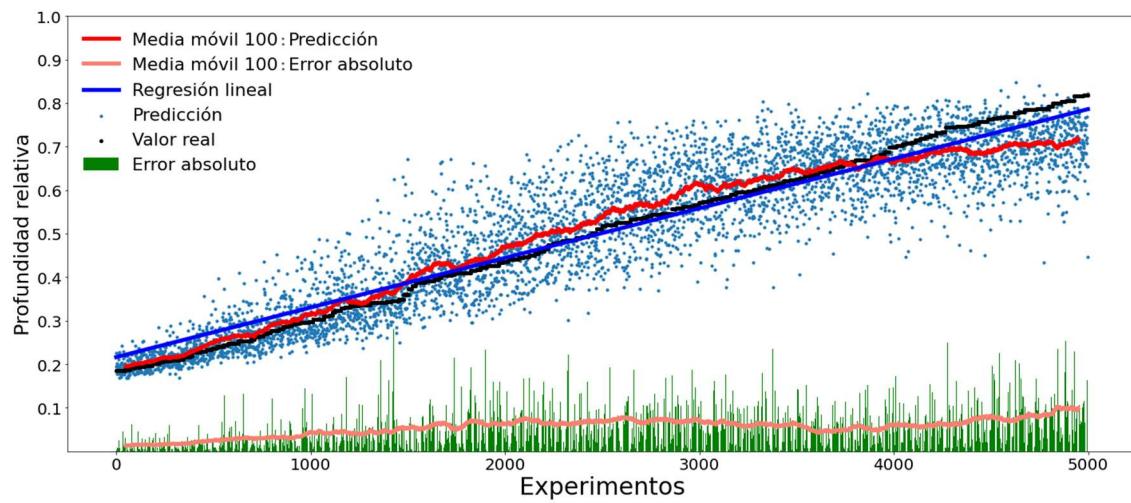


Figura 34: Resultados obtenidos del primer modelo de red neuronal dedicado a las imágenes termográficas.

Termografía: Modelo 2

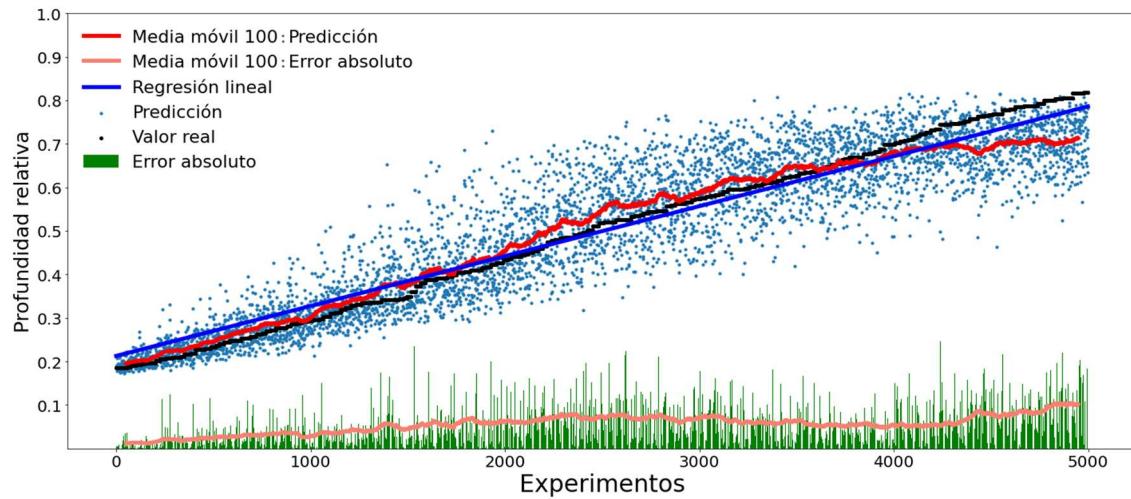


Figura 35: Resultados obtenidos del segundo modelo de red neuronal dedicado a las imágenes termográficas.

Termografía: Modelo 3

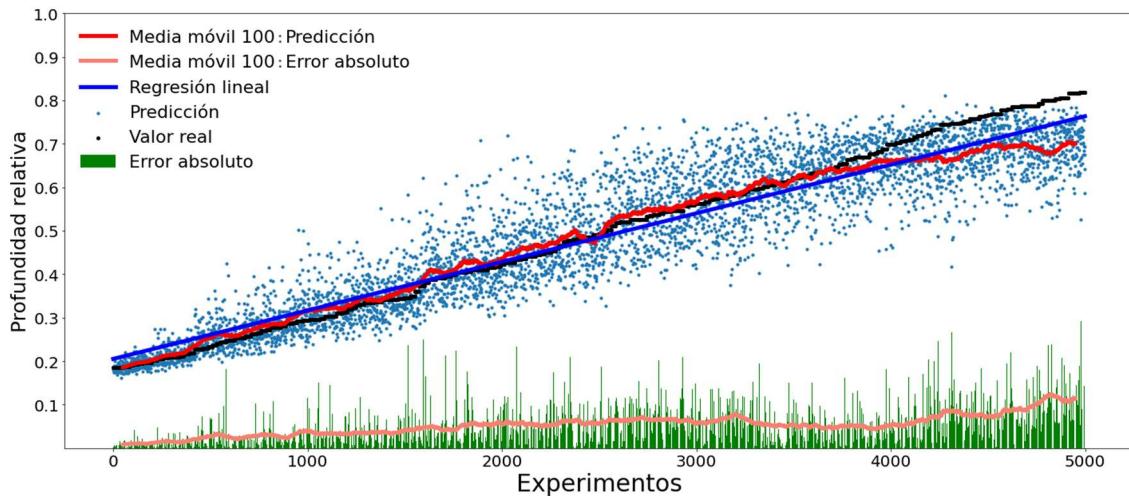


Figura 36: Resultados obtenidos del tercer modelo de red neuronal dedicado a las imágenes termográficas.

