UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Redes neuronales convolucionales con imágenes hiperespectrales para control de calidad alimentaria

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Calandín Vega, Andrés

Tutor/a: Paredes Palacios, Roberto

CURSO ACADÉMICO: 2021/2022

Resum

En l'últims anys s'ha consolidat una tendéncia cap a la digitalització y la semiautomatització o l'automatització. Aquest fet hi ha ocasionat l'implementació de múltiples tècniques d'aprenentatge automàtic o aprenentatge profund a camps de treball en els que en un inici no es va pensar que es pogueren aplicar. En aquest escrit es desenvolupa l'ús d'imatges hiperespectrals amb xarxes neuronals convolucionals amb el objectiu d'obtindre els paràmetres necesaris de la qualitat del producte destinat ser posat a la venda, toyina en lata. En aquest marc, l'ús del llenguaje de programació Python s'ha extés enormement per la comunitat científica y tècnica amb l'objectiu d'abocar mètods d'intel·ligència artificial fent gran èmfasis en l'ús de les ferramentes de Keras i Tensorflow en l'àmbit de l'anàlisis profund d'imatges, així com la creació de models capaços de realitzar accions y decisions en base a les imatges. Les imatges hiperespectrals proporcionen una resolució y rang espectral que les imatges convencionals RGB no logren, permetent medir determinat nombre d'amplades de banda per cada píxel en l'espectre de la llum. Aquest tipo d'imatges permet l'obtenció de valores que faciliten l'identificació del parametres, méss abstractes, de la textura del producte, podent emprar les xarxes convolucionales per obtindre una aproximació al paràmetre.

Paraules clau: Xarxa convolucional; Control de qualitat alimentaria; Imatge hiperespectral

Resumen

En los últimos años se ha consolidado una tendencia hacia la digitalización y la semiautomatización, o la automatización. Este hecho ha ocasionado la implementación de múltiples técnicas de aprendizaje automático o aprendizaje profundo a campos en los que en un principio no se pensó que se pudieran aplicar. En este escrito se trata el uso del análisis de imágenes hiperespectrales con redes neuronales convolucionales con el objetivo de obtener los parámetros necesarios de la calidad del producto destinado a la venta, atún en lata. En este marco, el uso del lenguaje de programación Python se ha extendido enormemente por la comunidad científica y técnica con el objetivo de emplear métodos de inteligencia artificial con gran énfasis en el uso de las herramientas de Keras y TensorFlow en el ámbito del análisis profundo de imágenes, así como la creación de modelos capaces de realizar acciones y decisiones en base a estas. Las imágenes hiperespectales proporcionan una resolución y rango espectral que las imágenes convencionales como las RGB no logran, permitiendo medir determinado número de anchos de banda para cada píxel en el espectro de la luz. Este tipo de imágenes permite la obtención de valores que facilitan la identificación de los parámetros, algo más abstractos, de la textura del producto, pudiendo usar las redes convolucionales para obtener una aproximación al parámetro.

Palabras clave: Redes convolucionales; Control de calidad alimentaria; Imagen hiperespectral

Abstract

In the last years a new tendency has been consolidated forward digitalization and automation in the industry. This fact has caused the implementation of multiple machine learning or deep learning techniques en fields of knowledge that in first sight are not viable to be applied. This paper is about the use of hyperpespectral imaging analysis with

convolutional neural networks with the purpouse of obtain the required parameters relatives to the quality of the product destined to be sold, canned tuna. In this context, the use of the programming language Python has now largely spread throughout the scientific and technique community with the purpouse of apply machine learning methods, with great emphasis on the tools Keras and Tensorflow in the field of the image analysis. Hyperespectral imaging provides better and wider espectral resolution than the conventional ones, with RGB, allowing measure various wavelenghts of the espectrum of light in one pixel. This type of images grant obtain easier values that make easier the measurement of the parameters, sometimes abstracts, of the texture of the product, using the convolutional networks to obtain this measures.

Key words: Convolutional neural network, Food quality control; Hyperspectral imaging

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En las dos últimas décadas, el empleo de las imágenes hiperespectrales se ha extendido en la industria alimenticia para analizar la calidad de los productos. En este aspecto, las redes convolucionales pueden ayudar a analizar estas imágenes, gracias a la gran capacidad de extraer las características y patrones principales que se encuentran escondidas en las imágenes. En el siguiente escrito se redacta la creación de una red neuronal convolucional que extraiga los parámetros requeridos sobre la textura del producto, para así poder automatizar la sección de control de calidad de la cadena de producción.

1.1 Motivación

El control de calidad en las cadenas de producción es uno de los puntos críticos de estas, en las que se decide si se descarta o no el producto de la fábrica. La automatización de este proceso es uno de los trabajos que más interés genera y el uso de las últimas técnicas de *Deep Learning* puede facilitar la tarea, a la vez que permite encontrar patrones y tendencias que el ojo y la mente humanas no alcanzan a distinguir. Debido al auge en el crecimiento de la población mundial, la industria alimenticia se ha visto necesitada de una velocidad y volumen de producción mayores que en los últimos años. Por lo que se ha decidido realizar este estudio, de cómo una red neuronal convolucional puede facilitar la automatización de la sección de control de calidad en la cadena de producción para acelerar el proceso y permitir mayor precisión en la calidad de los productos.

1.2 Objetivos

Dado lo escrito anteriormente, este trabajo tiene como fin el generar una red neuronal convolucional, ya sea única o por cada atributo requerido de la textura del producto, que obtenga de forma precisa el valor de cada atributo designado de la textura del mismo. De esta forma, se mejorará el control de calidad de la cadena de producción al poder determinar la medida de cada atributo de la textura, empleando como medida de comparación una muestra de control.

Como objetivo más específico, el vislumbrar si es más fácil el predecir el valor de unos atributos que de otros, es decir, identificar aquellos atributos que puedan ser descritos mejor por la red neuronal convolucional.

1.3 Impacto esperado

Se espera que la siguiente investigación, por medio de los experimentos descritos en el capitulo 4, permita avanzar camino en el campo del uso de redes neuronales convolucionales con el fin de analizar imágenes hiperespectales, esclarecer la facilidad con la que quedan descritos los atributos requeridos en los espectros analizados, y proporcionar herramientas que permitan a la empresa agilizar la cadena de producción y mejorar la calidad de sus productos.

1.4 Estructura de la memória

Primero se explicará a detalle el problema sobre el que se basa este proyecto en el capitulo 2. Acto seguido, se actualizará al lector de la evolución y uso de las redes neuronales, de las que se crean las redes neuronales convolucionales, empleadas en este proyecto, su uso y evolución. Y también se comentará el uso y evolución de las imágenes hiperespectrales, en el capitulo 3. En el capitulo 4 se explicará como se ha creado y procesado las imágenes para realizar los experimentos, explicando a detalle estos mismos y los resultados obtenidos. Por último, las conclusiones extraídas del proyecto expuestas en el capitulo 5. También se pueden ver los scripts usados en diferentes momentos del proyecto en los apéndices B y C.

CAPÍTULO 2
Descripción del problema

El control de calidad es una de las fases más importantes en la industria, esta fase determina si la muestra presenta los requisitos requeridos para su distribución y venta. En caso contrario, todo el lote en el que se encuentre la muestra analizada debe ser descartado. En una fábrica con varias líneas de producción en cadena el volumen de muestras y lotes a analizar es excesivo, sumado al poco tiempo que debe tomar para no suponer un retraso en la cadena, esta fase se ha convertido en uno de los objetivos de automatización más requeridos en los últimos tiempos.

Dado esto, actualmente la empresa utiliza un espectrómetro con el fin de obtener valores de la textura de la muestra mediante espectroscopia infrarroja. Este espectrómetro obtiene imágenes hiperespectrales de las muestras, que se deben analizar para poder obtener dichos valores de la textura, que son: fuerza, cohesión, adhesión, elasticidad, gomosidad y masticabilidad. Por tanto, se necesita una herremienta que pueda obtener los valores de estas seis características o atributos de la textura del producto, y dado que la información se encuentra en formato de imágenes hiperespectrales, no sirve cualquier herramienta, por lo que se debe usar una que pueda obtener información de las imágenes. Esta herramienta son las redes neuronales convolucionales, que extraen patrones y características de las imágenes para poder clasificar o realizar una regresión sobre estas. Por tanto, se necesita crear una red neuronal convolucional que permita obtener los valores requeridos de la imagen hiperespectral de la muestra.

CAPÍTULO 3
Metodología

3.1 Redes neuronales

En si, las redes neuronales surgieron como modelos del funcionamiento de las neuronas que se encuentran presentes en los cuerpos de los seres vivos, haciendo uso de circuitos conectados entre si con el fin de semejar un comportamiento inteligente mediante un circuito eléctrico simple, en 1943, por el neuropsicólogo Warren McCulloch y el matemático Walter Pitts. A raíz de esto, Donald Hebb desarrolló esta idea en su libro *The Organization of Behaviour* (1949). En su libro, propuso que los enlaces o caminos creados por la red neuronal se refuerzan con cada uso sucesivo, sobre todo con las neuronas que tendían a activarse a la vez. De esta forma comenzó la investigación en redes neuronales de cara a cuantificar procesos complejos de nuestro cerebro, que ha derivado en el contexto actual.

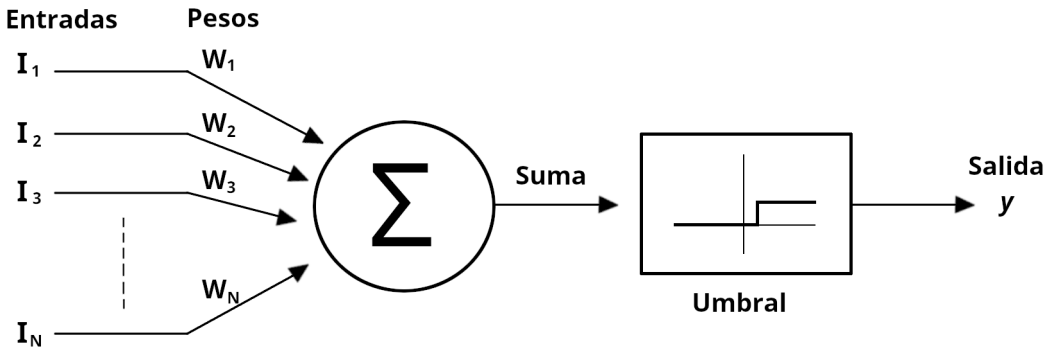


Figura 3.1: esquema de una neurona de McCulloch-Pitts. Fuente: autor.

En esta época surgieron los dos precursores principales de las redes neuronales actuales: el *Threshold Logic* o Lógica de Umbral, que se basa en la conversión de una señal continua a una discreta; y el *Hebbian Learning* o Aprendizaje Hebbiano, propuesto por Donald Hebb y que consiste en un modelo de aprendizaje basado en la plasticidad neuronal [1]. Ambos modelos fueron propuestos en la década de 1940, y no fue hasta la década de 1950 cuando los investigadores comenzaron a trabajar estas redes neuronales en los sistemas computacionales de la época, siendo la primera red Hebbiana implementada con éxito en el MIT el año 1954. Paralelamente, el psicólogo Frank Rosenblatt trataba de comprender las decisiones simples que se encontraban presentes en el ojo de una mosca. En uno de los intentos para comprender y cuantificar este proceso propuso la idea del

Perceptron, en 1958. Este sistema se componía de un simple proceso de entrada-salida modelado en una neurona de McCulloh-Pitts, propuesta anteriormente por los citados investigadores Warren McCulloch y Walter Pitts en 1943, que hace uso de un umbral lineal. Esta neurona recibe una serie de entradas, realiza una suma ponderada sobre ellas y devuelve 0 o 1 dependiendo de si el valor de la suma se encuentra por debajo o encima de un umbral determinado (figura 3.1). En la actualidad, las redes neuronales se basan en este perceptron, siendo que estas son una configuración de capas superpuestas de estos perceptrones. Ya en la década de los años 60, en Stanford, Bernard Widrow y Marcian Hoff consiguieron crear la primera red neuronal aplicada a un problema común en la vida real. Esta red neuronal se diferenciaba de los perceptrones en el hecho de que la salida devuelta por la red no era un valor discreto, sino la suma ponderada.

Toda la investigación y avance realizados llegó a un estancamiento en la década de los años 70 de la mano del fundador del MIT AI Lab y su director, Marvin Minsky y Seymour Papert. Su libro *Perceptrons* argumentaba que la aproximación de Rosenblatt no podía construir redes neuronales multicapa eficientes, dado que la evaluación y selección de los valores de los pesos de las neuronas a través de cada capa de la red, basadas en la salida final, requeriría una cantidad muy elevada de iteraciones, siendo el coste temporal inasumible. Mediante este problema y otros más escritos en el texto sobre las redes neuronales, Minsky convenció a la comunidad científica de que la investigación de redes neuronales basadas en perceptrones no proporcionaría solución alguna. Esto causó una extensa sequía de investigaciones en los siguientes 10 años, en los que ninguna institución aceptara patrocinar proyecto alguna que estuviera relacionado con las redes neuronales. Y de esta forma, este periodo de la investigación de las redes neuronales se ha llegado a denominar como *Invierno de la IA*.

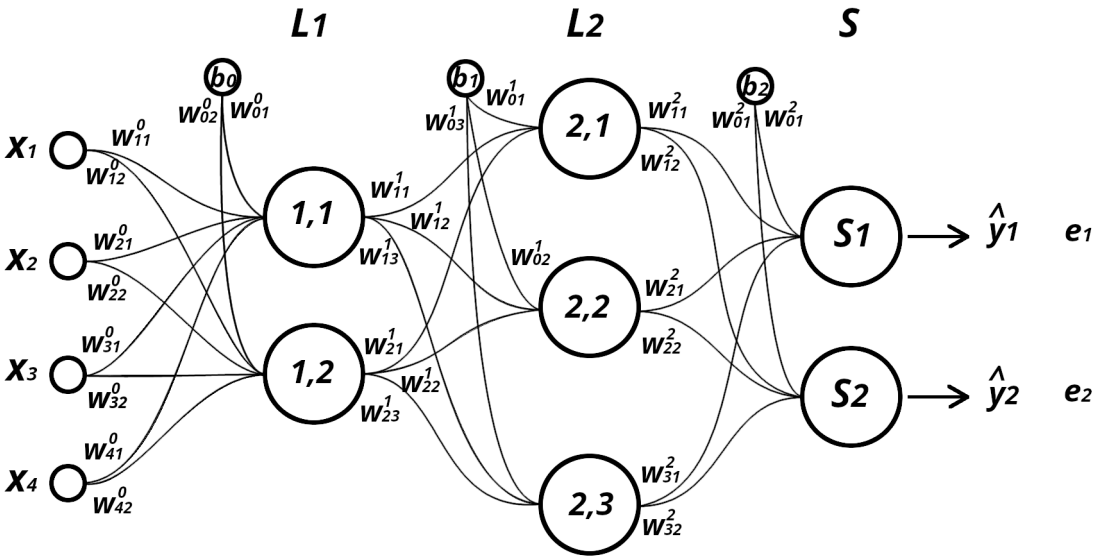


Figura 3.2: esquema de una red neuronal con dos capas ocultas. Fuente: autor.

