UNIVERSIDAD DE GRANADA

Trabajo de Fin de Grado

Grado en Ingeniería Informática y
Administración y Dirección de Empresas

Identificación de micro y macro organismos usando Redes Neuronales Convolucionales

---

Subtítulo

Autor
Jacobo Casado de Gracia

Director
Siham Tabik

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2022

Índice general

1. Introducción **3**
- 1.1. Descripción del problema . . . 3
- 1.2. Objetivos . . . 4
- 1.3. Planificación . . . 6
- 1.4. Estructura de la memoria . . . 6
  - 1.4.1. Fundamentos teóricos . . . 6
  - 1.4.2. Estado del arte . . . 6

2. Fundamentos teóricos **7**
- 2.1. Machine Learning . . . 7
- 2.2. Deep Learning . . . 7
  - 2.2.1. ¿Por qué surgió? . . . 7
- 2.3. Redes Neuronales Convolucionales para la clasificación . . . 10
  - 2.3.1. Extracción de características . . . 10
  - 2.3.2. Clasificación . . . 15
  - 2.3.3. Visualizando las *convnets* . . . 16
- 2.4. Entrenamiento de la red . . . 16
  - 2.4.1. Función de pérdida . . . 16
  - 2.4.2. Backpropagation . . . 16
- 2.5. Transfer Learning y Fine Tuning . . . 16
  - 2.5.1. Transfer learning (extracción de características fija) . . . 16
  - 2.5.2. Fine-Tuning . . . 16
  - 2.5.3. Reentrenamiento completo . . . 16

3. Estado del arte **17**
- 3.1. Clasificación de diatomas . . . 17
  - 3.1.1. Extracción manual de características . . . 17
  - 3.1.2. Extracción automática de características . . . 17
- 3.2. Redes Neuronales Convolucionales . . . 17
  - 3.2.1. EfficientNet . . . 17
  - 3.2.2. EfficientNetV2 . . . 17
- 3.3. *Frameworks* . . . 17

4. Creación del *dataset* **18**
- 4.1. Obtención de los datos . . . 18
- 4.2. Análisis del conjunto de datos . . . 18
- 4.3. Preprocesado . . . 18
  - 4.3.1. Recorte y ajuste . . . 18
  - 4.3.2. Data Augmentation . . . 18
- 4.4. Balanceo de datos . . . 18
- 4.5. Creación del *dataframe* . . . 18

5. Diseño e implementación **19**
5.1. Diseño de modelo *ad-hoc* 19
5.2. Extracción fija de características 19
5.3. Fine-tuning 19

6. Análisis de resultados **20**
6.1. Comparativa de arquitecturas 20
6.2. Visualización del aprendizaje 20

1 | Introducción

1.1. Descripción del problema

Los diatomas son un grupo de algas unicelulares, siendo el grupo más común de microorganismos en los hátitats acuáticos y presentes en gran cantidad de ellos. Son uno de los principales contribuidores en la cadena alimenticia de los ecosistemas acuáticos, formando parte de la base de ésta.

Debido a que los diatomas se adaptan al ecosistema y modifican su forma y textura dependiendo de éste, actualmente se usan como uno de los indicadores más precisos para medir la calidad del agua en estudios ambientales y, por tanto, para estudios más complejos como los relacionados con el cambio climático y su evolución. [1, 2]

Los diatomas tienen varias ventajas sobre otros bioindicadores que los hacen los bioindicadores ideales: es relativamente fácil obtener muestras grandes de ellos sin impactar en el ecosistema en el proceso, tienen una respuesta rápida a la variación de las condiciones del ecosistema y son muy sensibles a cambios del ambiente que otros bioindicadores ni siquiera plasman (sobre todo los cambios en las variables químicas del agua) [3].

Traducionalmente, al igual que en muchos otros campos, la tarea de identificación y clasificación de diatomas de muestras tomadas en microscopios es tarea de los biólogos. Estos profesionales generalmente buscan características **morfométricas** (longitud, anchura y forma del diatoma, sobre todo) así como detalles internos del organismo, como la densidad de estrías (**información sobre la textura**). Una vez se han medido estas características principales (*key features*), la manera de clasificarlos es compararlos respecto a diatomas previamente clasificados, pero hay varios retos en el camino:

1. Se requieren expertos en la taxonomía de los diatomas para resolver este problema, ya que algunas de las *key-features* que se toman para identificar al diatoma requieren conocimiento amplio en el tema, incluyendo medidas no cuantitativas.