Termografía: Modelo 4

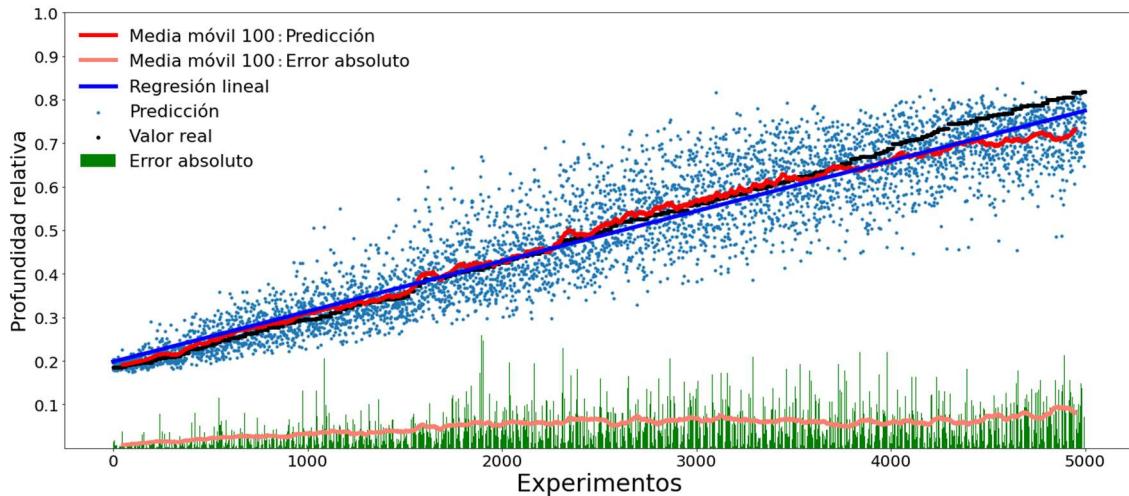


Figura 37: Resultados obtenidos del cuarto modelo de red neuronal dedicado a las imágenes termográficas.

Termografía: Modelo 5

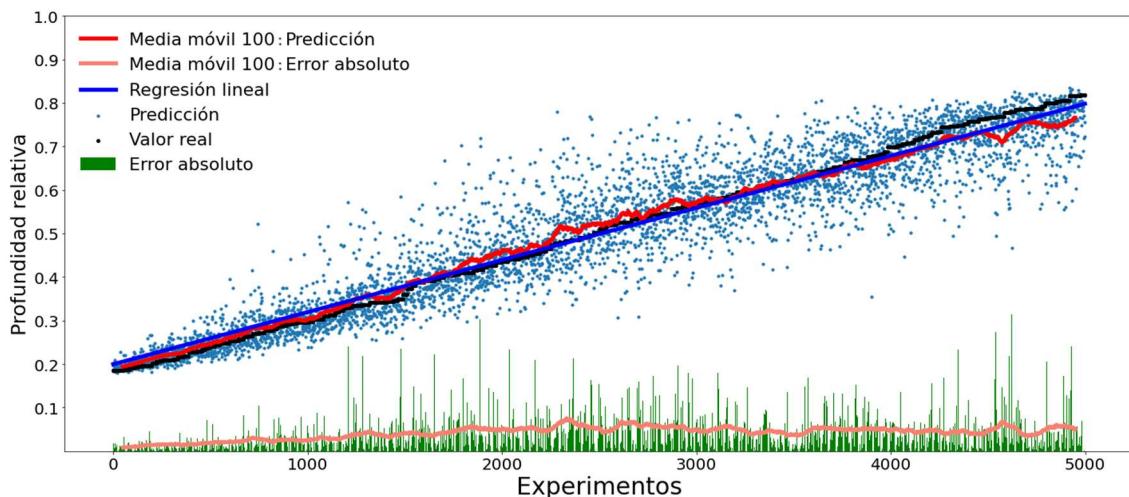


Figura 38: Resultados obtenidos del quinto modelo de red neuronal dedicado a las imágenes termográficas.

Termografía: Modelo 6

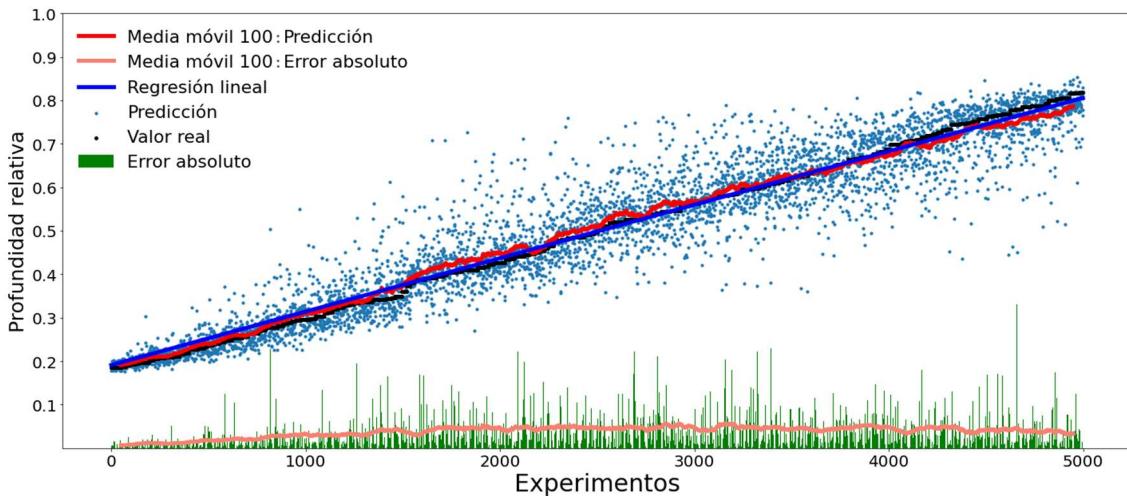


Figura 39: Resultados obtenidos del sexto modelo de red neuronal dedicado a las imágenes termográficas.

Tras estudiar los resultados obtenidos de los seis modelos (Figura 34 al 39), se puede apreciar los siguientes detalles:

- A medida que mejoran los modelos, la dispersión de las predicciones es considerablemente menor. Éstas se concentran más cerca de los valores reales y se puede visualizar perfectamente si comparamos el Modelo 6 y el Modelo 1.
- La línea roja, que representa una media móvil 100 centrada, nos indica cómo está siendo el promedio de las predicciones respecto a los valores reales (puntos negros). Y se puede apreciar que, en los tres últimos modelos, correspondientes a las redes que utilizan capas convolucionales, esta media se comporta mucho mejor que en los modelos sin dichas capas.

Una de las premisas que se traía de los trabajos anteriores, [1] y [2], es que las redes tienen dificultades a la hora de predecir la posición de los defectos más profundos. Y si nos fijamos en la parte final de las cinco primeras gráficas, nuestra media móvil tiende a alejarse de los valores reales, corroborando esta premisa. Pero si nos fijamos en el Modelo 6, que es nuestro mejor modelo, podemos apreciar que este problema se ha paliado con bastante éxito.