No fue hasta el año 1982 que Jon Hopfield publicó un articulo con una aproximación a la cual se ha llamado *Hopfield Net* [2] [3]. Este tipo de red neuronal se ha usado en imágenes para realizar funciones de restauración [4]. Otro uso encontrado ha sido en criptografía para generar llaves seguras [5]. Sin embargo, no fue este hallazgo lo que reimpulsó la investigación en redes neuronales, lo que lo ocasionó fue el redescubrimiento de un concepto que ya se había creado en la decada de los años 60, y que se había desarrollado aun más en el *Invierno de la IA*, propagación hacia atrás o *backpropagation*. El primero en

redescubrirlo y asociarlo a los métodos de aprendizaje profundo fue Paul Werbos, quién escribió sobre este concepto en su tésis. Igualamente, este no fue tomado en consideración hasta que Rumelhart, Hinton y Williams publicaran un articulo en el que se detallaba claramente el funcionamiento y sus implicaciones para las redes neuronales. *Backpropagation*, junto a *Descenso por Gradiente*, conforman los fundamentos y mayores potencias de las redes neuronales.

El descenso por gradiente permite actualizar los errores y pesos hacia el punto de mínimo coste de la función elegida. Se basa en reducir la función de coste θ al mínimo mediante el uso de derivadas parciales de esta función. Para realizar el algoritmo de descenso por gradiente se debe inicializar el proceso con un punto aleatorio en la función, siendo para $f(\theta)$ el punto aleatorio inicial $\theta_0 = (x_0, y_0)$. Una vez se tiene un punto inicial, se calcula el gradiente de la función en ese punto según la formula 3.1. Este gradiente usará en la formula 3.2 para actualizar el valor de θ, el punto de la función. El valor α se refiere al *learning rate* o ratio de aprendizaje, que modifica como de grande será el paso entre puntos, cuanto mayor sea α mayor distancia habrá entre el punto obtenido y el anterior. Encontrar el *learning rate* adecuado determinará cuantas iteraciones se necesitarán para llegar al mínimo global de la función y su precisión, dado que si el α es bajo se necesitarán muchos pasos para llegar a un mínimo, y no solo eso, existe el riesgo de quedar atrapado en un mínimo local por tener un paso demasiado pequeño. En cambio, si el paso es demasiado grande, dificultará el acertar en el mínimo, sea local o global. Existen diversos métodos que varían dinámicamente este *learning rate*, como el Descenso por gradiente con momentum, RMSProp o Adam.

$$\nabla f(\theta) = \begin{pmatrix} \dfrac{\partial f(\theta)}{\partial x} \\ \dfrac{\partial f(\theta)}{\partial y} \end{pmatrix} \tag{3.1}$$

$$\theta_{t+1} = \theta_t - \alpha \nabla f(\theta_t) \tag{3.2}$$

Sabiendo esto, la idea del concepto de *Backpropagation* es más fácil de entender. Como antes se ha explicado, las redes neuronales están compuestas de neuronas del tipo perceptron, estas neuronas reciben una serie de entradas con pesos y realizan una suma ponderada que se discretiza mediante un umbral o función discriminante lineal con activación (FDLA), siendo que si el valor obtenido por la suma es mayor que el umbral se proporciona un 1, si no, 0. La superposición de estas neuronas en capas, conectadas a otras crea una red de perceptrones multicapa o red neuronal moderna (figura 3.2). Estas redes presentan las entradas, los datos que se quieren usar para obtener las salidas o predicciones, clasificaciones, dependiendo del problema a resolver; y los pesos (w) para la suma ponderada en cada conexión entre neuronas, como se puede ver en la figura 3.2. Estos pesos se inicializan aleatoriamente y se obtiene una salida, la cual se emplea para calcular el error mediante la función de error, o coste, que será elevado. Por tanto, en la última capa, para calcular el error para una de las salidas, sería la formula:

$$C(a(Z^L)) = Error \tag{3.3}$$

siendo C la función de coste, a la función de activación o umbral y Z^L la suma ponderada dentro de la neurona de la capa L. Siendo Z la fórmula 3.4, en la que w^L son los pesos de las conexiones entrantes a esa neurona, a^{L-1} las entradas de esa neurona (y salidas de la anterior) que se multiplican a sus respectivos pesos w y b^L el parámetros de umbral o

bias. Por tanto, viendo lo anterior, se entiende que en la derivada de la función de coste influyen los parámetros w^L y b, y se deben obtener las respectivas derivadas parciales 3.5.

$$Z^L = w^L \cdot a^{L-1} + b^L \tag{3.4}$$

$$\begin{aligned} \frac{\partial C}{\partial w^L} &= \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial w^L} \\ \frac{\partial C}{\partial b^L} &= \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial b^L} \end{aligned} \tag{3.5}$$

Si se toma la multiplicación de las dos primeras derivadas parciales, $\frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L}$, se tiene que es igual que $\frac{\partial C}{\partial z^L}$, que indica en que medida se ve modificado el error o coste cuando se produce un cambio en la suma ponderada de la neurona, si la derivada es grande implica que con el menor cambio en el valor de la neurona se generará un gran cambio en el resultado final, permitiendo distinguir aquellas neuronas que tengan un mayor peso en la red. Es decir, esta derivada representa el error que se imputa a la neurona, soliendo representarse como δ^L. De tal forma, esto permite simplificar las dos ecuaciones anteriores.

$$\begin{aligned} \frac{\partial C}{\partial w^L} &= \delta^L \cdot \frac{\partial z^L}{\partial w^L} \\ \frac{\partial C}{\partial b^L} &= \delta^L \cdot \frac{\partial z^L}{\partial b^L} \end{aligned} \tag{3.6}$$

Si se observa la suma ponderada Z^L, se puede apreciar que $\frac{\partial z^L}{\partial w^L}$ sería igual a 1, dado que b no se ve afectado por ningún coeficiente. Por tanto la derivada del coste en función de b^L sería el propio error de la neurona. En cuanto a la derivada del coste en función de los pesos esta sería la activación de la capa previa. Quedando las derivadas parciales a obtener lo siguiente.

$$\begin{aligned} \frac{\partial C}{\partial w^L} &= \delta^L \cdot a^{L-1} \\ \frac{\partial C}{\partial b^L} &= \delta^L \end{aligned} \tag{3.7}$$

Una vez se tienen todas las formulas anteriores se puede realizar la actualización de los pesos y la retropropagación del error (*backpropagation*):

$$\begin{aligned} \nabla C(w^L) = \frac{\partial C}{\partial w^L} = \delta^L \cdot a^{L-1} \rightarrow w^{L\prime} = w^L + \alpha \cdot \nabla C(w^L) \\ b^{L\prime} = b^L + \alpha \cdot \delta^L \end{aligned} \tag{3.8}$$

De esta forma se siguen los siguientes pasos hasta llegar a la capa de entrada:

1. Computo del error de la última capa.

$$\delta^L = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial z^L}$$

2. Retropropagar el error a la capa anterior.

$$\delta^{L-1} = W^L \delta^L \cdot \frac{\partial a^{L-1}}{\partial z^{L-1}}$$

3. Calcular las derivadas de la capa usando el error.

$$\frac{\partial C}{\partial w^{L-1}} = \delta^{L-1} \cdot a^{L-2}, \frac{\partial C}{\partial b^{L-1}} = \delta^{L-1}$$

3.2 Redes neuronales convolucionales

Una vez se han visto las redes neuronales, la base para hablar sobre las redes neuronales convolunionales está ya construida. Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son la arquitectura más usada de *Deep Learning* en el campo del procesamiento de imágenes, y esto es por la facilidad con la que pueden extraer y descrifrar patrones y características muy complejos de *datasets* con inmensas cantidades de imágenes. Aportando, así, una gran parte del funcionamiento de la llamada *visión computacional* o *por ordenador*, del inglés, *computer vision*, la cuál, se ha extendido por varios y diversos campos de la ciencia y la tecnología como la seguridad, la medicina o la robótica; usándose para identificar el rostro (y sus propiedad) de un individuo, identificar anomalías en una imagen médica, como puede ser una radiografía, o identificar que un peatón invade la trayectoria de un coche autónomo, frenando así para evitar un accidente. Y de esta forma, la *computer vision*, permite a las máquinas percibir y obtener información del mundo que las rodea.

Como se puede intuir por el nombre, las redes convolucionales son una evolución o escisión de las redes neuronales actuales; y como tal, presentan parecidos entre ambas. Estos parecidos consisten en que ambos tipos de redes emplean una entrada o serie de entradas para obtener una salida siguiendo un complejo conjunto de conexiones ponderadas, y modificando los pesos de esas ponderaciones para que la salida de la red coincida lo mejor posible con la salida real, es decir, son métodos llamados *supervisados*. Las diferencias comienzan cuando identificamos estas entradas, salidas y pesos. Mientras que en las redes neuronales, como se ha comentado anteriormente, las entradas de las redes neuronales son números, los pesos son valores asociados a las conexiones entre neuronas para indicar su importancia dentro de la red y las salidas son, también números. En cambio, las redes convolucionales reciben como entradas imágenes, que son matrices tridimensionales o bidimensionales, sus pesos son los valores en los *kernels* de las convoluciones (que se explicarán más adelante) y las salidas son números. En esencia, las redes convolucionales son redes neuronales específicamente construidas para tratar con imágenes. La razón, viene a ser por una falla en el diseño de las redes neuronales para poder identificar patrones en imágenes. Esto es por qué, para analizar una imagen por una red neuronal, estas son aplanadas, es decir, se convierte en un véctor con los valores de los píxeles, siendo cada entrada un píxel. Esto genera varios problemas, entre los que se encuentra que, por cómo la red lo analiza, no se toma en cuenta la posición de un píxel respecto a los demás, pudiéndose proporcionar dos imágenes totalmente distintas y obtener el mismo resultado.

Para solucionar este problema se idearon las redes CNNs, que emulan el proceso de la visión humana. Los humanos basamos nuestra visión en la identificación de patrones,

formas y colores; proceso que las redes neuronales no pueden replicar fielmente, y que las CNN sí son capaces de realizar. Es decir, las redes neuronales convolucionales son capaces de identificar la estructura espacial de una imagen. Para ejemplificar este proceso, pongamos el ejemplo de un rostro. Los humanos somos capaces de identificar un rostro por la distinción de elementos más pequeños que combinados lo constituyen, un patrón, entre ellos: una boca, una nariz, dos orejas, dos cejas y dos ojos. Luego, estos elementos están constituidos por otros que, de igual forma, representan un patrón que hemos identificado como *ojo*: pupila negra rodeada de un iris de color, líneas que conforman las pestañas, superficies blancas que son el grueso del ojo. Todos estos elementos han sido identificados gracias a que nuestro cerebro es capaz de identificar patrones, cambios de contraste y texturas. Al final, replicando los pasos que realiza nuestro cerebro en la función visual, nos encontramos con un procesamiento en cascada en el que inicialmente se identifican patrones generales y básicos, y luego se identifican elementos más complejos, combinación de los anteriores.

Las CNN, para realizar todas las operaciones que se han explicado, emplean en las capas de la red una operación matemática llamada *convolución*. Una convolución se define como la transformación de dos funciones f y g distintas a una tercera que describe la magnitud con la que se superpone la función f sobre g mientras una se traslada. Es decir, una convolución se puede definir como una media móvil. En las redes convolucionales se emplea la convolución en 2D, cuya fórmula es la siguiente:

$$(f \otimes g)[n,m] = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} f[n-j][m-k]g[j][k] \tag{3.9}$$

$$(f \otimes g)[n,m] = \sum_{j=0}^{2} \sum_{k=0}^{2} f[n-j][m-k]g[j][k] \tag{3.10}$$

$$(f \otimes g)[n,m] = \sum_{j=-1}^{1} \sum_{k=-1}^{1} f[n-j][m-k]g[j][k] \tag{3.11}$$

En la que f y g son dos imágenes, siendo f la imagen de entrada y g una imagen pequeña que usualmente se denomina *kernel*, que actúa como filtro para la imagen de entrada. Por tanto, los límites de los sumatorios se definen como el tamaño de g 3.10, pudiendo ser también representadas las variables n,m como el centro del filtro, en el caso de que sea impar 3.11.

Este filtro genera una nueva imagen de menor tamaño que la anterior, a razón de $k\prime = k - (\lfloor n/2 \rfloor \cdot 2)$ y $j\prime = j - (\lfloor n/2 \rfloor \cdot 2)$, siendo j, k las dimensiones de la imagen original, $j\prime, k\prime$ la imagen modificada y n, m las dimensiones del filtro. Este filtro convolucional permite generar efectos que se usan hoy en día en muchas de las aplicaciones de edición o manipulación de imágenes, así como en los filtros presentes en muchas redes sociales. Dependiendo de los valores que haya en el *kernel* la imagen modificada presentará un aspecto u otro. Un kernel 3x3 que tenga todos los valores a 1 devolverá la imágen sin modificar, en cambio, si se disponen los valores como se ve en la imagen 3.3 se pueden identificar tipos de patrones como los horizontales, verticales y circulares. Además a diferenciar entre valores, no solo identificando con 0 o 1, sino con -1 se puede identificar cambios en el contraste de la imagen.

Por tanto, los valores que presente el filtro convolucional se veran reflejados en lo que se obtenga como resultado, siendo en el caso de las redes convolucionales, estos valores los pesos a que la red aprenderá y actualizará con cada iteración, dado que el principal

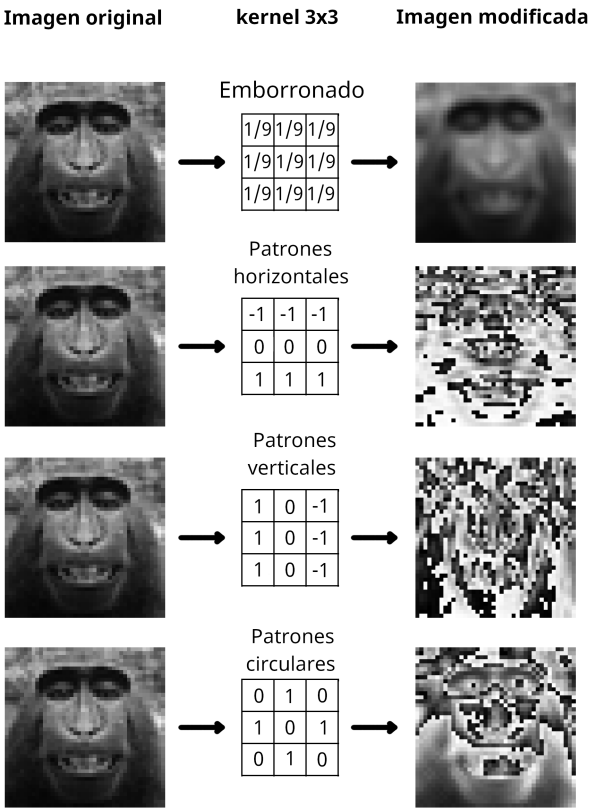


Figura 3.3: filtros convolucionales y su resultado. Fuente: autor.

papel de la red es identificar estos patrones y almacenarlos en estos resultados del filtro. Los resultados de los filtro convolucionales, en estas redes, son llamados mapas de características, o *feature maps*, ya que estos indican en qué ubicación de la imagen se ha detectado la característica que busca el filtro, codificándose como un píxel más claro o blanco. En cada capa de la red se realizan N filtros convolucionales, convirtiéndose estos mapas de características en la entrada de la siguiente capa de la red. Esta nueva capa efectuará los filtros sobre la imagen constituida por estos mapas de características. En esta mecánica de sucesión de capas encontramos la potencia de las redes convolucionales, si en la primera capa se comprimía la información de una región de nueve píxeles en uno, con cada capa la región se ensancha alcanzando más y más píxeles. Esta mecánica se puede potenciar al, de vez en cuando, reducir la resolución de la imágen mediante un submuestreo o *pooling*. Por si solo, un filtro convolucional no es tan potente como para detectar patrones más complejos que patrones y otros básicos, pero realizar detecciones sobre las detecciones obtenidas permite realizar patrones mucho más complejos. Por esta razón, las redes neuronales se representan con una forma de embudo, estas realizan fitros y submuestreos que reducen la resolución de la imágen, pero extraen mapas de características de la misma, aumentando la profundidad. Una vez los bloques convolucionales se han realizado, se tendrá un imagen con una resolución mucho menor que la original, compuesta por tantos mapas de características como filtros tenga el último bloque convolucional. Esta salida es procesada por una red neuronal convencional que obtiene el valor que se ha pedido a la red, ya sea una clase (0 o 1) o una regresión.

