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.

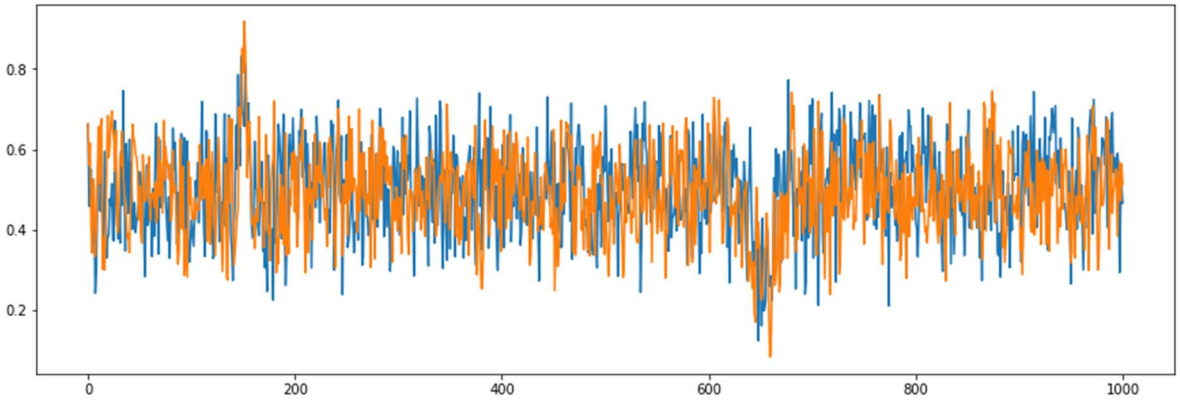

Figura 25: Datos de dos termografías reescaladas con el mismo defecto, pero diferente ruido.

- 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.

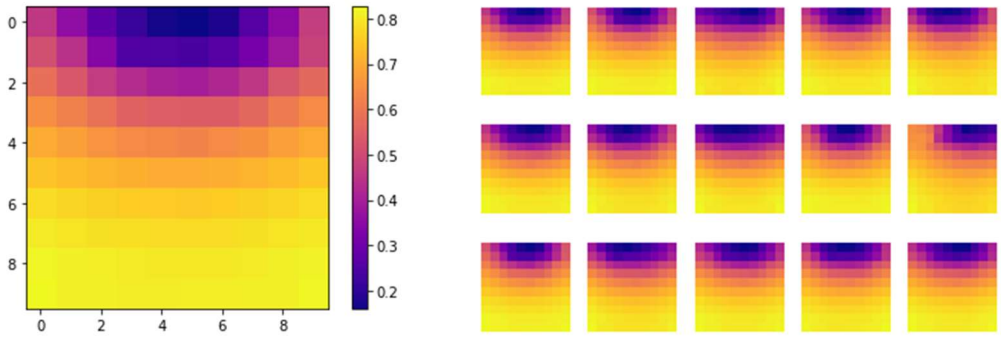

Figura 26: Datos reescalados de diferentes derivadas topológicas en la zona cercana al defecto.

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:

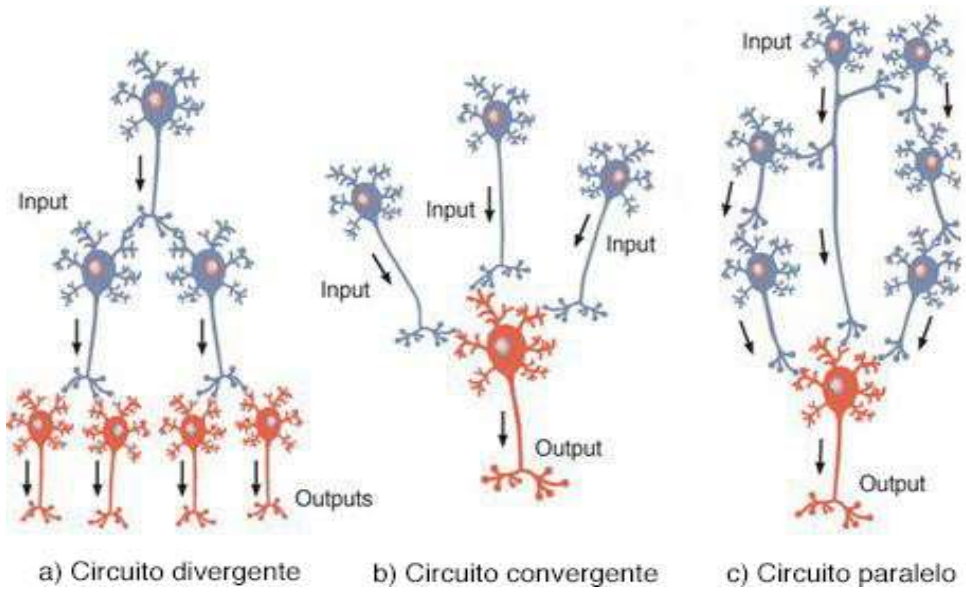

Figura 27: Estructuras neuronales biológicas.