- La línea naranja, que representa una media móvil 100 centrada, nos indica cómo está siendo el promedio de los errores absolutos (barras verdes). Y para visualizar correctamente su comportamiento en los diferentes modelos, se ilustran conjuntamente en la Figura 40:

Error absoluto T: media móvil 100 centrada

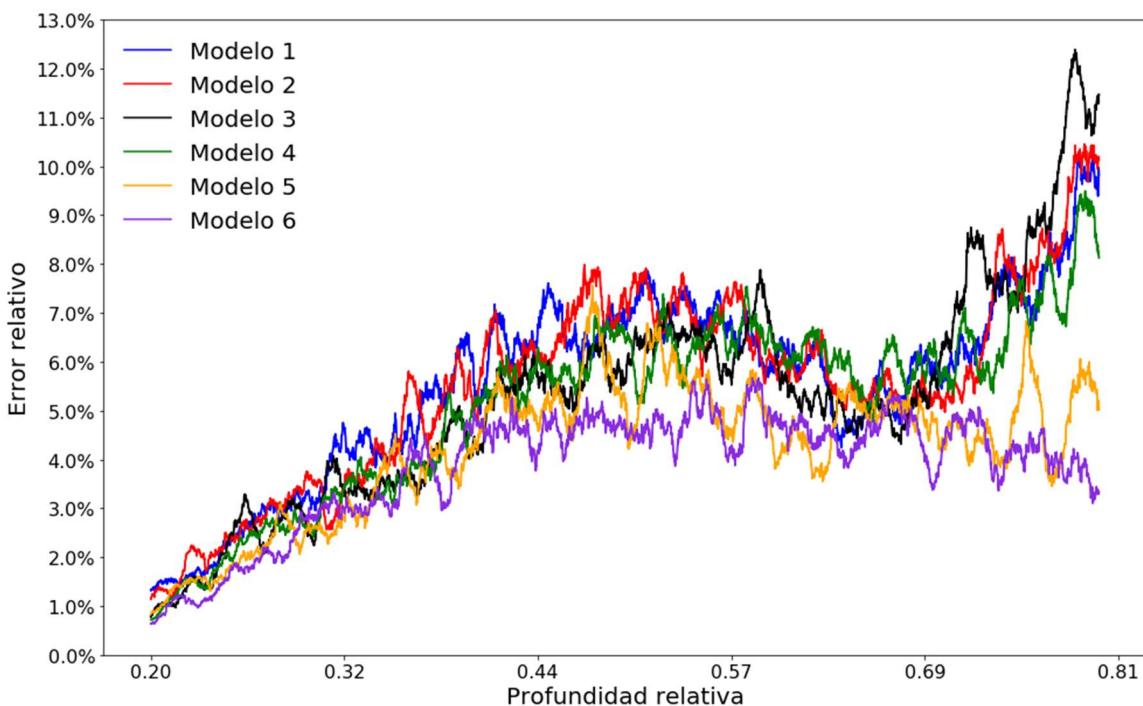


Figura 40: Media móvil centrada de 100 experimentos del error absoluto de los seis modelos de red neuronal para las imágenes termográficas.

En la Figura 40 se aprecia que en general, a mayor profundidad del defecto, los modelos empeoran su precisión. Pero cabe destacar que los Modelos 5 y 6 (líneas morada y amarilla) poseen una precisión notablemente superior, sobre todo a la hora de identificar la posición de los defectos situados a mayor profundidad.

En conclusión, tras estudiar los diferentes modelos de redes neuronales para las termografías, se puede afirmar que las capas convolucionales mejoran notablemente la precisión de las predicciones y pueden solventar el problema del aprendizaje deficitario de los casos en los que los defectos están situados a mayor profundidad.

7.2 Derivadas topológicas:

Para procesar la base de datos que contiene derivadas topológicas vamos a utilizar redes del mismo tipo que las consideradas a la hora de procesar termografías (véase el esquema de la Figura 33), por lo tanto, contendrán como base una red *fully-connected* estructurada de la forma que se indica en la Tabla 2. En la Tabla 4 se puede encontrar las diferentes combinaciones que se han propuesto para modelar cada una de las redes que será utilizada para procesar las derivadas topológicas.

Modelo	L.R. inicial	k	Momento + Nesterov	Capas convolucionales	Padding	Max-pooling	Batch	N. de épocas	Error promedio
1	0.01	0					50	20	3.30%
2	0.1	0.3					50	20	2.23%
3	0.1	0.3	✓				50	20	2.07%
4	0.1	0.4	✓	✓		✓	50	20	2.12%
5	0.1	0.4	✓	✓	✓	✓	50	20	2.17%
6	0.1	0.4	✓	✓	✓		50	20	2.15%
7	0.1	0.4	✓	✓			50	20	2.14%

Tabla 4: Diferentes modelos de redes neuronales para las derivadas topológicas.

En los resultados recogidos en la Tabla 4 se puede apreciar varios detalles:

- Al igual que el caso anterior (compárese con la Tabla 3), tanto la ratio de aprendizaje exponencial decreciente como el descenso acelerado de Nesterov mejoran sustancialmente la precisión de los modelos.
- A diferencia de lo observado en el caso de las termografías, cuando los datos corresponden a derivadas topológicas, las capas convolucionales no parecen impactar significativamente en el aprendizaje de la red, puesto que los Modelos 2 y 3, que no utilizan las capas convolucionales, han sido capaces de conseguir una precisión parecida a los que sí tienen dichas capas.
- En un principio, se pensaba que la técnica del *Padding* impactaría positivamente en el entrenamiento y el *Max-Pooling* negativamente, pero se ha visto que todos los modelos con redes convolucionales se han encontrado con un suelo alrededor del 2% de error.

Mirando las Tabla 3 y 4, se puede afirmar que trabajando con derivadas topológicas se obtienen resultados sustancialmente mejores que con termografías. Las técnicas que se han utilizado para mejorar la ratio de aprendizaje y el descenso del gradiente han impactado muy positivamente en los modelos que actúan sobre derivadas topológicas, aunque la adición de capas convolucionales no ha supuesto un impacto significativo como en los modelos con termografías. El motivo puede deberse al tamaño del propio elemento de entrenamiento, ya que, como se comentó con anterioridad, cada elemento es una matriz de tamaño 10×10 que contiene los valores de la derivada topológica en una pequeña región que contiene al defecto.

A continuación, en las Figuras 41 a 47 se ilustran gráficamente los resultados obtenidos de los diferentes modelos de las redes de la Tabla 4.