En si las redes neuronales convolucionales son una arquitectura de las redes neuronales que extrae mediante operaciones complejas consecutivamente ejecutadas unas características y patrones muy complejos, similares a los que el ser humano identifica

mediante la visión, para emplearlos en una red neuronal convencional que identificará aquellos mapas de características que describan mejor la imagen original.

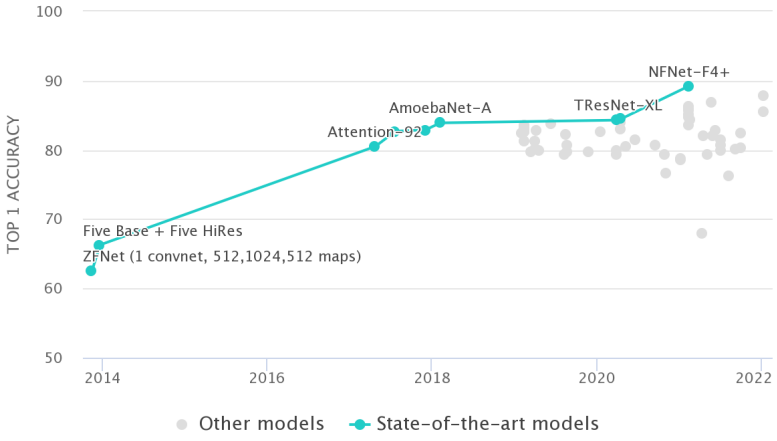


Figura 3.4: evolución de los mejores modelos de clasificación para ImageNet (CNN)

Las redes convolucionales se utilizan sobre todo en la clasificación de imágenes, y como se ve en la figura 3.4, estos últimos años se han producido avances significativos llegando casi al 90 % de precisión en las clasificaciones. Aun así, las redes convolucionales pueden usarse también para detección y segmentación en imágenes. La clasificación consiste en discriminar una imagen entre dos clases, el ejemplo por excelencia es la discriminación entre imágenes de gatos y de perros, haciendo que la CNN indique que imágenes con de perros y cuales de gatos. Por otra parte, la detección es otro tipo de clasificación, pero localizada, en la que la red señala una región en la que ha determinado que se encuentra el objeto deseado. Por último, la segmentación se basa en diferenciar diferentes objetos de la imagen, destacándolos de forma diferente para diferenciarlos.

Ya en la década de 1990 y en la de los 2000 se usaban CNN en investigación, pero no fue hasta el año 2012 que estas obtuvieron una gran popularidad, que continúa hasta hoy en día, gracias una arquitectura llamada AlexNet, como se ve en la figura 3.5, que obtuvo el mejor rendimiento en la clasificación de imágenes en el desafío de ImageNet [6].

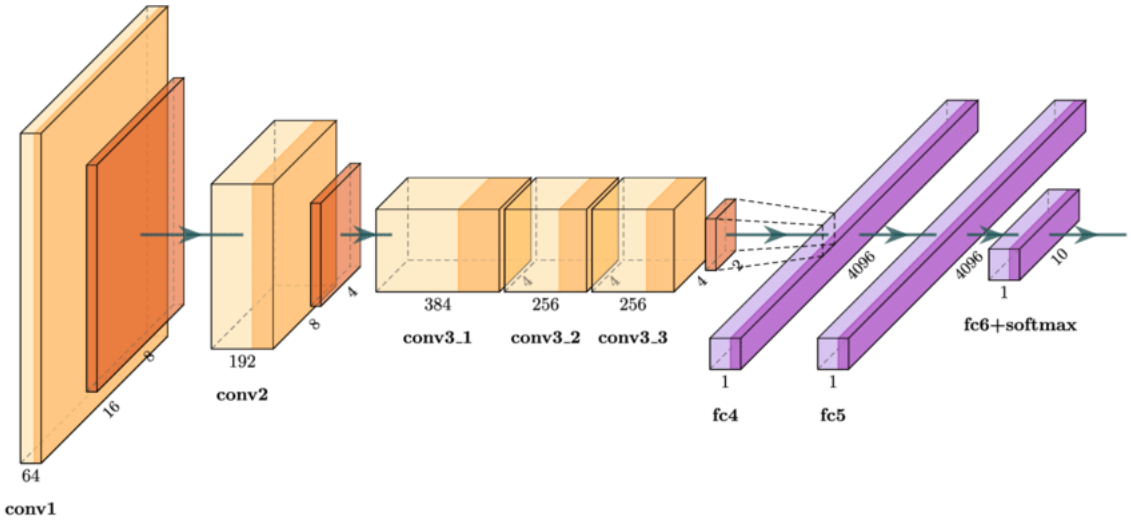


Figura 3.5: arquitectura AlexNet

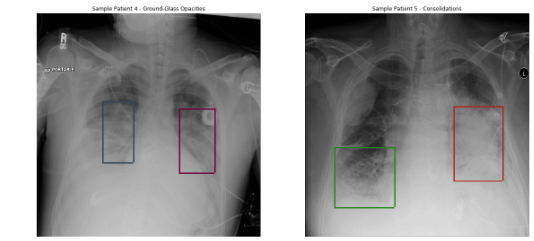
Desde entonces, el uso de las CNN se ha extendido por todos los sectores (figura 3.6), con gran incidencia en la automatización de tareas que se suelen realizarse con el

ojo o con pruebas de laboratorio, ya sea en el campo de la agricultura [7], la medicina [8], imágenes de satélite para detección de objetos [9] o de zonas catastróficas, como en el caso de zonas asoladas por terremotos [10]. Una de las formas en las que el publico general usa redes neuronales convolucionales en su día a día es el uso de la inteligencia artificial en los coches automáticos, como los de Tesla, que hacen uso de varias capas de *Deep Learning* con el fin de identificar distancias, objetos y posibles peligros en carretera [11]. Otros usos de las redes convolucionales son la reconstrucción en imágenes [12], que permite reconstruir una imagen dañada o modificada; el coloreado de fotos en blanco y negro, muy usado para dar color a imágenes o metrajes antiguos [13].

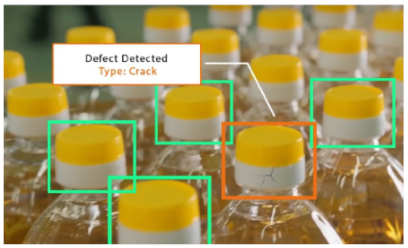
En cuanto al uso de las redes convolucionales con imágenes hiperespectrales, su uso es sobre todo con imágenes hiperespectrales satelitales, para identificación de elementos terrestres o del terreno, sean campos, bosques, o edificios [14] [15] [16]. La figura 3.7 muestra una clasificación de terreno basado en imágenes hiperespectrales analizadas por una CNN. Otros estudios con imágenes hiperespectrales aportan al campo de las CNN con el objetivo de crear métodos y arquitecturas capaces de obtener salidas de alta resolución de las imágenes introducidas [17], también existen estudios que usan las CNN con regresión para cuantificar, ya sea en el ámbito natural o en laboratorio [18].

En cuanto al análisis de imágenes hiperespectrales con redes convolucionales en la industria alimenticia, su uso siempre es en el campo del control de calidad, ya sea para que el producto cumpla unos parámetros designados o en busca de posibles objetos extraños [19] [20] [21]

En base a todo lo comentado anteriormente, las redes neuronales convolucionales llevan decadas siendo usadas en investigación para diferentes fines en laboratorios de todo el mundo, ya sea para mejorar modelos de inteligencia artificial que presenten funciones parecidas a las del ojo humano, o para generar herramientas que puedan ayudar en la vida cotidiana de la sociedad. Estas CNN nornalmente han sido usadas con imágenes convencionales RGB, pero en los últimos años han comenzado a usarse para captar patrones y características ocultas en las imágenes hiperespectrales, que suelen provenir de imágenes proporcionadas por satélites artificiales. En todos los casos anteriores, el objetivo de usar CNN es el de clasificar, discriminar o detectar elementos en la imagen, pero pocas veces se han usado como método de regresión para obtener valores cuantificables de interes de las muestras.

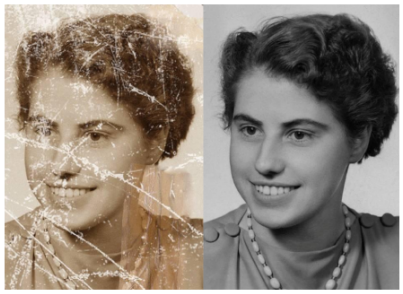


Detección en imagen médica

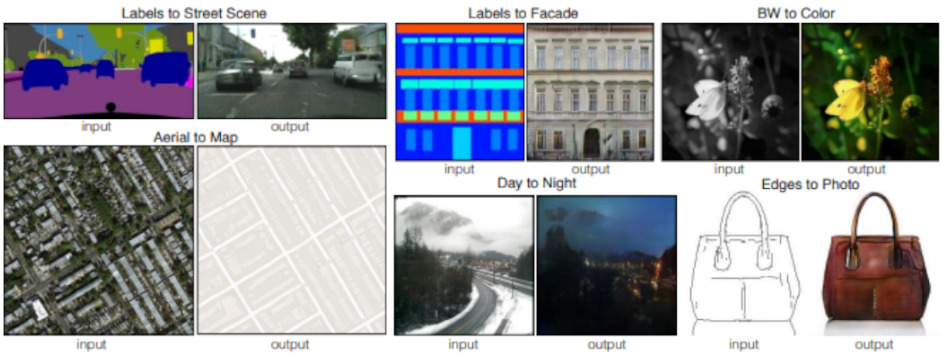


Control en producción en cadena

Coloreado de imágenes



Reconstrucción



Traducción

Figura 3.6: ejemplos de uso de CNN en la industria.

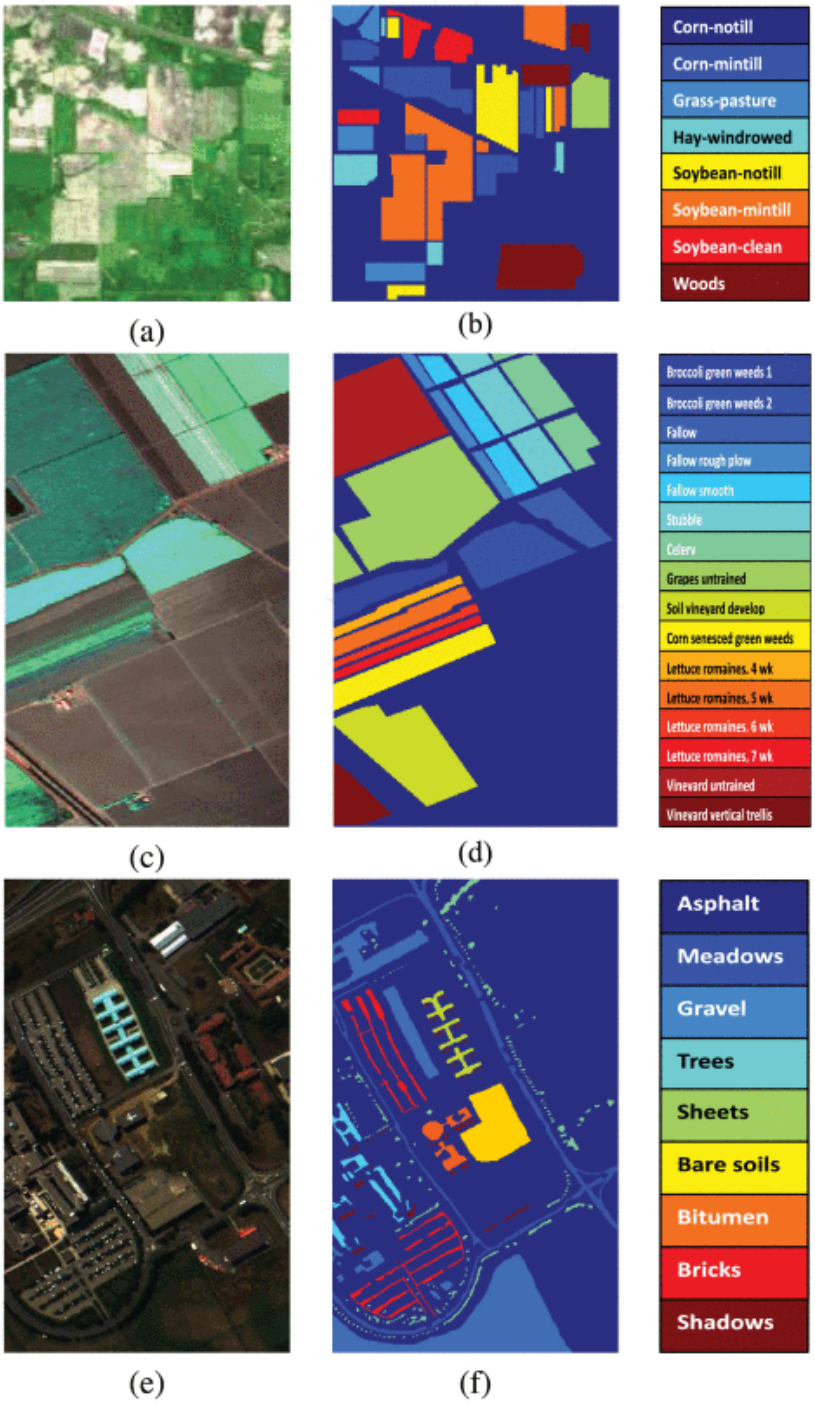


Figura 3.7: discriminación en imágenes hiperespectrales realizada por una CNN.

3.3 Imágenes hiperespectrales

En cuanto a las imágenes hiperespectrales, se usan en diferentes áreas e industrias como la agricultura, reciclaje, medicina, industria alimentaria, minerología, vigilancia o astronomía. Este tipo de imágenes se diferencia de las imágenes convencionales RGB en el hecho de que estas últimas recogen solo tres tipos de longitudes de onda, las visibles al ojo humano: longitud largas (color rojo), longitud media (color verde) y longitud corta (color azul). Sin embargo, las imágenes hiperespectrales recogen bandas que no pertenecen al espectro de la luz visible.

Existen cuatro formas de adquisición de las imágenes hiperespectrales: el escaneo espacial, el espectral, sin escaneo y el escaneo espacio-espectral, ejemplificadas en la figura 3.8. Que involucran tres tipos de arquitecturas de obtención de los espectros para conformar el cubo de datos (*datacube*): escaneo de barrido a detalle (*whiskbroom scanning*), configuración de imágen estática (*staring imager configuration*) y escaneo de barrido general (*pushbroom scanner*). El tipo *whiskbroom* obtiene píxel por píxel todos los espectros a analizar, es decir, por cada posición i,j ($i \in x$, $j \in y$) del cubo de datos de la figura anterior, el espectroscopio recoge toda la columna de espectros λ. El tipo *pushbroom*, al igual que el anterior, obtiene todos los espectros de un solo escaneo, pero en vez de obtenerlos de píxel en píxel, este obtiene toda una fila de la imagen. Por último, el tipo *staring imager configuration* se distingue por obtener de un solo escaneo una capa del espectro, es decir, una imagen bidimensional de un solo espectro, obteniendo las capas de longitudes de onda una por una.