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 (*ELU*)
2	1024	*ELU*
3	512	*ELU*
4	256	*ELU*
5	128	*ELU*
6	64	*ELU*
7	32	*ELU*
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 *overfiting* 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:

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

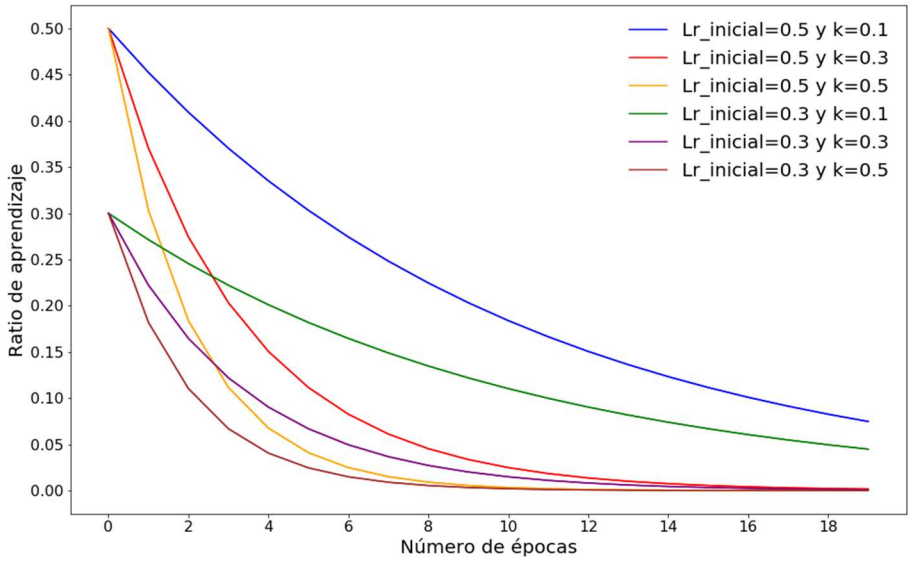

Figura 28: Diferentes modelos de ratios de aprendizaje exponencial decreciente.

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 *overfiting* 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.

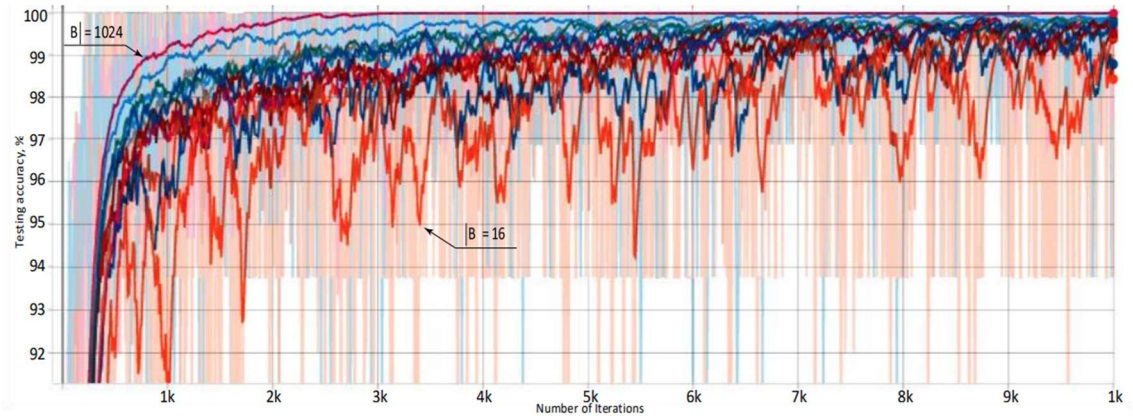

Figura 29: Ejemplo teórico de cómo una correcta agrupación de experimentos ayuda al entrenamiento. Fuente: [41]

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 *overfiting*, 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.