Derivada topológica: Modelo 1

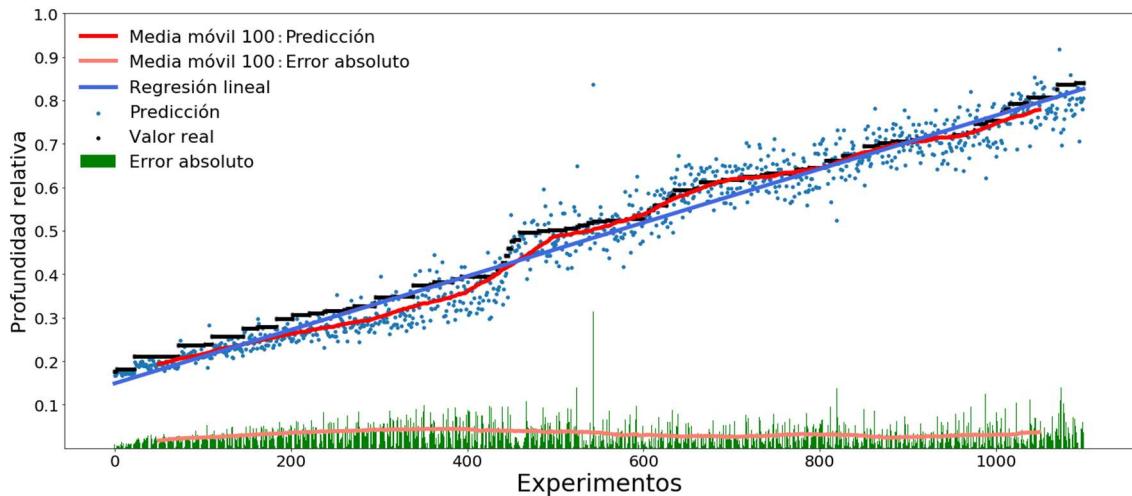


Figura 41: Resultados obtenidos del primer modelo de red neuronal dedicado a las derivadas topológicas.

Derivada topológica: Modelo 2

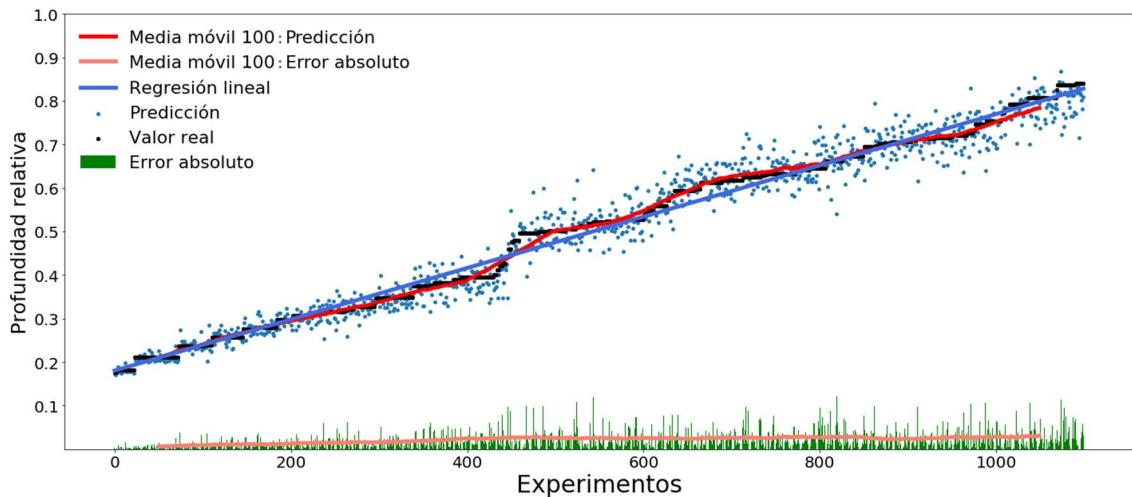


Figura 42: Resultados obtenidos del segundo modelo de red neuronal dedicado a las derivadas topológicas.

Derivada topológica: Modelo 3

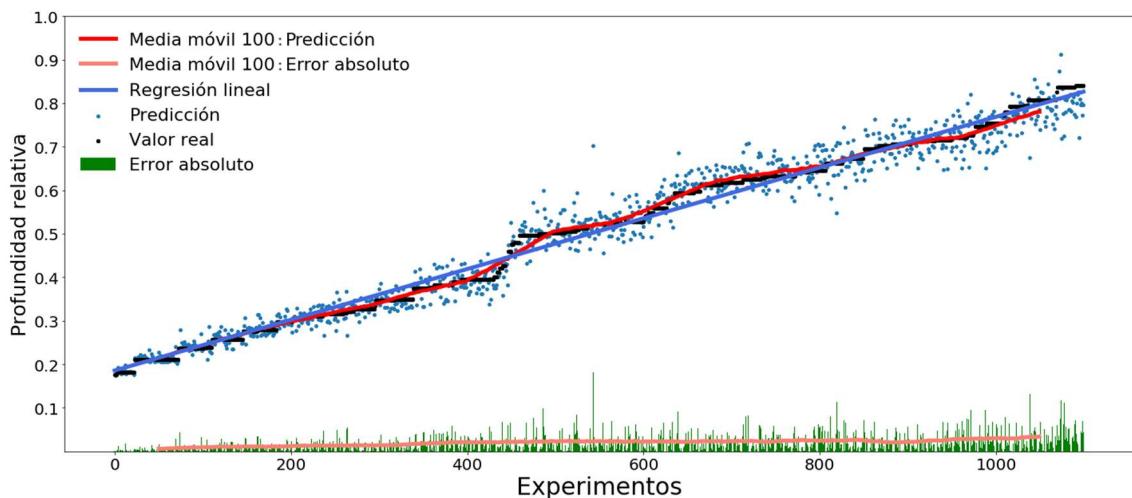


Figura 43: Resultados obtenidos del tercer modelo de red neuronal dedicado a las derivadas topológicas.

Derivada topológica: Modelo 4

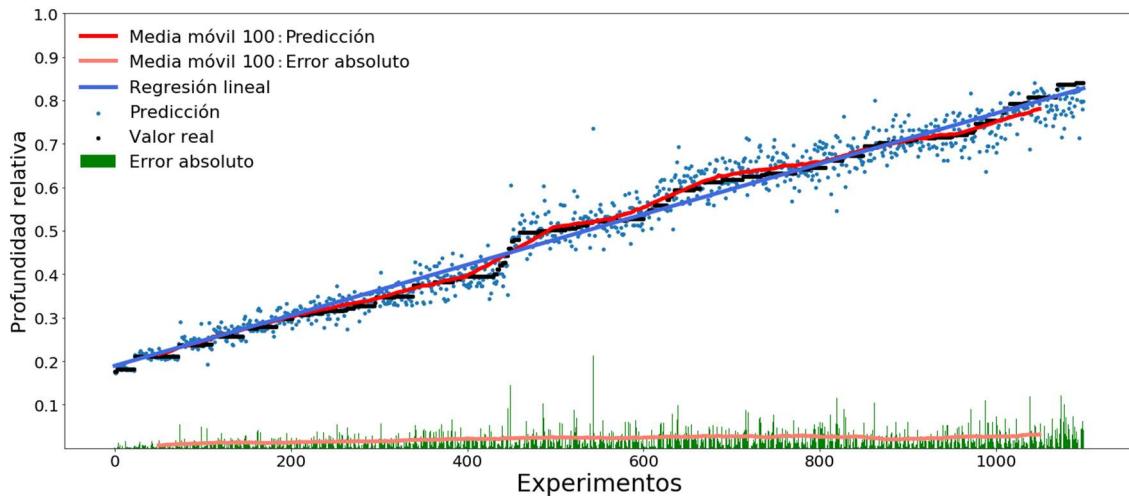


Figura 44: Resultados obtenidos del cuarto modelo de red neuronal dedicado a las derivadas topológicas.