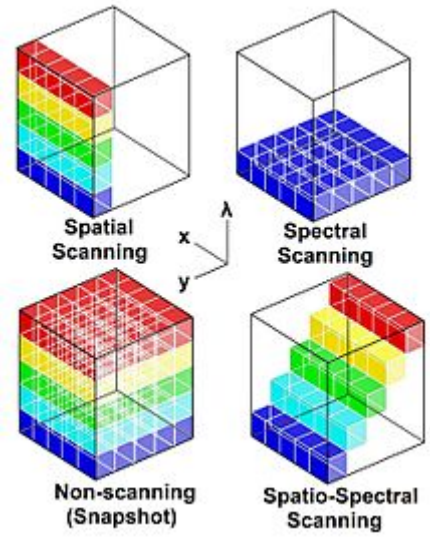


Figura 3.8: técnicas de adquisición de imágenes hiperespectrales, visualizadas como secciones de un *datacube* con dos dimesiones espaciales (x,y) y una espectral (lambda). Fuente: Wikipedia.

Comentado lo anterior, en cuanto a las formas de adquisición de la imagen hiperespectral, la primera, el *escaneo espacial*, involucra el tipo de escaneo *pushbroom*, que como se ha definido antes, obtiene una fila de la imágen por escaneo. El *escaneo espacial* implica que la muestra a analizar (o bien el sensor) se muevan en una dirección, y así con este movimiento, el escáner completa el cubo de datos fila a fila de la imagen. El tipo de *escaneo espectral* hace uso de la arquitectura de escáner *staring imager configuration*, realiza una serie de escaneos consecutivos con filtros ópticos y para cada escaneo cambia un el filtro, enfocándose en una longitud de onda específica, y así completando el cubo de datos. El tipo *no escaneo* no hace uso de alguna de las anteriores arquitecturas de escaneo, en este

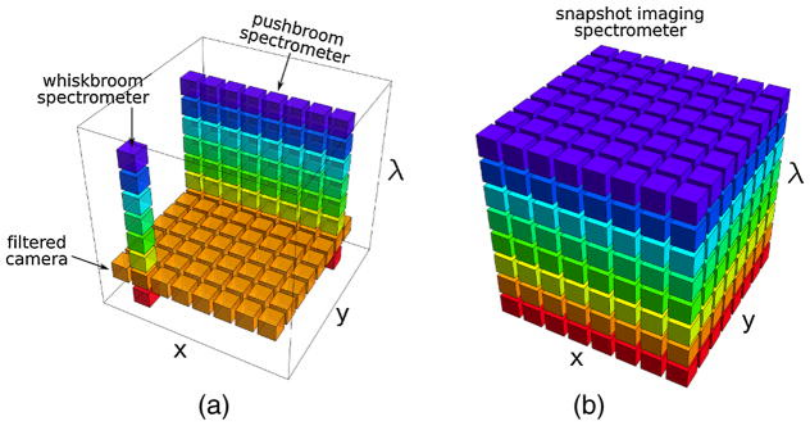


Figura 3.9: técnicas de escaneo en imágenes hiperespectrales en las que se han usado tecnicas de escaneo (a) y de instantánea (b).

caso, el cubo de datos es formado en un único escaneo, de esta forma se reduce el tiempo de adquisición.

Otra forma de distinguir o clasificar las imágenes hiperespectrales es la diferenciación entre imágenes hiperespectrales e imágenes multiespectrales. Mientras las imágenes hiperespectrales usan un rango continuo de longitudes de onda (ej.: 800-2500 nm con saltos de 1 nm) para formar el cubo de datos, las imágenes multiespectrales son un tipo de las anteriores que hacen uso de un selección de longitudes de onda espaciadas uniformamente (ej.: 800-2500 nm en saltos de 2 nm). Por tanto, las imágenes multiespectrales no muestran el espectro de un objeto y la única diferencia entre este tipo de imágenes y la fotografía a color es el hecho de que las imágenes multiespectrales abarcan no solo el espectro visible de la luz (6500 cm^{-1} - 14386 cm^{-1} o 350 nm - 700 nm).

Las imágenes hiperespectrales, como se ha escrito arriba, tiene múltiples usos en varios campos. El primer campo en el que se uso fue en el de la astronomía, dado que por la lejanía de los objetos a estudiar, estos solo pueden ser vistos en longitudes de onda fuera del espectro de la luz visibles, o en su defecto, estudiar la composición del objeto. Los espectrómetros son usados constantementes en misiones NEO (Near Earth Object) para analizar y comprender mejor estos objetos [23] [24]. La aplicación del estudio de imágenes hiperespectrales no se ha limitado solo a objetos espaciales, sino que se aplica también en el análisis de el efecto que las condiciones extremas del espacio tienen en los materiales de protección que se usan en los satélites o naves usadas en las misiones [25]. Otro de los campos en los que más se usa la espectroscopia y las imágenes es la geología. Las imágenes hiperespectrales de satélite o aéreas permiten estudiar la composición del terreno [26] [27]. Las imágenes hiperespectrales son usadas también en medicina [28] dado que ofrecen un gran potencial para analizar tejidos de forma no invasiva, permitiendo diagnosis en el momento [29]. Por la misma razón anterior, el hecho de que la espectroscopia no es dañina ni invasiva al realizarse, es una técnica extendida en restauración y conservación de arte y la arqueología [30].

No solo se ha extendido el empleo de las imágenes hiperespectrales en los campos anteriores, sino que también se encuentra presente en la industria para el control de calidad de los productos [31]. El rango de longitudes de onda usado comúnmente es el NIR (near infra-red), que son las longitudes de onda directamente a la derecha de aquellas pertenecientes al espectro de la luz visibles, como puede verse en la figura . El uso de las técnicas de imagen hiperespectral comenzó en la década de los 2000, convirtiéndose en un proceso de análisis emergente y novedoso en el control de calidad de la industria alimenticia [32]. Estas técnicas se usan, sobre todo en la identificación de daños o defectos

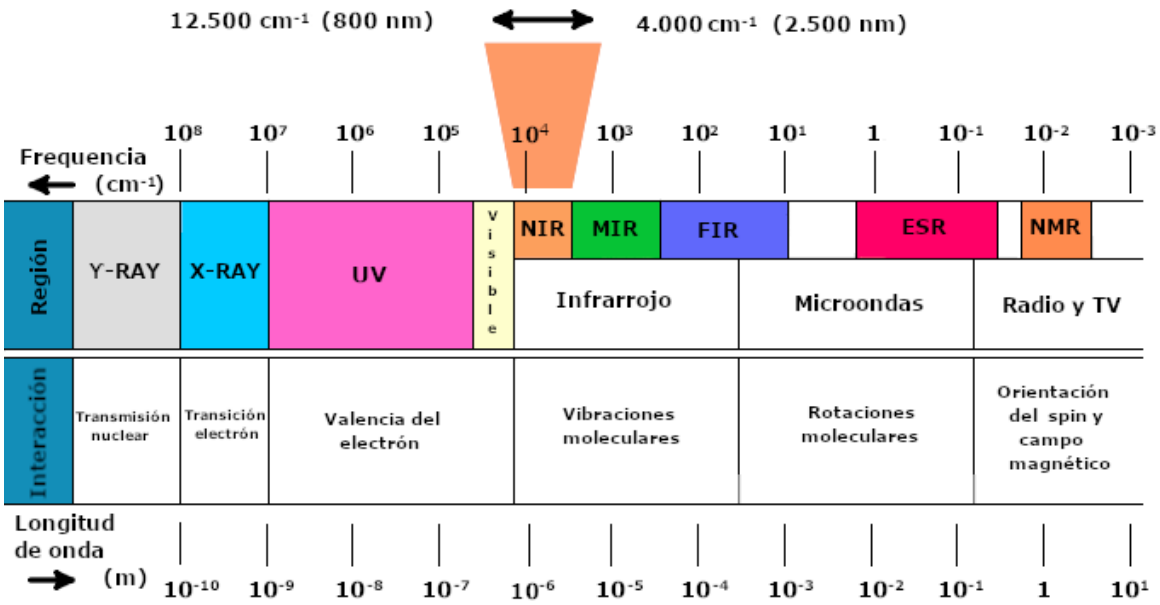


Figura 3.10: espectro electromagnético con las longitudes de onda NIR resaltadas. Fuente: Bruker Optik, 2022

en los productos [33] [34], aunque son empleadas también en la estimación de las características del producto que resultan atractivas de determinar, como aquellas que definan la textura o el aspecto, dado que los consumidores buscarán aquellos productos con mejor aspecto, como la firmeza de la fruta [35] o el marmoleo de la carne [36].

CAPÍTULO 4
Experimentos y Resultados

4.1 Creación del conjunto de datos

Las muestras del producto utilizadas en este estudio han sido facilitadas por una empresa alimentaria que elabora, entre otros productos en conserva, atún en lata. El conjunto se compone de atunes de diferentes tipos: *Thunnus albacares* (*yellowfin*, o de aleta amarilla) y *Katsuwonus pelamis* (conocido como Listado, Rayado o Bonito). Cada lata de atún analizada pertenece a un lote específico, las muestran se dividen en 11 lotes, los lotes del 5 al 13 presentan ocho latas de atún cada uno, mientras los lotes 14 y 15 presentan 5. Para cada muestra, se han proporcionado: la imágen hiperespectral a analizar más la información anterior referente al lote y la especie, junto a los valores de los seis atributos de la textura a predecir.

Tabla 4.1: tabla con información de los lotes analizados (número de lote. código de lote, código de las muestras, número de muestras en el lote, línea de producción y especie de atún utilizada).

Lote	Código de lote	Muestra	n	Línea	Especie
1	5	66-73	8	3	Listado
2	6	74-81	8	1	Yellowfin
3	7	82-89	8	1	Yellowfin
4	8	90-97	8	3	Yellowfin
5	9	98-105	8	5	Listado
6	10	106-113	8	3	Listado
7	11	114-121	8	1	Listado
8	12	122-129	8	3	Yellowfin
9	13	130-137	8	3	Yellowfin
10	14	138-142	5	1	Yellowfin
11	15	143-147	5	1	Yellowfin

Las imágenes hiperespectrales han sido tomadas mediante el espectrómetro Bruker Optics Matri-F dúplex, un equipo NIR (Near Infra-red) de alta resolución que trabaja con espectroscopia infrarroja por transformada de Fourier. Para cada muestra, se ha usado una sonda de 10 *mm* de diámetro iluminando la muestra en la zona central con 60 *mm* de

diámetro. Los espectros obtenidos pertenecen a un intervalo de 12000 a 4000 cm^{-1}, 833.3 a 2500 nm. Estos espectros se registraron realizando 16 escaneados con una resolución de onda de 8 cm^{-1}/s, este proceso se realizó por triplicado para cada muestra.

Cada imágen hiperespectral de cada muestra ha sido almacenada en un fichero *.bil*, en este caso se tienen 42 ficheros, de los cuáles, dos de ellos representaban una muestra cada uno, y los demás presentaban dos muestras cada fichero. Cada fichero *.bil* es acompañado por un fichero cabecera (*.bil.hdr*) que proporciona metadatos de la imagen, como la altura, anchura o profundidad, además de los anchos de banda de cada uno de los canales (o capas de profundidad) de la imágen. De esta forma, cada imágen conforma un cubo de datos en el que las filas y las columnas son la altura y anchura de la imágen respectivamente, mientras que la profundidad representa cada uno de los canales con sus respectivos anchos de banda, cada imágen presenta 300 canales de profundidad, con una diferencia de 2 unidades entre el ancho de banda de un canal y su siguiente o anterior.

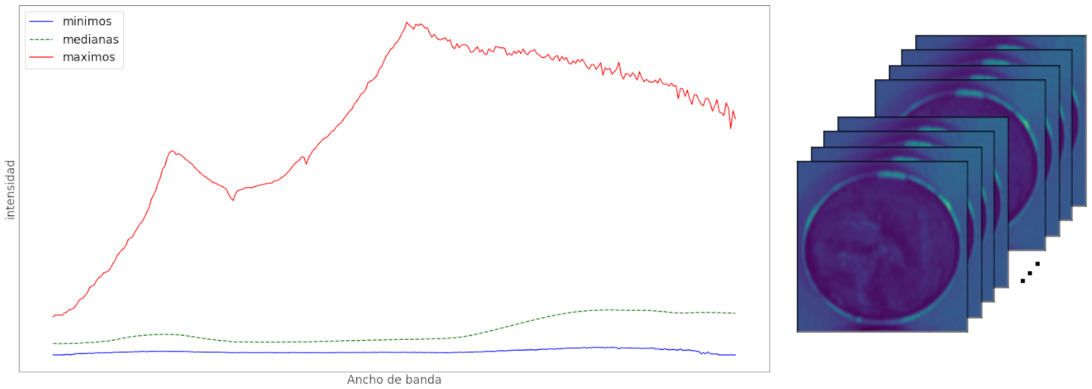


Figura 4.1: esquema de una de las imágenes hiperespectrales en la que se ve la estructura de canales de ancho de banda junto a un gráfico que muestra la distribución de la intensidad para cada canal. Fuente: autor.

Como las imágenes son de tipo hiperespectral, estas no pueden ser abiertas como las imágenes convencionales RGB usando las librerías de PILLOW u OpenCV, por tanto, se ha usado el módulo `gdal` de la librería `osgeo` (apéndice B.1). Esta librería se usa normalmente para trabajar con imágenes de satélite en formato hiperespectral, por lo que es la librería idónea para acceder a la información dentro de los ficheros. Al abrir el fichero, la información contenida en este es almacenada en un objeto `Dataset` del módulo `gdal`, pudiendo acceder a los diferentes atributos como si de variables de una clase se tratasen. Los canales de la imágen son accesibles mediante el método `GetRasterBand()`, al cuál se le proporciona un índice, este índice corresponde con la ubicación del canal, siendo el índice 0 el primer canal y el índice n el último canal. El objetivo es obtener, para cada uno de los ficheros, un matriz numpy igual a la que se obtendría de los módulos OpenCV o PILLOW. Para conseguirlo se abre cada fichero, se obtiene el número de canales (300) y se recorre un bucle que añade a una lista el canal, que está representado como una matriz 2D de ancho y alto igual a la imágen, esta lista es convertida en una matriz numpy en pos de procesarla más fácilmente.

Al tener 300 canales de profundidad sería difícil el poder usar las imágenes en una red neuronal convolucional dado que el coste sería demasiado elevado, por tanto, se debe realizar una compresión en a los canales de la imágen. El tipo de compresión elegido es el calculo de la media de los valores de los pixeles cada 30 canales, generando de esta

forma una imágen con 10 canales de profundidad, reduciendo el coste de procesado por la red neuronal a niveles aceptables.

Una vez se han realizado los pasos previos, si en el fichero presenta dos muestras (visible en el nombre de la muestra, ej.: L5M66M67.bil) estas deben ser separadas, por lo que se realiza un *cut & crop* por el que se secciona la imagen por la mitad horizontalmente, y se recortan los lados para hacer que la imagen sea cuadrada. En el caso de caso de que la imágen presente una sola muestra se realiza un *cut & crop* para centrar y ajustar la muestra (apéndice B.1).