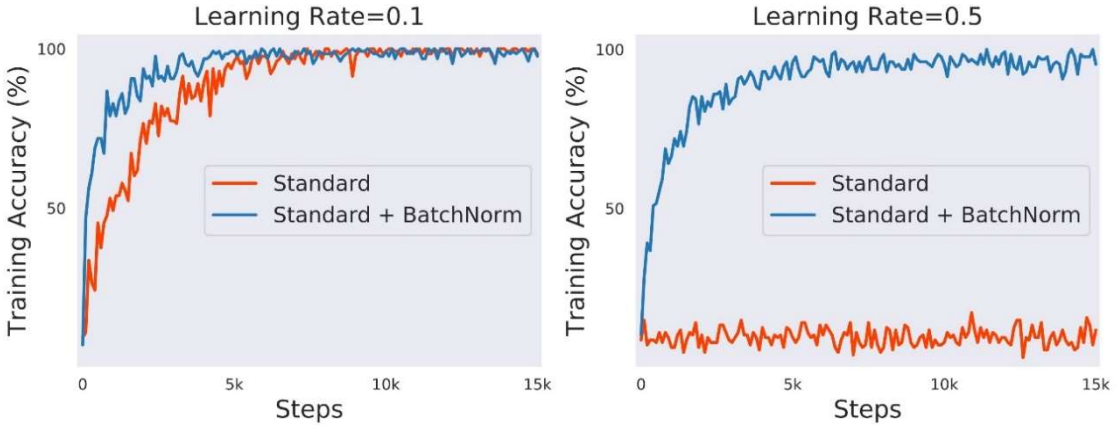

Figura 30: Ejemplo teórico del aporte del *Batch Normalization* al entrenamiento de redes neuronales. Fuente: [27]

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 *overfiting, underfiting* 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.

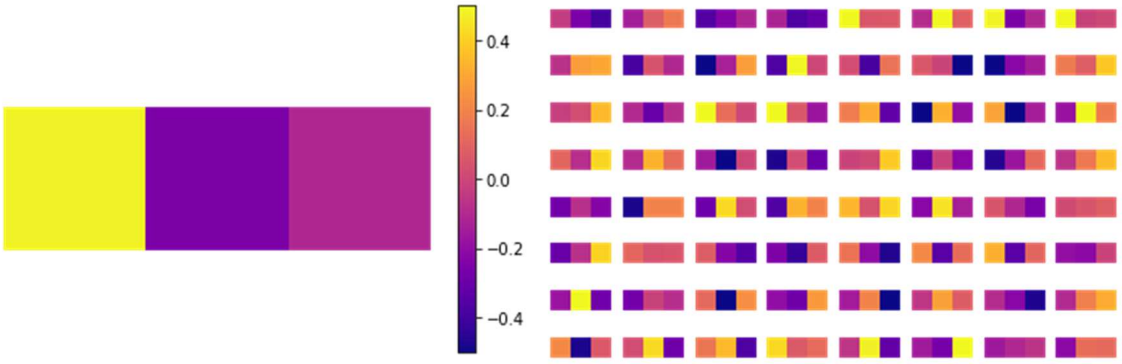

Figura 31: Filtros de la primera capa convolucional para la base de datos formada por termografías.

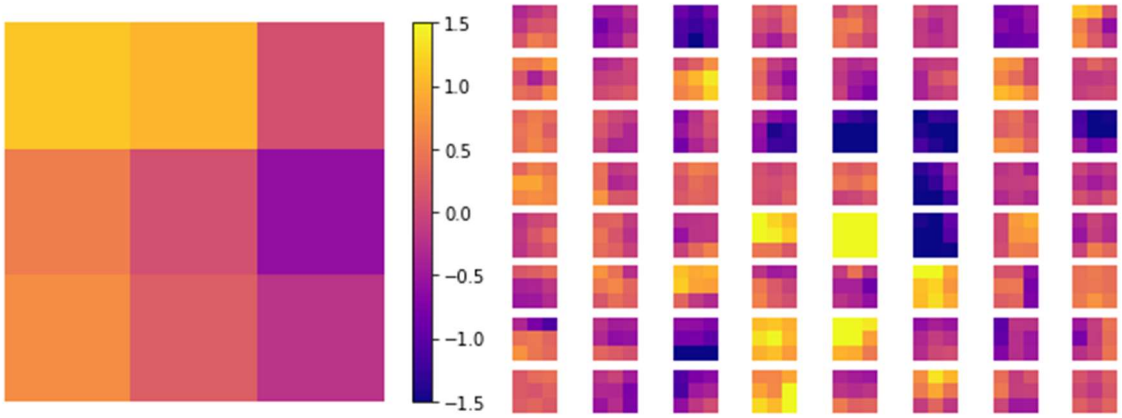

Figura 32: Filtros de la primera capa convolucional para la base de datos formada por las derivadas topológicas.

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.