Derivada topológica: Modelo 5

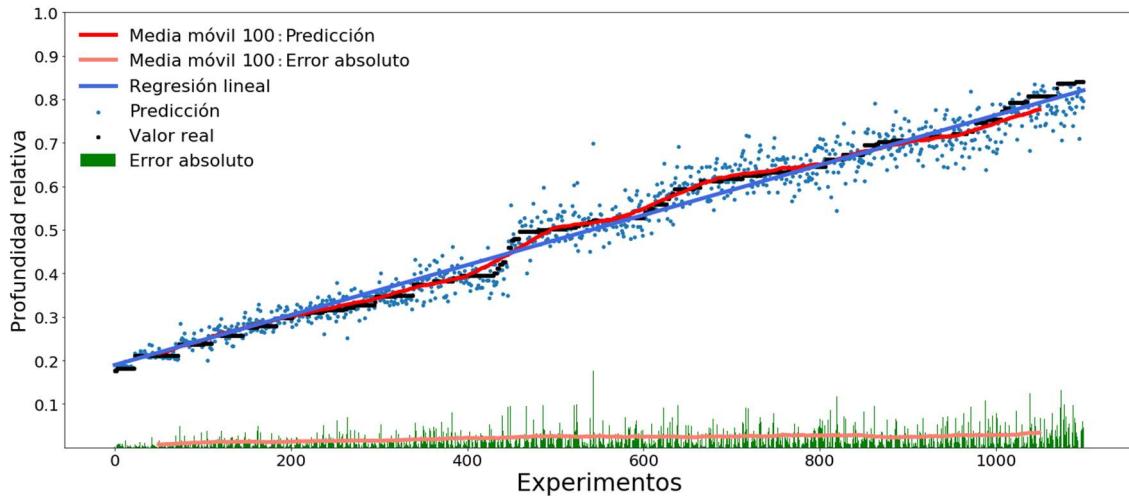


Figura 45: Resultados obtenidos del quinto modelo de red neuronal dedicado a las derivadas topológicas.

Derivada topológica: Modelo 6

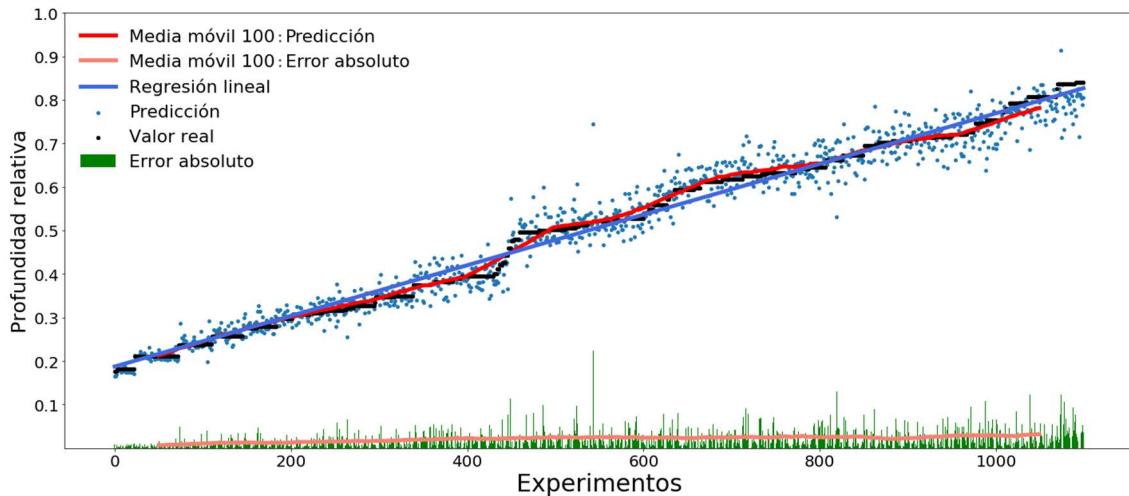


Figura 46: Resultados obtenidos del sexto modelo de red neuronal dedicado a las derivadas topológicas.

Derivada topológica: Modelo 7

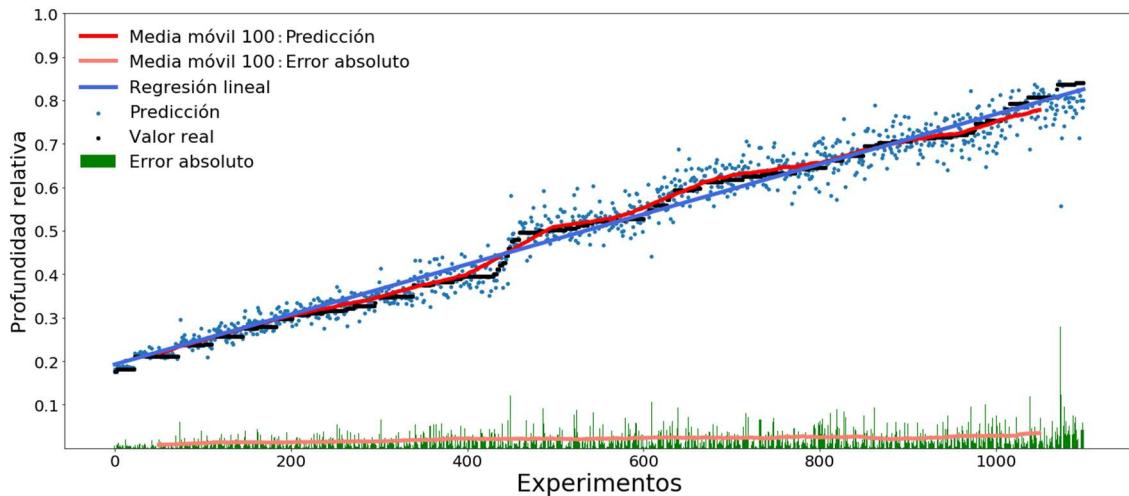


Figura 47: Resultados obtenidos del séptimo modelo de red neuronal dedicado a las derivadas topológicas.