Por último, las matrices son nuevamente comprimidas, esta compresión consiste en reducir las dimensiones de la matriz a una matriz cuadrada 256x256. Para realizar esta compresión se usa el método `zoom()` de la librería `numpy` de Python, al que se proporcionan como parámetros la mátriz, y las razones por la que se quiere comprimir, en este caso 256 dividido la dimensión (un argumento es el alto de la matriz y otro el ancho). Realizada esta compresión se guardan las matrices procesadas para poder importarlas cuando se hagan los experimentos, y para guardar las matrices, se usa el método `save()` de la librería `numpy`. Este método genera un fichero .npy que puede ser cargado como un objeto `numpy.array` de Python con el método `numpy.load()` (apéndice B.1).

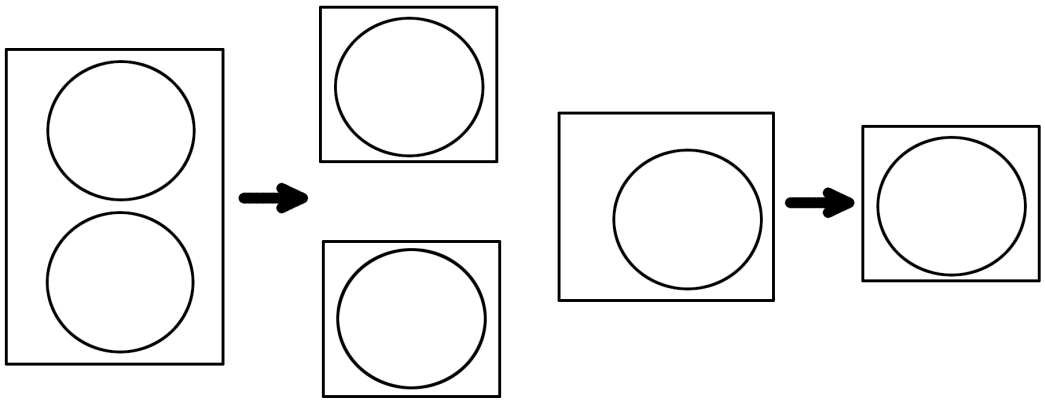


Figura 4.2: prceso de cut & crop en la imagen hiperespectral dependiendo del número de muestras. Fuente: autor.

Por todo lo anterior, de las imágenes facilitadas para los experimentos se ha extraído los datos, los registros hiperespectrales, y se han transformado en matrices *numpy* que son más manejables a la hora de analizar y procesar. Estas matrices se han preprocesado para eliminar imperfecciones como muestras mal centradas o dos muestras a la vez en una imagen, siendo al final de este paso todas las matrices similares en forma. Debido al tamaño de las matrices a manejar es necesario realizar una compresión de estas para que sea más fácil analizarlas por la red neuronal convolucional, por tanto, se realiza un redimensión de las matrices a una matriz cuadrada de tamaño 256x256, y al final del proceso se guarda cada matriz en un fichero de tipo *numpy* con el identificador de la muestra L-Lote-M-Muestra.

Más adelante en los experimentos, se accederá a estos ficheros para extraer las matrices y usarlas. Antes de construir los conjuntos de datos de entrenamiento y prueba se accedera a los datos y se alamcenará en diversas variables (apéndice B.2). En cuanto a las imágenes, estas serán alamcenadas en una lista creando una estructura de dimensiones N, y, x, λ siendo N el número de muestras, y y x las dimensiones altura y anchura de las imágenes y λ el número de canales de la matriz (o profundidad). Las etiquetas serán diferenciadas entre atributos, separándose estos en diferentes listas de Python. Una vez

se tiene las estructuras necesarias, se normalizan los datos, de esta forma, a futuro, se podrán realizar los experimentos con las muestras que se obtengan con mayor facilidad (apéndice B.2).

4.2 Topología de las CNN empleadas

Para realizar los experimentos se ha decidido modificar la topología de la red, el tamaño de *batch* y generar más datos mediante rotar y voltear la imágen. Dado que la cantidad de muestras proporcionada es muy exigua. Dado el objetivo específico de vislumbrar cuales de los diferentes atributos de la textura del producto es ostenta mayor facilidad de describirse por la información de la imagen hiperespectral, se ha decidido generar una CNN para cada uno de los atributos de la textura a estudiar: Fuerza (H), Cohesión, Adhesión, Elasticidad, Masticabilidad y Gomosidad. En todas las redes neuronales convolucionales construidas se empleará como función de activación la función ReLu y no se realiza ningún *padding*.

Lo primero que se planteó fue la topología de la red, se decidió realizar cuatro topologías, la primera inspirada en la arquitectura AlexNet (figura 4.3). Esta red presenta 4 bloques convolucionales y 3 bloques *fully concected* o densamente conectados, cada uno de los bloques convolucionales es complementado con un submuestreo *maxpool* de diferentes tamaños. El primer bloque convolucional presenta 96 filtros convolucionales de tamaño de *kernel* 11x11, un submuestreo de tamaño 3x3 y su salida presenta dimensiones 85x85x96. El segundo asciende a 256 filtros de tamaño 5x5 y un submuestreo de 3x3, con una salida 28x28x256. El tercer bloque presenta 3 subbloques en los que los dos primeros tienen 384 filtros mientras el último tiene 256, todos con tamaño de 3x3, y con un *pooling* de 3x3 que ofrece una salida de tamaño 9x9x256. El cuarto bloque convolucional presenta 256 filtros de tamaño 3x3 con un *pooling* de tamaño 9x9, que de esta forma presenta una salida de tamaño 1x1x256. Estos submapas avanzan a las dos siguiente bloques densamente conectados, de tamaño 4096x1x1 ambos, y el último bloque densamente conectado, del cual la salida es un único elemento, el valor requerido (apéndice C).

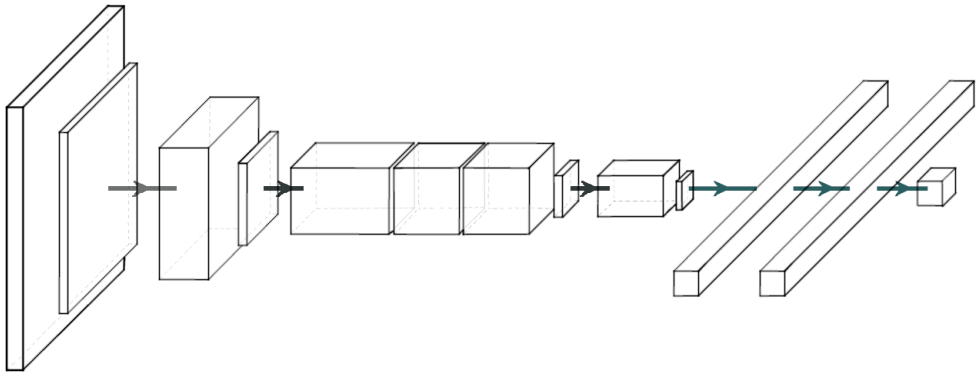


Figura 4.3: arquitectura AlexNet propuesta. Fuente: autor.

La segunda arquitectura ideada se trata de una más simple que la anterior, la AlexNet, y es una configuración basada en la arquitectura *ShallowNet*. Este tipo de arquitectura consiste únicamente en un bloque convolucional y un bloque densamente conectado que obtiene la salida. La última capa, la densamente conectada ofrece el valor requerido. El bloque convolucional genera 32 filtros de tamaño de *kernel* 3x3 con la función de

activación ReLu (figura 4.4). Esta es la arquitectura más simple de las cuatro propuestas (apéndice C).

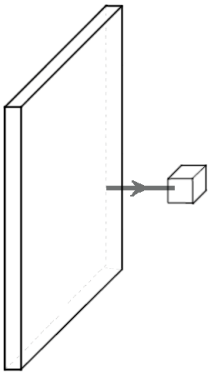


Figura 4.4: arquitectura ShallowNet propuesta. Fuente: autor.

La tercera configuración de CNN se basa en la arquitectura *LeNet*, la cual se conforma de dos bloques convolucionales con sus respectivos bloques de submuestreo o *pooling*, seguidos de dos capas densamente conectadas y la salida de la red, que consiste de un único valor, el requerido que consiste en el valor del atributo buscado (figura 4.5). El primer bloque convolucional consiste en 6 filtros de tamaño de *kernel* 3x3, seguido de un submuestreo o *pooling* de tamaño con 2x2. El segundo bloque presenta 16 filtros convolucionales de tamaño 5x5, terminado en un *pooling* de tamaño 128x128. La salida de este segundo bloque convolucional procede a las capas densamente pobladas, siendo la primera consistente de 120 nodos, teniendo la salida un tamaño de 1x1x120. La segunda capa densamente poblada presenta 84 nodos y obtiene una salida de tamaño 1x1x84. La salida de esta segunda capa se traslada a la capa densamente poblada que funciona como salida de la red que obtiene una salida de tamaño 1x1x1, consistente del valor requerido del atributo que se esté analizando (apéndice C).

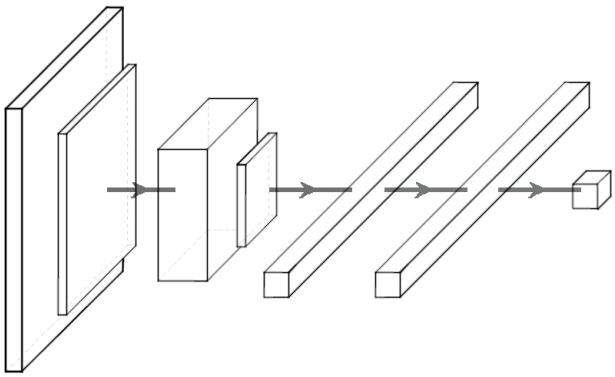


Figura 4.5: arquitectura LeNet propuesta. Fuente: autor.

La cuarta configuración de CNN es una arquitectura ideada sin basarse en ninguna otra, consiste en siete bloques convolucionales, una capa de aplanamiento y un capa densamente conectada que sirve de salida de la red (figura 4.6). El primer bloque convolucional consiste en 32 filtros de tamaño 3x3 de *kernel*, seguido de un *pooling* de tamaño 2x2. Los siguientes bloques convolucionales presentan una estructura similar al primero,

variando únicamente el número de filtros convolucionales en cada uno de ellos, aumentando al doble en cada uno de ellos respecto al anterior. De esta forma, se tiene que el segundo bloque convolucional presenta 64 filtros, en el tercero son 128, el cuarto se compone de 256, y los tres últimos bloques mantiene la cantidad a 512 filtros. Una vez se ha llegado a último bloque convolucional, la salida de este se aplana con una capa de aplanamiento, la cuál sirve de entrada para la capa densamente conectada que se emplea como salida de la red. De esta forma, se obtiene un único valor que describe el atributo de la textura a analizar (apéndice C).

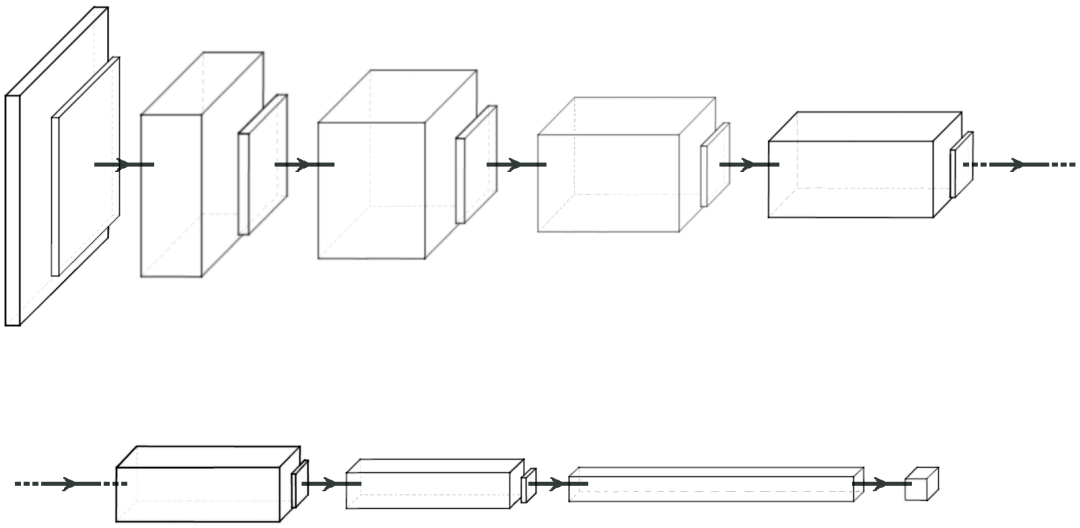


Figura 4.6: arquitectura propia propuesta. Fuente: autor.

4.3 Estrategias de los experimentos

Una vez se han configurado las arquitecturas a emplear en los experimentos, se han decidido los demás aspectos que definirán los experimentos. Estos se variarán en cada experimento en el que se entrene las redes convolucionales para encontrar la CNN óptima que describa mejor los valores de los atributos a analizar. En cuanto a los entrenamientos de las redes convolucionales, se ha fijado el número de *epcohs* a 100, considerándose este como un número lo suficiente elevado para que conduzca a una convergencia en el valor de pérdida de la red y lo suficiente bajo para que el coste temporal de entrenamiento sea permisible. La función de pérdida, con la cual se calcula el valor de pérdida de la red, que sirve de control del desempeño de la misma se ha definido como el error cuadrático medio (MSE) u la raíz del error cuadrático medio (RMSE). Y como optimizador se ha elegido el optimizador ADAM con un factor de aprendizaje de 0.0001. Dado que la población de muestra es pequeña se hará uso del método `ImageDataGenerator()` de `Tensorflow`. Este método genera más imágenes con base en las imágenes existentes distorsionando o modificándolas, como girar la imagen cierto ángulo o dar la vuelta a la misma horizontal o verticalmente, además de otras modificaciones como el brillo, recortar, cambiar canales de orden, reescalar la imagen o modificar el *zoom* a la imagen. En estos experimentos se usaran los modificadores de giro y volteado, teniendo diferentes giros: 5, 10, 12 y 15 grados de giro (figura 4.7).

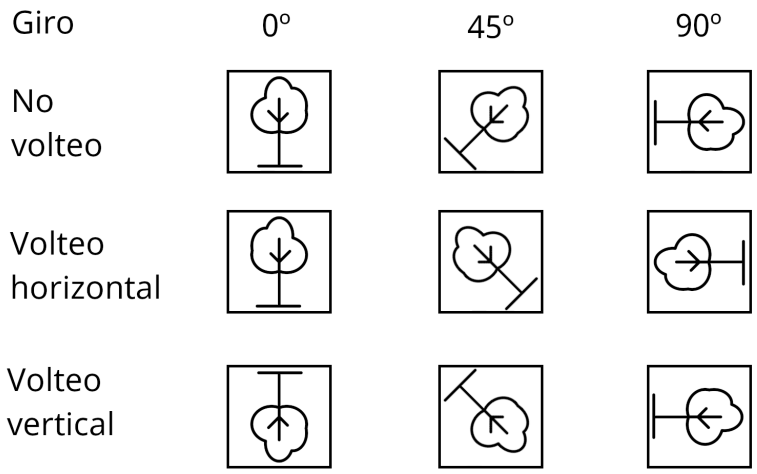


Figura 4.7: ejemplos de uso del método ImageDataGenerator(). Fuente: autor.