Los resultados obtenidos de los anteriores modelos (Figura 41 a 47), nos permiten apreciar los siguientes detalles:

- La dispersión de las predicciones al trabajar con derivadas topológicas es mucho menor en cualquiera de los modelos que la obtenida al trabajar directamente con las termografías.
- La media móvil 100 centrada (línea roja), nos indica cómo está siendo el promedio de las predicciones respecto a los valores reales (puntos negros). Y se puede apreciar que, en el Modelo 1, esta media se encuentra por debajo de los valores reales en los defectos situados a profundidades pequeñas. Este comportamiento también se encuentra en los modelos de las termografías, pero con los defectos más profundos.
- La línea naranja, que representa una media móvil 100 centrada, nos indica cómo está siendo el promedio de los errores absolutos (barras verdes). Y para visualizar correctamente su comportamiento en los diferentes modelos, se ilustran conjuntamente en la Figura 48:

Error absoluto DT: media móvil 100 centrada

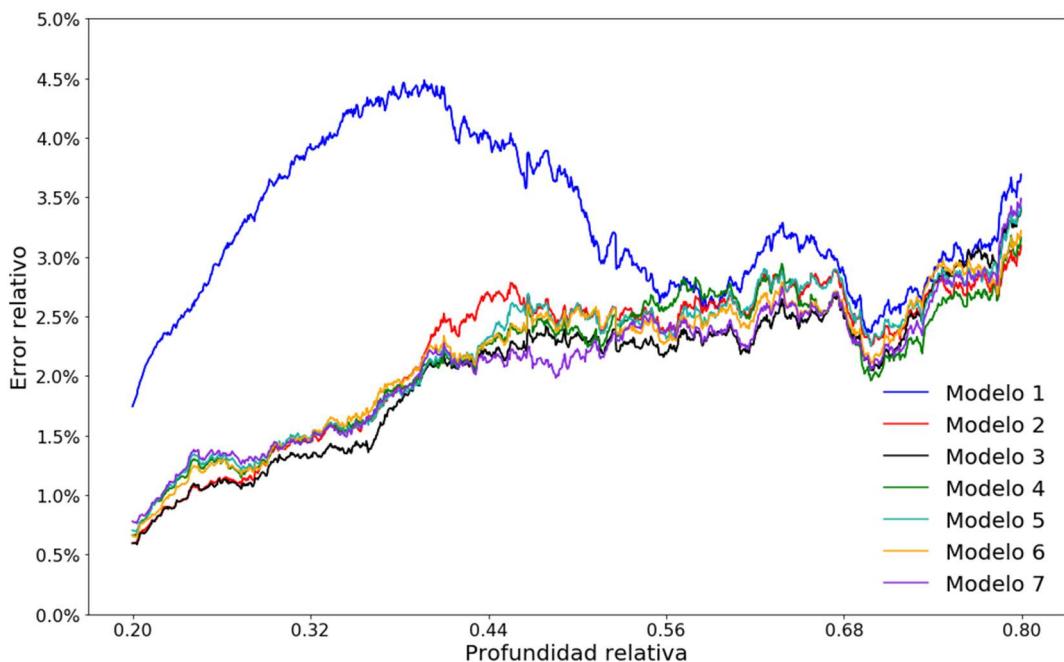


Figura 48: Media móvil centrada de 100 experimentos del error absoluto de los siete modelos de red neuronal para las derivadas topológicas

En vista de los datos recogidos en la Figura 48, se puede afirmar que todos los modelos presentan errores mucho menores a los obtenidos al considerar las termografías sin postprocesar vía la derivada topológica (compárese con la Figura 40). Se puede apreciar que todos los modelos del 2 al 7 empeoran su precisión a medida que la posición del defecto se hace más profunda, aunque el Modelo 1 parece haber tenido dificultades a la hora de aprender los experimentos con defectos a menores profundidades.

En conclusión, los diferentes modelos de redes neuronales para las derivadas topológicas tienen una precisión mucho mayor a los modelos de las termografías. Las capas convolucionales en estos modelos no consiguen una mejora sustancial para el aprendizaje de la red neuronal, aunque las técnicas de la ratio de aprendizaje exponencial decreciente y el descenso acelerado de *Nesterov* sí han contribuido significativamente a mejorar el entrenamiento de la red, solventando el problema del aprendizaje deficitario en los casos en los que los defectos están situados a menor profundidad (Modelo 1).

Como resumen final, hay que destacar que los resultados obtenidos provienen de termografías con apreciable ruido donde era muy complicado detectar el lugar de los fallos. Con las derivadas topológicas era posible situar los defectos en altura, pero no se obtenía la información sobre su profundidad. A partir de los resultados que se obtuvieron de las redes neuronales utilizadas en los trabajos de Sergio Colubi [1] y Ángel Mateo [2], se observó que los modelos de redes con derivadas topológicas salían muy beneficiados respecto a los modelos con termográficas. Con las mejoras implementadas en este TFG: las capas convolucionales, la ratio de aprendizaje exponencial decreciente, el descenso

acelerado de *Nesterov*, el *Batch Normalization*, la estructura convergente de la red neuronal, etc., en nuestros mejores modelos, gracias a estas técnicas, se ha conseguido unos resultados con errores medios por debajo del 4% para los modelos con termografías y sobre el 2% para los modelos con derivadas topológicas, todo ello con entrenamientos de menos de 50 épocas.

En el trabajo de Sergio Colubi [1], en una de las redes neuronales que construye, en concreto la red 13.1 (su red neuronal más efectiva), se realiza un entrenamiento de 1000 épocas red simple *fully-connected* consiguiendo unos resultados muy precisos. Esta red consigue un error medido en error cuadrático medio de $4.7899 \cdot 10^{-4}$.

Para comparar la efectividad de nuestras redes, se ha realizado un entrenamiento con la misma base de datos usada por Sergio, con una red neuronal que tiene la configuración del Modelo 6 de la Tabla 4, y se ha visto que consigue llegar al mínimo en únicamente 20 épocas. Los errores resultantes son los mostrados en la Tabla 5.

Error cuadrático medio	$4.2051 \cdot 10^{-4}$
Error absoluto medio	1.48 %

Tabla 5: Resultados obtenidos con el modelo 6 de red neuronal aplicado a la base de datos utilizada por Sergio en la red 13.1 del trabajo [1].

Por lo tanto, se puede apreciar una pequeña mejora alrededor del 12%, aunque el punto a destacar es el hecho de haber conseguido una precisión superior, con un número de épocas de entrenamiento 50 veces inferior. En la Figura 49, se compara los resultados obtenidos con el Modelo 6 respecto a la red 11.3 del trabajo de Sergio. Y como comentamos anteriormente solo hay una mejora alrededor del 12%, por lo que es difícil apreciarlo gráficamente.