4.4 Cálculo de error, evaluación y optimización

Como antes se ha comentado, para los experimentos se ha definido como método de cálculo del error, o función de pérdida y de evaluación de la red convolucional los estadísticos MSE (4.1) y RMSE (4.2).

$$MSE = \frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{N} \quad | \quad 0 < i < N, \quad N \in \Re \tag{4.1}$$

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N}(y_i - \hat{y}_i)^2}{N}} \quad | \quad 0 < i < N, \quad N \in \Re \tag{4.2}$$

El MSE es un estimador que mide la media de los errores cuadrados, por tanto es la media de la diferencia al cuadrado entre el valor estimado y el valor real. En *machine learning* se usa como un estimador del riesgo empírico, que implica la pérdida media que se observa en la población de muestra, que es una estimación de la verdadera pérdida media sobre la población total. Por esto, es usado como función de pérdida, que describe el coste asociado al estado n-ésimo de la red, también llamado función de coste. Como se puede ver en la fórmula 4.1, este es calculado como la suma de las diferencias al cuadrado, dividido entre el tamaño de la muestra. En regresión, se minimiza el MSE cuanto mejor precisión muestre el modelo, por tanto, se ha definido en la red convolucional como el parámetro destinado a ser minimizado.

En cambio, el RMSE se ha empleado para comparar las diferentes redes dado su uso en la comparación de modelos, ya que representa la precisión del modelo respecto a una muestra dada. Específicamente, sirve para agregar la magnitud del error a las predicciones para definir la capacidad de predicción del modelo, en este caso la red convolucional. Por tanto, el RMSE es una forma modificada del MSE y se calcula como la raíz cuadrada de este mismo, como se puede ver en la formula 4.2.

Por otro lado, como optimizador de la red se ha empleado el optimizador ADAM (formula 4.3).

$$V_t = \beta V_{t-1} + (1-\beta)\nabla f(\theta_t) \qquad V_{t,corr} = \frac{V_t}{1-\beta^t}$$

$$S_t = \rho S_{t-1} + (1-\rho)(f(\theta_t) \cdot f(\theta_t)) \qquad S_{t,corr} = \frac{S_t}{1-\rho^t} \tag{4.3}$$

$$\theta_{i,t+1} = \theta_{i,t} - \alpha \frac{V_{i,t,corr}}{\epsilon + \sqrt{S_{i,t,corr}}}$$

Es un optimizador basado en descenso por gradiente que funciona bien con redes profundas y modelos de *deep learning*, y su nombre viene de ADAptative Moment Estimation (ADAM estimation). Este método emplea internamente otros dos llamados RMSProp y Descenso del gradiente por momentum (GD con momentum), lo cuál se ve en su formula, siendo el parámetro V_t originario de GD con momentum, y S_t de RMSProp. Este método que adapta la estimación conforme se avanza en el descenso, haciendo uso del cuadrado del gradiente para escalar el *learning rate* y genera un *momentum* en el movimiento mediante la media móvil del gradiente. Como es un método adaptativo, proporciona un *learning rate* en cada iteración, adaptándose así al gradiente de la función y permitiendo un descenso controlado.

4.5 Experimentos y resultados

El objetivo, como bien se ha explicado en el apartado 1.2, es obtener una red neuronal convolucional que permita, para cada atributo, obtener los valores de este de una imagen dada. Por tanto, los experimentos han sido dirigidos en pos de obtener las CNN que obtengan mejores resultados para cada atributo. Uno de los mayores problemas que se han encontrado en la realización del proyecto es la exigua cantidad de datos a manejar, los experimentos y proyectos realizados con técnicas y herramientas de *Deep Learning* y *Machine Learning* se realizan con masivas cantidades de datos con el fin de tener la mayor conjunto de la población real representado en la población de muestra, para así poder representar las tendencias, patrones y comportamientos de la forma más precisa posible. Uno de los hiperparámetros que define el número de muestras a trabajar antes de actualizar los parámetros internos de la red en el *batch size*, este puede ser desde 1 al tamaño de la muestra (N). Para poder realizar varias operaciones y aprovechar la poca cantidad de muestras se estudiará el desempeño de las CNNs anteriormente descritas con cuatro tipos de *batch size*: 4, 8, 16, 32. Para elegir cuál de los tamaños de *batch* se empleará en los experimentos se realiza un entrenamiento con cada una de las arquitecturas ideadas con un aumento de datos usando `ImageDataGenerator()` con un ángulo de 5º.

En total, se realizaron 16 entrenamientos con cuatro tipos de arquitectura distinta, modificando en en cada cuarteto el tamaño de *batch*. Como se ve en las tablas 4.2 y 4.3, la arquitectura original del experimento genera resultados mejores que las demás en cada uno de los tipos de tamaño de *batch*, por tanto, se ha decidido emplear este tipo de arquitectura en los experimentos siguientes. En cuanto al tamaño de *batch*, se ve claramente como el tamaño que presenta en general un valor de MSE menor es el tamaño 8. Por tanto, el tamaño de *batch* que desemboca en un mejor desempeño de la red es el que genera ocho subgrupos de muestras aleatorias que permite entrenar la red ocho veces en un único epoch del entrenamiento general, permitiendo recoger y aprender de patrones y comportamientos de estas ocho muestras y actualizar con las siguientes, pero sin dar demasiada importancia a las propias muestras, como sería con un tamaño de *batch* menor, ni generalizarlas en demasía, como ocurre cono mayores *batch*, al generar grupos más amplios.

Tabla 4.2: tabla con los valores de pérdida (MSE) de las arquitecturas elegidas para los tamaños *batch* seleccionado y un amuento de datos con duplicados 5º inclinados.

Batch	Arquitectura	H	Cohesión	Adhesión	Elasticidad	Gomosidad	Masticabilidad
4	AlexNet	1.41	1.05	0.69	1.07	1.60	1.46
	ShallowNet	1.60	1.09	0.69	1.10	1.59	1.58
	LeNet	1.69	1.43	0.53	1.09	1.40	1.60
	CNN propia	1.35	0.49	0.58	0.90	1.27	1.10
8	AlexNet	1.39	1.12	0.90	1.05	1.59	1.54
	ShallowNet	1.42	1.19	0.90	1.00	1.52	1.62
	LeNet	1.42	1.13	0.99	1.00	1.47	1.63
	CNN propia	**1.23**	**0.42**	**0.36**	**0.77**	**1.04**	**1.08**
16	AlexNet	1.63	0.48	1.38	1.61	1.45	1.59
	ShallowNet	1.79	0.52	1.10	1.49	1.33	1.83
	LeNet	1.75	0.96	1.23	1.45	1.32	1.71
	CNN propia	1.34	0.44	1.02	1.43	1.16	1.22
32	AlexNet	2.10	1.19	1.98	1.83	2.00	1.48
	ShallowNet	2.01	0.96	1.41	1.45	1.96	1.39
	LeNet	1.99	1.26	1.59	1.67	1.53	1.30
	CNN propia	1.75	0.77	1.31	1.39	1.44	1.15

Tabla 4.3: tabla con los valores de pérdida (RMSE) de las arquitecturas elegidas para los tamaños *batch* seleccionado y un amuento de datos con duplicados 5º inclinados.

Batch	Arquitectura	H	Cohesión	Adhesión	Elasticidad	Gomosidad	Masticabilidad
4	AlexNet	1.19	1.03	0.84	1.04	1.27	1.21
	ShallowNet	1.27	1.05	0.84	1.05	1.26	1.26
	LeNet	1.3	1.2	0.73	1.05	1.18	1.27
	CNN propia	1.17	0.7	0.76	0.95	1.13	1.05
8	AlexNet	1.18	1.06	0.95	1.03	1.26	1.24
	ShallowNet	1.19	1.09	0.95	1.0	1.23	1.28
	LeNet	1.19	1.06	1.0	1.0	1.22	1.28
	CNN propia	**1.11**	**0.65**	**0.61**	**0.88**	**1.02**	**1.04**
16	AlexNet	1.28	0.7	1.18	1.27	1.21	1.26
	ShallowNet	1.34	0.73	1.05	1.22	1.15	1.35
	LeNet	1.32	0.98	1.11	1.21	1.15	1.31
	CNN propia	1.16	0.66	1.01	1.2	1.08	1.11
32	AlexNet	1.45	1.09	1.41	1.36	1.42	1.22
	ShallowNet	1.42	0.98	1.19	1.2	1.4	1.18
	LeNet	1.41	1.12	1.26	1.3	1.24	1.14
	CNN propia	1.33	0.88	1.15	1.18	1.2	1.08

Estudiando la tabla se puede observar que, general, las arquitecturas AlexNet, ShallowNet y LeNet presentan un desempeño similar en cada uno de los tamaños de *batch*, todos los valores se encuentran de una a seis décimas por encima del valor de pérdida de la red con la arquitectura original. Resaltar que en ciertos atributos esta diferencia se encuentra más visible, como en el caso de los atributos *Cohesión*, *Adehsión* y *Elasticidad*, mientras que este efecto es menor en los atributos *Fuerza (H)*,*Gomosidad* y *Masticabilidad*. Otro aspecto que se puede ver en la tabla es el hecho de que *Cohesión*, *Adehsión* y *Elasticidad* son mejor descritos por la red neuronal convolucional que los otros tres. Este hecho puede ser debido a la naturaleza de las imágenes hiperespectrales, dado que las muestras se han tomado con imágenes hiperespectrales de valores infrarrojos. Estos valores infrarrojos registran las vibraciones de las moléculas, por lo que son más susceptibles a quedar grabados aquellos atributos que se puedan describir por el estado molecular de la sustancia, como la cohesión, la adhesión y elasticidad.

Por todo lo anterior, se puede afirmar que, entre las cuatro arquitecturas de red neuronal convolucional seleccionadas, la configuración original de este trabajo es la que mejor desempeño muestra, obteniendo un valor de pérdido menor que las demás. No solo eso, sino que entre los cuatro tipos de tamaño de *batch*, el que mejor desempeño proporciona a la red es el tamaño de 8 muestras por *batch*. Esto puede deberse a que el tamaño de la muestra de entrenamiento consiste en 64 imágenes, que divididas en grupos de ocho generan ocho grupos, mientras que los otros tamaños generaban grupo o muy pequeños o muy grandes en los que o bien no se alcanzaba a discernir una individualidad por haber demasiadas muestras que comparar o bien no se alcanzaba una tendencia por disponer de pocas muestras que analizar a la hora de actualizar los pesos de la red.

Tabla 4.4: tabla con los valores de pérdida de la arquitectura elegida en el primer experimento.

H	Cohesión	Adhesión	Elasticidad	Gomosidad	Masticabilidad
1.2328	0.4276	0.3682	0.7758	1.0416	1.0874

Una vez se ha elegido una arquitectura y un tamaño de *batch* se puede pasar a experiemtos con el tamaño de la población de muestra. Dado que la población facilitada para el proyecto es pequeña se requiere el uso de las herramientas de generación de datos a partir de los existentes, modificándolos. Como se ha explicado antes, el proceso más extendido en el análisis de imágenes por *Deep Learning* es el modificar la imagen ya sea inclinándola, girándola o volteándola, cambiando el contraste u otras modificaciones. Dado que las muestras se toman con unas condiciones muy estrictas de luminosidad no tiene sentido el modificar aspectos como el contraste o el brillo de la imagen, ya que esta será siempre la misma a la hora de recoger la información por el espectroscopio. Dado esto, se ha decido realizar un aumento de datos mediante inclinar las imágenes distinto ángulos: 5º, 10º, 12º y 15º; voltear las imágenes en horizontal y en vertical. Al igual que con la anterior explicación del porqué de no usar modificadores de brillo y contraste, la toma en cadena de las imágenes hiperespectrales puede generar pequeñas modificaciones en la posición de la muestra, por tanto, es plausible el hecho de que una muestra similar a una de entrenamiento se encuentre girada o en espejo respecto a la usada para entrenar la red, por este razonamiento se ha estimado que el uso de estos modificadores puede ser beneficioso para el entrenamiento y plausible para generar nuevas imágenes y obtener una población mayor.

Como se puede observar en las tablas 4.5 y 4.6, mayoritariamente el desempeño de la red mejora por atributos al añadir duplicados de las muestras con la modificación de volteo horizontal, la única excepción es la cohesión, cuya red con mejor valor pérdida se ha logrado solo al inclinar la imagen. También la mayoría de los atributos han presentado

Tabla 4.5: tabla con los valores de pérdida (MSE) de las distintas configuraciones de *data generation*

Vertical	Horizontal	Ángulo	H	Cohesión	Adhesión	Elasticidad	Gomosidad	Masticabilidad
1	1	5	0.9668	0.5615	0.4345	0.4785	1.0329	1.0432
		10	0.8735	0.5979	**0.2732**	0.4882	0.9982	0.9232
		12	0.9191	0.5649	0.3962	0.4723	1.0734	1.0058
		15	0.9307	0.7489	0.2833	0.3103	0.8803	1.0536
0	1	5	**0.6627**	0.4979	0.3412	0.5943	0.8059	0.8681
		10	0.7864	0.4377	0.3509	0.5041	0.8043	1.0235
		12	0.8441	0.4903	0.3531	0.6949	0.8411	0.7937
		15	0.8987	0.4669	0.3394	0.4647	0.9147	0.9741
1	0	5	1.0625	0.6645	0.4097	0.6102	1.1246	1.1194
		10	1.2946	0.4649	0.2876	0.6177	1.1314	1.1581
		12	1.2753	0.5292	0.3871	0.7075	1.1239	1.1723
		15	1.1038	0.7421	0.3407	0.5132	1.1699	0.9336
0	0	5	1.2328	0.4276	0.3682	0.7758	1.0416	1.0874
		10	1.0087	**0.4049**	0.4199	0.7103	1.0932	1.1092
		12	1.0452	0.4372	0.3631	0.6589	1.0321	1.0722
		15	0.9625	0.4638	0.3211	0.6845	0.9529	1.3715

Tabla 4.6: tabla con los valores de pérdida (RMSE) de las distintas configuraciones de *data generation*

Vertical	Horizontal	Ángulo	H	Cohesión	Adhesión	Elasticidad	Gomosidad	Masticabilidad
1	1	5	0.98	0.75	0.66	0.69	1.02	1.02
		10	0.93	0.77	**0.52**	0.7	1.0	0.96
		12	0.96	0.75	0.63	0.69	1.04	1.0
		15	0.96	0.87	0.53	0.56	0.94	1.03
0	1	5	**0.81**	0.71	0.58	0.77	0.9	0.93
		10	0.89	0.66	0.59	0.71	0.9	1.01
		12	0.92	0.7	0.59	0.83	0.92	0.89
		15	0.95	0.68	0.58	0.68	0.96	0.99
1	0	5	1.03	0.82	0.64	0.78	1.06	1.06
		10	1.14	0.68	0.54	0.79	1.06	1.08
		12	1.13	0.73	0.62	0.84	1.06	1.08
		15	1.05	0.86	0.58	0.72	1.08	0.97
0	0	5	1.11	0.65	0.61	0.88	1.02	1.04
		10	1.0	**0.64**	0.65	0.84	1.05	1.05
		12	1.02	0.66	0.6	0.81	1.02	1.04
		15	0.98	0.68	0.57	0.83	0.98	1.17

un mejor desempeño al inclinar la imagen unos 10º respecto a las demás inclinaciones, independientemente de si se voltea horizontal o verticalmente. Cabe recalcar que, al igual que en el primer experimento, se distingue la tendencia de que los atributos *Cohesión*, *Adhesión* y *Elasticidad* presentan, en conjunto, valores de pérdida menores que *Fuerza*, *Gomosidad* o *Masticabilidad*. También destacar que *Adhesión* presenta un diferencia entre el mayor y el menor valor de pérdida muy pequeña, por lo que es posible que se haya llegado a un límite en la precisión que pueda aportar la red con los datos disponibles.