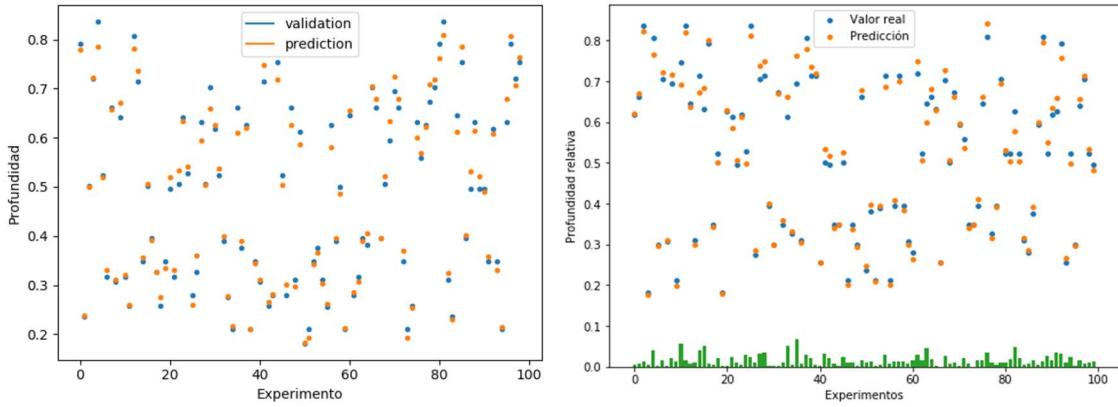


Figura 49: Comparativa entre resultados de la red 13.1 del trabajo de Sergio [1] (izquierda), y los obtenidos con el modelo 6 de red neuronal aplicadas a la misma base de datos (derecha).

Capítulo VIII

Conclusiones y líneas futuras

Conclusiones

A continuación, se enumerarán varias conclusiones a las que se ha llegado a lo largo del trabajo:

- Las capas convolucionales han realizado una gran aportación a la hora predecir la profundidad de los defectos, y han resultado muy efectivas, sobre todo en las imágenes termográficas, incluso cuando los datos de entrenamiento poseen un ruido muy variado.
- La ratio de aprendizaje exponencial descendente ha sido fundamental para conseguir un entrenamiento efectivo y rápido que nos ha permitido disminuir el tiempo de entrenamiento incluso en dos órdenes de magnitud.
- En cuanto a la optimización, el uso del descenso acelerado de *Nesterov* con *momento* ha permitido no solo acelerar el entrenamiento en cuestión sino también a evitar el *overfitting* que aparece en los entrenamientos.
- El *Batch Normalization* ha sido sorprendentemente eficaz, puesto que sin ello no habría sido posible realizar entrenamientos con una ratio de aprendizaje tan alto. Además, consigue que el entrenamiento sea mucho más estable y rápido.
- Por lo que se ha podido observar, cuando el resultado de la red neuronal solamente consiste en un único valor (en nuestro caso la profundidad a la que se encuentra el defecto), las redes convergentes resultan ser las óptimas.
- Para la función de activación, se ha comprobado que, en todas las configuraciones realizadas en el trabajo, la función ELU ha tenido un mejor rendimiento que la función ReLU. Esto puede ser debido a que cuando en la red neuronal, los valores de las neuronas se vuelven negativas (debido al sesgo), la función ReLU simplemente desecha el valor de esa neurona, mientras que la función ELU retiene dicha información.
- Las redes neuronales convolucionales han resultado ser mucho más efectivas cuando los elementos a analizar han resultado ser más grandes, como en el caso de las imágenes termográficas que tienen 1000 valores, mientras que las derivas topológicas solamente tienen 100 valores. Por lo tanto, se ha podido ver que estas redes convolucionales son más efectivas cuanto más “dificiles” sean los datos a analizar.

- La utilización de GPUs para el entrenamiento ha sido primordial, puesto que ha disminuido el tiempo de entrenamiento casi en dos órdenes de magnitud y nos ha permitido realizar todas las pruebas que se han propuesto en un tiempo muy comedido. Gracias a ello se ha podido ver cómo afectan los diferentes parámetros y las combinaciones de ellas en nuestras redes neuronales.

Líneas futuras

- Como ya se ha comentado, las redes neuronales diseñadas en este trabajo que actúan sobre derivadas topológicas no consiguen mejorar sustancialmente aplicando capas convolucionales y uno de los posibles motivos puede deberse al tamaño reducido de la propia imagen. Por lo que en futuros trabajos sería ideal entrenar a la red utilizando los valores de la derivada topológica correspondientes a todos los puntos de la placa, en vez de utilizar únicamente unos pocos valores correspondientes a una zona de la placa donde se encuentra el defecto.
- Uno de los problemas fundamentales de los trabajos de entrenamiento de redes neuronales es la cantidad de datos necesarios para realizar el entrenamiento. Por lo que en futuros proyectos sugiero que se utilice un método muy simple para crear más datos a partir de los datos originales. Estas técnicas son la rotación y la simetría a las imágenes de datos. Hay que tener en cuenta que existen muchos ángulos de rotación y que, además, combinados con la simetría, dan lugar a un conjunto muy grande de datos nuevos. Aunque hay que comprobar que los datos resultantes tienen sentido físico y son aptos para el entrenamiento de redes.
- Para próximos proyectos en los que se creen más bases de datos con nuevos experimentos, se puede plantear que se varíen tanto el tamaño como la forma de los defectos (ovalados, romboides, cuadrados, rectangulares...), y comprobar si se puede crear una red neuronal capaz de diferenciar los diferentes tipos de grietas y sus tamaños. Esto permitiría acercar este proyecto a un entorno más realista.
- Extender los resultados de este trabajo a una situación más realista, donde la placa se modele como una placa tridimensional.

Bibliografía y referencias

- [1] S. Colubi. Algoritmos basados en redes neuronales y derivadas topológicas para la detección de daño en estructuras a partir de termografías infrarrojas. *Trabajo Fin de Máster. Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio. Universidad Politécnica de Madrid.* 2020.
- [2] Á. Mateo. Detección de defectos estructurales mediante el análisis de termografías aplicando redes neuronales. *Trabajo de Fin de Grado. Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio. Universidad Politécnica de Madrid.* 2021.
- [3] M. Pena y M.-L. Rapún. Detecting Damage in Thin Plates by Processing Infrared Thermographic Data with Topological Derivatives. *Hindawi.* 2019. <https://doi.org/10.1155/2019/5494795>
- [4] M. Pena y M.-L. Rapún. Application of the topological derivative to post-processing infrared time-harmonic thermograms for defect detection. *Journal of Mathematics in Industry.* 2020. <https://doi.org/10.1186/s13362-020-0072-9>
- [5] A.W. Senior, R. Evans, J. Jumper et al. Improved protein structure prediction using potentials from deep learning. *Nature* 2020. <https://doi.org/10.1038/s41586-019-1923-7>
- [6] D. Balageas, X. Maldague, D. Burleigh et al. Thermal (IR) and Other NDT Techniques for Improved Material Inspection. *Journal Nondestructive Evaluation.* 2016. <https://doi.org/10.1007/s10921-015-0331-7>
- [7] FreeFem documentation. <https://doc.freefem.org/introduction/index.html>
- [8] J. Sokolowski y A. Zochowski. On the Topological Derivative in Shape Optimization. *SIAM Journal on Control and Optimization.* 1999. <https://pubs.siam.org/doi/pdf/10.1137/S0363012997323230>
- [9] M. Higuera, J.M. Perales, M.-L. Rapún y J.M. Vega. Solving inverse geometry heat conduction problems by postprocessing steady thermograms. *International Journal of Heat and Mass Transfer.* 2019. <https://doi.org/10.1016/j.ijheatmasstransfer.2019.118490>
- [10] F. Le Louër y M.-L. Rapún. Topological Sensitivity for Solving Inverse Multiple Scattering Problems in Three-Dimensional Electromagnetism. Part II: Iterative Method. *Society for Industrial and Applied Mathematics Journal on Imaging Sciences.* 2018. <https://pubs.siam.org/doi/pdf/10.1137/17M1148359>
- [11] Características y ventajas del acelerador Tesla K80. <https://www.nvidia.com/es-es/data-center/tesla-k80/>
- [12] Nvidia. What is CUDA? <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/>
- [13] F. Rosenblatt. Perceptron Simulation Experiments. *Proceedings of the IRE.* 1960. <https://doi.org/10.1109/JRPROC.1960.287598>

Bibliografía y referencias

- [14] D. Hebb. The Organization of Behavior. Wiley. 1949. <https://archive.org/details/in.ernet.dli.2015.226341>
- [15] D. Rumelhart, G. Hinton & R. Williams. Learning representations by back-propagating errors. *Nature*. 1986. <https://doi.org/10.1038/323533a0>
- [16] I. Goodfellow, Y. Bengio and A. Courville. Deep Learning. (2016). MIT Press. [Deep Learning \(deeplearningbook.org\)](http://DeepLearning(deeplearningbook.org))
- [17] A. F. Agarap. Deep Learning using Rectified Linear Units (ReLU). *ArXiv, Cornell University*. 2018. <https://arxiv.org/abs/1803.08375>
- [18] D.-A. Clevert, T. Unterthiner & S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ArXiv, Cornell University*. 2015. <https://arxiv.org/abs/1511.07289>
- [19] J. Merino y M.-J. Noriega. *Fisiología General, Tema 7-Bloque III: Señales eléctricas*, página 5. Universidad de Cantabria. <https://ocw.unican.es/pluginfile.php/879/course/section/967/Tema%25207-Bloque%2520II-Senales%2520Electricas.pdf>
- [20] Funciones de activación disponibles en Tensorflow, en su versión 2.1. https://www.tensorflow.org/versions/r2.1/api_docs/python/tf/keras/activations
- [21] Funciones de coste disponibles en Tensorflow, en su versión 2.1. https://www.tensorflow.org/versions/r2.1/api_docs/python/tf/keras/losses
- [22] H. B. Curry. The method of steepest descent for non-linear minimization problems. *Quart. Appl. Math.* 2. 1944. <https://doi.org/10.1090/qam/10667>
- [23] H. Robbins and S. Monro. A Stochastic Approximation Method. *Ann. Math. Statist.* 1951. <https://projecteuclid.org/euclid.aoms/1177729586>
- [24] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks 12. Center for Neurobiology and Behavior, Columbia University*. 1999. [https://doi.org/10.1016/S0893-6080\(98\)00116-6](https://doi.org/10.1016/S0893-6080(98)00116-6)
- [25] Y. Nesterov, A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269, 543-547. 1983.
- [26] S. Ioffe, C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv, Cornell University*. 2015. <https://arxiv.org/abs/1502.03167>
- [27] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry. How Does Batch Normalization Help Optimization? *ArXiv, Cornell University*. 2018. <https://arxiv.org/abs/1805.11604>
- [28] Tipos de redes neuronales más frecuentes. <https://www.asimovinstitute.org/neural-network-zoo/>
- [29] D.H. Hubel, T.N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*. 1959. <https://doi.org/10.1113/jphysiol.1959.sp006308>

- [30] J. Schmidhuber, M. Eldracher, and B. Foltin. Semilinear Predictability Minimization Produces Well-Known Feature Detectors. *The MIT Press Journals*. 1996. <https://doi.org/10.1162/neco.1996.8.4.773>
- [31] B.A. Olshausen, D.J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*. [https://doi.org/10.1016/S0042-6989\(97\)00169-7](https://doi.org/10.1016/S0042-6989(97)00169-7)
- [32] P. O Hoyer, A. Hyvärinen. A multi-layer sparse coding network learns contour coding from natural images. *Vision Research*. [https://doi.org/10.1016/S0042-6989\(02\)00017-2](https://doi.org/10.1016/S0042-6989(02)00017-2)
- [33] F. Giménez, J.A. Monsoriu, E. Alemany. Aplicación de la convolución de matrices al filtrado de imágenes. *Modelling in Science Education and Learning*. <https://doi.org/10.4995/msel.2016.4524>
- [34] Y.L. Boureau, J. Ponce, Y. Lecun. A theoretical analysis of feature pooling in visual recognition. *ICML*. 2010. http://yann.lecun.com/exdb/publis/pdf/boureau_icml-10.pdf
- [36] A. Zhang, Z.C. Lipton, M. Li, A.J. Smola. Dive into Deep Learning. 2019. <https://d2l.ai>
- [35] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *The MIT Press Journals*. 2008. <https://doi.org/10.1162/neco.1989.1.4.541>
- [37] A. Krizhevsky, I. Sutskever, G.E. Hinton. ImageNet classification with deep convolutional neural networks. *Comunications of the ACM*. 2012 <https://dl.acm.org/doi/10.1145/3065386>
- [38] K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv, Cornell University*. 2014. <https://arxiv.org/abs/1409.1556>
- [39] M. Lin, Q. Chen, S. Yan. Network In Network. *ArXiv, Cornell University*. 2014. <https://arxiv.org/abs/1312.4400>
- [40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed et al. Going deeper with convolutions. *ArXiv, Cornell University*. 2015. <https://arxiv.org/abs/1409.4842>
- [41] M. Radiuk. Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Information Technology and Management Science*. 2017. [doi: 10.1515/itms-2017-0003](https://doi.org/10.1515/itms-2017-0003)
- [42] V. Dumoulin, F. Visin. A guide to convolution arithmetic for deep Learning. *ArXiv, Cornell University*. 2018. <https://arxiv.org/abs/1603.07285>