Otro experimento que se ha realizado es la creación de una única red neuronal convolucional, con la arquitectura escogida, que obtenga los valores de los atributos. La organización de este último experimento ha sido similar al anterior, se ha realizado un entrenamiento a 16 redes convolucionales con diferentes configuraciones de generación de datos, variando la inclinación de la imágen y volteando horizontal y verticalmente. Estudiando las tablas 4.7 y 4.8 se puede encontrar una diferencia muy notable entre los valores de pérdida obtenidos por las redes convolucionales dedicados a un atributo cada una y los valores de la red convolucional general. Estos valores obtenidos por la red convolucional general son menores, globalmente, que los de las redes convolucionales individuales. Además, los mejores valores, para cada uno de los atributos, pertenecen a la misma configuración de *data augmentation*, en concreto a la configuración: inclinar 5 grados la imagen, voltear verticalmente y no horizontalmente. Otra de las grandes diferencias entre los resultados de esta red y la anterior es la ausencia de grandes diferencias entre los valores por atributo. Mientras que en las anteriores las diferencias entre valores de pérdida entre atributos podía llegar a ser de cinco décimas, en el caso de la red global la máxima diferencia es de dos décimas. Y recalcar también, que a diferencia de las redes anteriores, en esta, los valores de pérdida de *Fuerza*, *Gomosidad* y *Masticabilidad* llegan a ser más bajos que los valores de *Cohesión*, *Adhesión* y *Elasticidad*. Puede ser que los valores de estos atributos sean dificiles de obtener por separado, pero relacionados con las medidas de *Cohesión*, *Adhesión* y *Elasticidad* se puedan obtener. Por todo lo anterior, se puede afirmar que es más rentable, ya sea en coste temporal, de cómputo o de precisión, el uso de una red neuronal convolucional que obtenga todas las medidas de una muestra al mismo tiempo, en vez de emplear paralelamente diversas redes que obtengan cada una la medida de un único atributo.

Tabla 4.7: tabla con los valores de pérdida (MSE) de las distintas configuraciones de *data generation* para una red global.

Vertical	Horizontal	Ángulo	Global	H	Cohesión	Adhesión	Elasticidad	Gomosidad	Masticabilidad
1	1	5	0.5	0.50	0.54	0.75	0.52	0.47	0.48
		10	0.45	0.41	0.47	0.54	0.46	0.41	0.39
		12	0.63	0.59	0.62	0.72	0.66	0.58	0.59
		15	0.54	0.48	0.54	0.73	0.52	0.48	0.48
1	0	5	0.18	0.18	0.10	0.20	0.24	0.19	0.19
		10	0.31	0.28	0.26	0.42	0.34	0.27	0.29
		12	0.33	0.28	0.28	0.51	0.36	0.28	0.29
		15	0.40	0.35	0.37	0.54	0.40	0.36	0.38
0	1	5	0.3	0.33	0.34	0.44	0.35	0.35	0.36
		10	0.43	0.42	0.42	0.49	0.40	0.40	0.43
		12	0.45	0.41	0.45	0.56	0.41	0.42	0.43
		15	0.50	0.43	0.49	0.67	0.48	0.45	0.46
0	0	5	0.24	0.25	0.13	0.24	0.24	0.27	0.29
		10	0.26	0.26	0.17	0.35	0.27	0.24	0.29
		12	0.26	0.27	0.15	0.26	0.31	0.26	0.30
		15	0.31	0.28	0.25	0.38	0.32	0.30	0.33

Tabla 4.8: tabla con los valores de pérdida (RMSE) de las distintas configuraciones de *data generation* para una red global.

Vertical	Horizontal	Ángulo	Global	H	Cohesión	Adhesión	Elasticidad	Gomosidad	Masticabilidad
1	1	5	0.74	0.71	0.74	0.87	0.73	0.69	0.7
		10	0.67	0.65	0.69	0.74	0.68	0.64	0.63
		12	0.8	0.77	0.79	0.85	0.82	0.76	0.77
		15	0.74	0.7	0.74	0.86	0.72	0.7	0.7
1	0	5	0.43	0.43	0.32	0.45	0.49	0.44	0.45
		10	0.56	0.53	0.51	0.65	0.58	0.53	0.55
		12	0.58	0.53	0.53	0.72	0.6	0.54	0.54
		15	0.64	0.6	0.61	0.74	0.64	0.61	0.62
0	1	5	0.61	0.58	0.59	0.67	0.6	0.6	0.6
		10	0.66	0.65	0.65	0.7	0.64	0.63	0.66
		12	0.67	0.65	0.68	0.75	0.64	0.65	0.66
		15	0.71	0.66	0.71	0.82	0.7	0.68	0.69
0	0	5	0.49	0.5	0.37	0.5	0.49	0.53	0.54
		10	0.52	0.51	0.42	0.59	0.52	0.5	0.54
		12	0.51	0.52	0.39	0.51	0.56	0.52	0.55
		15	0.56	0.54	0.51	0.62	0.57	0.55	0.58

CAPÍTULO 5
Conclusiones

En este apartado se van a discutir las conclusiones sobre los resultados obtenidos en los experimentos comentados en el anterior capítulo. Se relacionará el trabajo realizado con las asignaturas y conocimientos obtenidos durante el grado y se discutirán los problemas que han surgido durante la duración del proyecto.

5.1 Conclusiones del trabajo

Una vez se han observado y discutido los resultados de los experimentos, se puede afirmar que es posible generar una red neuronal convolucional que obtenga medidas de los atributos de interés de la textura de la muestra. A su vez, es posible determinar que presenta más y mejores ventajas el uso de una única red neuronal convolucional que obtenga las medidas de la muestra, en lugar de emplear diferentes redes convolucionales para cada uno de los atributos, obteniendo una medida cada una.

Una de estas ventajas son la visible reducción del valor de la función de pérdida de cada atributo por la red global respecto a los valores de pérdida obtenidos por las redes convolucionales dedicadas cada una a un único atributo. Mientras que otra de las ventajas es el hecho de que los valores los valores de pérdida de los atributos *Fuerza*, *Gomosidad* y *Masticabilidad* alcanzan valores mucho menores que los alcanzados por las redes dedicadas a un único atributo, sin verse afectados los buenos resultados obtenidos también en el resto de atributos, que ya con las redes diferenciadas se encontraban con valores bajos, y siendo que con la red global alcanzan valores aun más bajos.

Otra ventaja, no relacionada con los valores de los resultados, es la reducción del coste computacional al tener una única red neuronal convolucional en uso, a diferencia del escenario con las redes convolucionales dedicadas a un atributo. Siendo que en ese caso se tendría que ejecutar paralelamente tantas CNNs como atributos de la textura de la muestra a medir.

En relación a los conocimientos y contenidos del grado, este trabajo puede ser relacionado con campos de conocimiento en los que se puede dividir algunas de las asignaturas del grado. Este trabajo ha servido para afianzar conocimientos extraídos de estas áreas, como el área de estadística, en la que impartieron métodos de análisis y evaluación de modelos, así como estructuras avanzadas como las redes neuronales. Otra área de análisis matemático con la optimización de métodos de aprendizaje automático, que se impartieron en el área de aprendizaje automático y aprendizaje profundo, en la que se impartió la creación de redes neuronales y convolucionales.

Este trabajo ha sido también beneficioso ya que ha permitido aprender y reforzar múltiples competencias como comprensión e integración de conocimientos, dado que en el

transcurso del trabajo se ha realizado una investigación en los temas que se han tratado y se han asimilado con el fin de poder realizar el trabajo lo más actualizado posible. Otra competencia es la aplicación del pensamiento práctico, dado que se ha aplicado conocimientos y estrategias impartidas en el grado. A su vez, la resolución de problemas es una competencia que se ha visto reforzada, al surgir problemas que debían ser resueltos; o el pensamiento crítico, que permite juzgar el trabajo realizado y decidir las acciones a seguir. Por supuesto, el aprendizaje permanente presenta una gran importancia, sobre todo en el área del aprendizaje automático, en la que cada poco tiempo surgen avances y nuevos métodos; y sin olvidar la planificación y gestión del tiempo, de las competencias más importantes a la hora de realizar un proyecto.

5.2 Problemas surgidos en el trabajo

A lo largo de la ejecución de cualquier proyecto surgen problemas de diversas índoles, en este proyecto se ha encontrado las siguientes.

El primer obstáculo que supuso un problema grave al proyecto surgió al principio mismo del trabajo. Este consistía en la falta de datos, la empresa que facilitó los datos proveyó un disco duro en el que se encontraban varios directorios con ficheros .bil, el tipo de fichero de las imágenes hiperespectrales. El problema con el que no encontramos fue el hecho de que el directorio que contenía la mayor cantidad de imágenes, y consistentes de una muestra cada una, era desgraciadamente inutilizable dado que las estas no contenían información, contenían 0KB. Por tanto se tuvo que emplear las imágenes de otro de los directorios, que contenía menos de ellas y la inmensa mayoría contenía dos muestras en una misma imagen.

Como consecuencia de esto, surgió otro nuevo obstáculo, las imágenes que se podían emplear para los experimentos fueron reducidas a 80, de las casi 200 de las que se podía disponer. Lo cual, podría haber influido en los resultados de los experimentos, esto por el hecho de que al disponer de una población de muestra pequeña, las predicciones y tendencias que obtiene la red pueden no describir las tendencias reales de la población real y las predicciones serán menos precisas. Siendo una de las características de un modelo de aprendizaje automático la gran cantidad de datos necesarios para poder generar los modelos matemáticos que describan con fidelidad la población, este hecho ha supuesto un problema que por el contrario, en futuros trabajos, podría suponer una capacidad de precisión enormemente aumentada.

Debido a que las imágenes contenían más de una muestra, estas no podían ser proporcionadas a la red para entrenar sin un preprocesado intensivo, y no solo eso, la gran cantidad de longitudes de onda almacenadas en las imágenes suponía un problema para las redes neuronales, ya que el coste de cómputo se vería seriamente afectado. Debido a esto se tuvo que realizar un preprocesado sobre las imágenes, creando los ficheros Python necesarios, que se pueden consultar en el apéndice B.1.

CAPÍTULO 6
Trabajos futuros

Como más arriba se ha comentado, si se pudiera volver a realizar este proyecto, se destinaría más atención a comprobar que las imágenes proporcionadas fueran suficientes en cantidad para poder realizar un entrenamiento con las redes convolucionales, una técnica de aprendizaje profundo, que pueda proporcionar una precisión mejorada sobre la población real. Dado que las redes convolucionales extraen características y patrones de estas que comparan con el valor real que se intenta estimar. Por tanto, cuantas más imágenes, más patrones y características se dispondrán con el fin de perfilar los atributos del producto que se requieren.

Otra forma de mejorar el desempeño de la red, sería construir otros tipos de arquitecturas, que extraigan más mapas de características. Y no solo eso, sino realizar pruebas con diferentes optimizadores y herramientas que faciliten la optimización.

Bibliografía

[1] GALVÁN, A. Neural plasticity of development and learning. *Human Brain Mapping*, 2010, vol. 31, no. 6, pp. 879-890. Disponible en: https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.21029

[2] HOPFIELD, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 1982, vol. 79, no. 8, pp. 2554-2558. Disponible en: https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554

[3] HOPFIELD, J.J. Hopfield network. *Scholarpedia*, 2007, vol. 2, no. 5, pp. 1977. Disponible en: http://www.scholarpedia.org/article/Hopfield_network

[4] PAIK, J.K. y KATSAGGELOS, A.K. Image restoration using a modified Hopfield network. *IEEE Transactions on Image Processing*, 1992, vol. 1, no. 1, pp. 49-63. Disponible en: https://ieeexplore.ieee.org/abstract/document/128030

[5] CHAN, C.-K. y CHENG, L.M. The convergence properties of a clipped Hopfield network and its application in the design of keystream generator. *IEEE Transactions on Neural Networks*, 2001, vol. 12, no. 2, pp. 340-348. Disponible en: https://ieeexplore.ieee.org/abstract/document/914528

[6] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 2022, Vol. 25. Disponible en: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[7] LU, Jinzhu, TAN, Lijuan and JIANG, Huanyu. Review on Convolutional Neural Network (CNN) Applied to Plant Leaf Disease Classification. *Agriculture* [online]. 27 July 2021. Vol. 11, no. 8, p. 707. Disponible en: http://dx.doi.org/10.3390/agriculture11080707

[8] Junayed, Masum Shah and Jeny, Afsana Ahsan and Atik, Syeda Tanjila and Neehal, Nafis and Karim, Asif and Azam, Sami and Shanmugam, Bharanidharan. AcneNet - A Deep CNN Based Classification Approach for Acne Classes. *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, 2019, p. 203-208. Disponible en: https://ieeexplore.ieee.org/document/8850935

[9] GUO, Wei, YANG, Wen, ZHANG, Haijian and HUA, Guang. Geospatial Object Detection in High Resolution Satellite Images Based on Multi-Scale Convolutional Neural Network. *Remote Sensing* [online]. 18 January 2018. Vol. 10, no. 1, p. 131. Disponible en: http://dx.doi.org/10.3390/rs10010131

[10] Hacıefendioğlu, K., Başağa, H.B. & Demir, G. Automatic detection of earthquake-induced ground failure effects through Faster R-CNN deep learning-based object

detection using satellite images. *Nat Hazards* 03 October 2020. Vol. 105, p. 383-403. Disponible en: https://doi.org/10.1007/s11069-020-04315-y

[11] S. OwaisAli Chishti, S. Riaz, M. BilalZaib and M. Nauman. Self-Driving Cars Using CNN and Q-Learning. *IEEE 21st International Multi-Topic Conference (INMIC)*, 2018, p. 1-7. Disponible en: https://ieeexplore.ieee.org/abstract/document/8595684

[12] C. Tan, S. Lv, F. Dong and M. Takei Image Reconstruction Based on Convolutional Neural Network for Electrical Resistance Tomography. *IEEE Sensors Journal* 1 January 2019, vol. 19, no. 1, p. 196-204. Disponible en: https://ieeexplore.ieee.org/abstract/document/8493596

[13] Zhang, R., Isola, P., Efros, A.A. Colorful Image Colorization *Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science()*. 2016, vol. 9907. Disponible en: https://doi.org/10.1007/978-3-319-46487-9_40

[14] H. Lee and H. Kwon Going Deeper With Contextual CNN for Hyperspectral Image Classification. *IEEE Transactions on Image Processing*, October 2017, vol. 26, no. 10, p. 4843-4855. Disponible en: https://ieeexplore.ieee.org/abstract/document/7973178

[15] M. Zhang, W. Li and Q. Du. Diverse Region-Based CNN for Hyperspectral Image Classification. *IEEE Transactions on Image Processing*, October 2018, vol. 27, no. 6, p. 2623-2634. Disponible en: https://www.sciencedirect.com/science/article/pii/S1350449520300876

[16] Vaddi, Radhesyam and Manoharan, Prabukumar. Hyperspectral image classification using CNN with spectral and spatial features integration. *Infrared Physics & Technology*, 2020, vol. 107, p. 103-296. Disponible en: https://doi.org/10.1016/j.infrared.2020.103296

[17] P. V. Arun, K. M. Buddhiraju, A. Porwal and J. Chanussot. CNN-Based Super-Resolution of Hyperspectral Images. *IEEE Transactions on Geoscience and Remote Sensing*, Septiembre 2020, vol. 58, no. 9, p. 6106-6121. Disponible en: https://ieeexplore.ieee.org/abstract/document/9018371

[18] JongCheol Pyo, Hongtao Duan, Sangsoo Baek, Moon Sung Kim, Taegyun Jeon, Yong Sung Kwon, Hyuk Lee and Kyung Hwa Cho. A convolutional neural network regression for quantifying cyanobacteria using hyperspectral imagery. *Remote Sensing of Environment*, 2019, vol. 233, p. 111-350. Disponible en: https://www.sciencedirect.com/science/article/pii/S0034425719303694

[19] Al-Sarayreh, M., Reis, M.M., Yan, W.Q., Klette, R. A Sequential CNN Approach for Foreign Object Detection in Hyperspectral Images. *Computer Analysis of Images and Patterns*. Springer International Publishing, 2019, p. 271-283. Disponible en: https://doi.org/10.1007/978-3-030-29888-3_22

[20] Sun, D.-W. Hyperspectral Imaging for Food Quality Analysis and Control. Elsevier, Amsterdam (2010)

[21] M. Al-Sarayreh, M. M. Reis, W. Q. Yan and R. Klette. Deep Spectral-spatial Features of Snapshot Hyperspectral Images for Red-meat Classification. *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2018, p. 1-6 Disponible en: https://ieeexplore.ieee.org/abstract/document/8634783

[22] Hongfei Zhu, Lianhe Yang, Jiyue Gao, Mei Gao and Zhongzhi Han. Quantitative detection of Aflatoxin B1 by subpixel CNN regression. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 2022, vol. 268, p. 120-633 Disponible en: https://www.sciencedirect.com/science/article/pii/S1386142521012105

[23] de Sanctis, Maria Cristina, Filacchione, Gianrico, Capaccioni, Fabrizio, Ammannito, Eleonora, Capria, Maria Teresa, Coradini, Angioletta, Migliorini, Alessandra. Imaging Spectrometer for NEO Mission: Seta Instrument. *38th COSPAR Scientific Assembly*, January 2010, vol. 38, p. 8. Disponible en: https://ui.adsabs.harvard.edu/abs/2010cosp...38..684D

[24] Coradini, A., Capaccioni, F., Drossart, P. Virtis: An Imaging Spectrometer for the Rosetta Mission. *Space Sci Rev*, 2017, vol. 128, p. 529-559. Disponible en: https://doi.org/10.1007/s11214-006-9127-5

[25] R.A. Ruane and K.A.J. Doherty and R. Dorrepaal and B. Twomey and A. Gowen and J. Flanagan and D. de Faoite and K.T. Stanton. Hyperspectral imaging with unsupervised pattern recognition: A novel surface characterization technique for thermal control coatings. *Materials Letters*, 2019, vol. 254, p. 273-277. Disponible en: https://www.sciencedirect.com/science/article/pii/S0167577X19310717

[26] Sima Peyghambari and Yun Zhang. Hyperspectral remote sensing in lithological mapping, mineral exploration, and environmental geology: an updated review. *Journal of Applied Remote Sensing*, 2021, vol. 15, no. 3, p. 1-25. Disponible en: https://doi.org/10.1117/1.JRS.15.031501

[27] Cooper, B. L. and Salisbury, J. W. and Killen, R. M. and Potter, A. Midinfrared spectral features of rocks and their powders. *Journal of Geophysical Research: Planets*, 11 April 2002, vol. 107, p. 1-17. Disponible en: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2000JE001462

[28] Baowei Fei. Chapter 3.6 - Hyperspectral imaging in medical applications. *Hyperspectral Imaging*, 2019, vol. 32, p. 523-565. Disponible en: https://www.sciencedirect.com/science/article/pii/B9780444639776000213

[29] Tuan Vo-Dinh. A hyperspectral imaging system for in vivo optical diagnostics. *IEEE Engineering in Medicine and Biology Magazine*, vol. 23, no. 5, pp. 40-49. Disponible en: https://ieeexplore.ieee.org/abstract/document/1360407

[30] Liang, H. Advances in multispectral and hyperspectral imaging for archaeology and art conservation. *Appl. Phys.*, 2012, Vol. 106, p. 309-323. Disponible en: https://doi.org/10.1007/s00339-011-6689-1

[31] HUANG, Hui, LIU, Li and NGADI, Michael. Recent Developments in Hyperspectral Imaging for Assessment of Food Quality and Safety. *Sensors*, 22 April 2014, vol. 14, no. 4, p. 7248-7276. Disponible en: http://dx.doi.org/10.3390/s140407248

[32] A.A. Gowen and C.P. O'Donnell and P.J. Cullen and G. Downey and J.M. Frias Hyperspectral imaging – an emerging process analytical tool for food quality and safety control. *Trends in Food Science & Technology*, 2007, vol. 18, no. 12, p. 590-598. Disponible en: https://www.sciencedirect.com/science/article/pii/S0924224407002026

[33] Juan Xing and Cédric Bravo and Pál T. Jancsók and Herman Ramon and Josse De Baerdemaeker. Detecting Bruises on 'Golden Delicious' Apples using Hyperspectral Imaging with Multiple Wavebands. *Biosystems Engineering*, 2005, vol. 90, no. 1,

p.27-36. Disponible en: https://www.sciencedirect.com/science/article/pii/S1537511004001515

[34] Bosoon Park and Kurt C. Lawrence and William R. Windham and Douglas P. Smith. Performance of hyperspectral imaging system for poultry surface fecal contaminant detection. *Journal of Food Engineering*, 2006, vol. 75, no. 3, p. 340-348. Disponible en: https://www.sciencedirect.com/science/article/pii/S0260877405002578

[35] Renfu Lu and Yankun Peng. Hyperspectral Scattering for assessing Peach Fruit Firmness. *Biosystems Engineering*, 2006, vol. 93, no. 2, p. 161-171. Disponible en: https://www.sciencedirect.com/science/article/pii/S1537511005002552

[36] Jun Qiao and Michael O. Ngadi and Ning Wang and Claude Gariépy and Shiv.O. Prasher Pork quality and marbling level assessment using a hyperspectral imaging system. *Journal of Food Engineering*, 2007, vol. 83, no. 1, p. 10-16. Disponible en: https://www.sciencedirect.com/science/article/pii/S0260877407001185

APÉNDICE A
Objetivos de desarrollo sostenible

El día 25 de septiembre de 2015, mediante la Organización de Naciones Unidas, los líderes de todo el mundo decidieron adoptar un conjunto de objetivos diseñados para modificar o erradicar problemas sociales y culturales que azotan nuestra sociedad, como son la pobreza, desigualdad, el cambio climático o la contaminación. Dichos objetivos se plantearon con el fin de proteger el la Tierra y la humanidad, para hacer posible la convivencia entre nosotros y nuestro planeta, junto todos los seres vivos que lo pueblan. Todos los objetivos definidos en esta cumbre se construyeron con metas específicas a alcanzar antes de la década de 2030. A continuación se encuentran listados e indicando el grado de relación con el presente trabajo.

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

Reflexión sobre la relación del TFG con los ODS y con los ODS más relacionados.

ODS 8. Trabajo decente y crecimiento económico. Creo que este proyecto promueve la mejora de los niveles de productividad en la industria alimentaria (concretamente la del atún en lata), mediante la tecnología propuesta en este trabajo. Esta tecnología está diseñada para agilizar y aumentar la velocidad de la etapa de control de calidad de la cadena de producción de atún en lata y por tanto, la de la cadena de producción en si. Debido a que la industria alimentaria es uno de los sectores de gran valor creo que este objetivo se alinea con este trabajo.

> **8.2** Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra.

ODS 9. Industria, innovación e infraestructuras. También creo que la red convolucional creada en este trabajo permite modernizar y mejorar la industria, facilitando herramientas que mejorarán la producción, eficacia y capacidad tecnológica de esta. En este caso, mejorando la producción por ser una herramienta de control de calidad, mejorando la eficacia por permitir proveer el producto con la mejor calidad posible, y mejora la capacidad tecnológica por implementar nuevas tecnologías en la industria, permitiendo la investigación en este ámbito. Además, creo que este proyecto ayuda a mejorar la sostenibilidad de la producción del atún, ayudando a propiciar un consumo justo y racional.

> **9.4** De aquí a 2030, modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adopción de tecnologías y procesos industriales limpios y ambientalmente racionales, y logrando que todos los países tomen medidas de acuerdo con sus capacidades respectivas.

> **9.5** Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países, en particular los países en desarrollo, entre otras cosas fomentando la innovación y aumentando considerablemente, de aquí a 2030, el número de personas que trabajan en investigación y desarrollo por millón de habitantes y los gastos de los sectores público y privado en investigación y desarrollo.

ODS 12. Producción y consumo responsables. Como antes he escrito, el trabajo beneficia la correcta ejecución de la sección del control de calidad presente dentro de la cadena de producción de cualquier fábrica alimenticia de envasado de atún en lata, por tanto, creo que este proyecto ayuda en la tarea de generar una producción responsable, no solo con el medio ambiente, sino para con el consumidor, de forma que se reduzca el desperdicio de alimentos, tanto dentro de la cadena de producción, como por el consumidor al identificar correctamente qué muestras se encuentran en buen estado, permitiendo que el consumidor no deseche la lata de atún a causa de alguna imperfección en la textura.

> **12.2** De aquí a 2030, lograr la gestión sostenible y el uso eficiente de los recursos naturales.

> **12.3** De aquí a 2030, reducir a la mitad el desperdicio de alimentos per capita mundial en la venta al por menor y a nivel de los consumidores y

reducir las pérdidas de alimentos en las cadenas de producción y suministro, incluidas las pérdidas posteriores a la cosecha.

APÉNDICE B
Código de procesado de los datos

B.1 Carga inicial de datos

```
from osgeo import gdal
import matplotlib.pyplot as plt
import numpy as np
import os
import re
from datetime import datetime
```

Código B.1: Modulos empleados en la carga incial de los datos.

```
def get_data(img_path, console_output):
    """
    Abre un fichero .bil, dado su path, y obtiene los registros de la imagen
        hiperespectral.
    """
    bands = []
    data = []
    img = gdal.Open(img_path)
    for i in range(img.RasterCount):
        band = img.GetRasterBand(i+1)
        band_data = band.ReadAsArray()
        data.append(band_data)
        bands.append(list(img.GetMetadata_Dict().values())[i])
        clear_output()
        print(console_output,f"\n{i+1}/{img.RasterCount}")
    return data, bands

def binning(array):
    """
    Dada un array de profundidad 300, comprime la profundidad a 10 dividiendo
        los 30 en grupos de 30 y obtiendo la media de valores.
    """
    i_ = 0
    binned_array = []
    for i in range(29, 300, 30):
        binned_array.append(np.sum(array[i_:i],axis=0)/30)
        i_ = i
    return np.array(binned_array)

def cut(array, unique = False, file=None):
    """
    Dada una imagen con doble muestra divide por la mitad horizontalmente la
        imagen, centrando las que sean individuales (identificadas por unique=
        True).
    """
    cutted1 = None
```

```
    cutted2 = None
    if (unique and (file is not None)):
        if (file=="L14M142"):
            cutted1 = array[:,200:,200:890]
        elif (file == "L15M147"):
            cutted1 = array[:,120:790,150:850]
    else:
        cutted1 = array[:,:660,140:800]
        cutted2 = array[:,640:,140:800]
    return [cutted1, cutted2]
```

Código B.2: Métodos de obtención y procesado de los datos

```
def save_np_array(array, filename):
    np.save(filename, array, allow_pickle=False, fix_imports=False)

def load_numpy(filename):
    array = np.load(filename)
    return array
```

Código B.3: Métodos de guardado y cargado de las matrices.

B.2 Procesado de los datos

```
def resize(array):
    x =data.shape[1]; y = data.shape[2]
    new_array = zoom(array, (1, 256/x, 256/y))
    return new_array
```

Código B.4: Método de compresión de las imágenes a tamaño 256x256.

```
import os
import numpy as np
from scipy import stats

PATH = os.path.realpath('..')
file_path = "\\datos\\"
data_type = ".npz"
data = []
y0 = []
y1 = []
y2 = []
y3 = []
y4 = []
y5 = []

def get_filenames(path):
    files_d = []
    for root,dirs,files in os.walk(path):
        for filename in files:
            if not(("Correction" in filename) or (".hdr" in filename)):
                files_d.append(filename[:-4])
    return files_d

filenames = get_filenames(PATH+file_path)
for filename in filenames:
    f = np.load(PATH+file_path+filename+data_type)
    data.append(f['data'])
    y0.append(f['values'][0])
    y1.append(f['values'][1])
    y2.append(f['values'][2])
    y3.append(f['values'][3])
```

```
    y4.append(f['values'][4])
    y5.append(f['values'][5])
```

Código B.5: Carga de datos en los experimentos.

```
data = np.array(data)
y0 = np.array(y0)
y1 = np.array(y1)
y2 = np.array(y2)
y3 = np.array(y3)
y4 = np.array(y4)
y5 = np.array(y5)

norm_data = []
for im in data:
    i_max = np.max(im)
    im2 = im / i_max
    norm_data.append(im2)
norm_data = np.array(norm_data)

norm_y0 = stats.zscore(y0)
norm_y1 = stats.zscore(y1)
norm_y2 = stats.zscore(y2)
norm_y3 = stats.zscore(y3)
norm_y4 = stats.zscore(y4)
norm_y5 = stats.zscore(y5)

train_data = norm_data[:64]
test_data = norm_data[64:]
```

Código B.6: Normalizado de los datos en los experimentos.

APÉNDICE C
Código de configuración de las CNN

```
def alexnet():
    model = models.Sequential()
    model.add(layers.Conv2D(96, (11,11), padding='same', activation='relu',
        input_shape=(256,256,10)))
    model.add(layers.MaxPool2D((3,3)))

    model.add(layers.Conv2D(256, (5,5), padding='same', activation='relu',
        input_shape=(85,85,96)))
    model.add(layers.MaxPool2D((3,3)))

    model.add(layers.Conv2D(384, (3,3), padding='same', activation='relu',
        input_shape=(28,28,256)))
    model.add(layers.Conv2D(384, (3,3), padding='same', activation='relu',
        input_shape=(28,28,384)))
    model.add(layers.Conv2D(256, (3,3), padding='same', activation='relu',
        input_shape=(28,28,384)))
    model.add(layers.MaxPool2D((3,3)))

    model.add(layers.Conv2D(256, (3,3), padding='same', activation='relu',
        input_shape=(9,9,256)))
    model.add(layers.MaxPool2D((9,9)))

    model.add(layers.Dense(4096))
    model.add(layers.Dense(4096))
    model.add(layers.Dense(1))
    return model
```

Código C.1: Arquitectura AlexNet.

```
def shallownet():
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3,3), padding='same', activation='relu',
        input_shape=(256,256,10)))
    model.add(layers.Dense(1))
    return model
```

Código C.2: Arquitectura ShallowNet.

```
def lenet():
    model = models.Sequential()
    model.add(layers.Conv2D(6, (3,3), padding='same', activation='relu',
        input_shape=(256,256,10)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Conv2D(16, (5,5), padding='same', activation='relu',
        input_shape=(128,128,6)))
```

```
    model.add(layers.MaxPool2D((64,64)))

    model.add(layers.Dense(120))
    model.add(layers.Dense(84))
    model.add(layers.Dense(1))
    return model
```

Código C.3: Arquitectura LeNet.

```
def generate_cnn():
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3,3), padding='same', activation='relu',
        input_shape=(256,256,10)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Conv2D(64, (3,3), padding='same', activation='relu',
        input_shape=(128,128,32)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Conv2D(128, (3,3), padding='same', activation='relu',
        input_shape=(64,64,64)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Conv2D(256, (3,3), padding='same', activation='relu',
        input_shape=(32,32,128)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Conv2D(512, (3,3), padding='same', activation='relu',
        input_shape=(16,16,256)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Conv2D(512, (3,3), padding='same', activation='relu',
        input_shape=(8,8,512)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Conv2D(512, (3,3), padding='same', activation='relu',
        input_shape=(4,4,512)))
    model.add(layers.MaxPool2D((2,2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))
    return model
```

Código C.4: Arquitectura propia.