2. Los expertos deben de analizar continuamente estas *key-features* cuando quieren clasificar una nueva diatomea, por lo que tienen que realizar un análisis de características distintivas cada vez que se encuentran con una nueva clase.

3. El análisis manual de las imágenes es un proceso muy largo, y para llegar a una conclusión válida sobre la calidad del agua se necesitan alrededor de 400 imágenes por estudio. [4]

4. Los diatomas tienen similitudes inter-especies y disimilitudes intra-especies que hacen esta tarea más árdua aún. Como se comenta en [5], actualmente hay tasadas alrededor de 10.000 especies, pero se estima que el número total de especies es mayor a 200.000, por lo que el tamaño del problema hace que la tarea de clasificación sea aún más complicada.

5. Para más complejidad, un diatoma cambia de forma durante su ciclo de vida, por lo que sufre cierto *polimorfismo* que hace que sea mucho más difícil clasificarlo con métodos manuales. [6]

En este proyecto se trabaja en un marco de investigación con la Universidad de Granada, en la creación de un sistema de de clasificación de diatomas con un enfoque basado en el aprendizaje automático (*machine learning*), proponiéndose una serie de arquitecturas conocidas en el mundo de la *visión por computador* como Redes Neuronales Convolucionales (CNN) para resolver este problema de clasificación, estudiando distintas arquitecturas y realizando una comparativa a modo de análisis para ver de cuál de ellas es la que mejor se adapta a este problema en cuestión y obtiene mejores resultados.

1.2. Objetivos

El objetivo de este trabajo no es sólo dar un enfoque diferente para resolver este problema de clasificación usando las técnicas de *aprendizaje automático*, sino paliar los problemas surgidos del enfoque clásico de clasificación de diatomas, automatizando la tarea sin necesidad de recurrir a expertos en el ámbito, reducir el error humano de estos problemas de clasificación y ahorrar un tiempo en la tarea de búsqueda de características y de clasificación.
En primer lugar, se aporta un esquema del (*workflow*, y posteriormente se detallarán los objetivos de una manera más específica.

El **objetivo global** de este Trabajo de Fin de Grado es:
Diseñar un sistema automático para la clasificación de Diatomeas en imágenes microscópicas.
Para lograr ese objetivo, es necesario desglosar y trabajar en cumplir los objetivos parciales, vistos en el esquema.

A continuación, se desglosan estos objetivos para explicar la utilidad y la importancia de cada uno de ellos.

- **Subobjetivo 1. Construir una base de datos de Diatomeas para la clasificación de Diatomeas usando Redes Neuronales Convolucionales.**
  Antes de diseñar una red neuronal convolucional que sea capaz de clasificar Diatomeas, es necesario construir una base de datos de Diatomeas lo suficientemente grande como para poder hacer que la red sea capaz de extraer las características principales de cada clase de Diatomea y, además de eso, tenemos de tener en cuenta de que nuestro conjunto de datos debe de tener un tamaño suficiente, para que la red sea capaz de **generalizar**, es decir, dar una correcta clasificación con nuevas imágenes, y que cada imagen con la que entrenemos nuestra red sea representativa del problema y no introduzca ruido a la base de datos.

  No podemos olvidar que estamos en un problema de *machine learning*, y, por tanto, la solución al problema se plantea mediante el aprendizaje, donde los **datos** (en nuestro caso, **imágenes**) forman un rol muy importante. En el apartado de fundamentos teóricos hablaremos del enfoque del *machine learning* para solucionar este problema y todo lo necesario para entender qué se está haciendo de forma teórica.

- **Subobjetivo 2. Desarrollar métodos de preprocesado para mejorar la generalización.** Dado que la base las imágenes para que sean datos relevantes para el proceso de aprendizaje.
  Una vez extraídas las imágenes, que serán los datos y la única fuente de información con la que el algoritmo entrenará, es importante preprocesar las imágenes para que sean entradas correctas a nuestra red, aplicar aumento de datos a las imágenes para que la red sea, de cierto modo, invariante a cambios en la intensidad, rotación, o escala de nuestras imágenes...

- **Subobjetivo 3. Analizar el uso de Redes Neuronales Convolucionales para la clasificación de Diatomas**.
  En este apartado se usarán las Redes Neuronales Convolucionales como extractores de características y clasificadores de las imágenes de Diatomas.

  Para analizar el uso de estas redes, se comenzará utilizando una Red Neuronal Convolucional *ad-hoc* (es decir, diseñada desde cero), y posteriormente se usarán arquitecturas ya prediseñadas para esta tarea que han logrado resultados satisfactorios en bases de datos de millones de imágenes, como DenseNet o EfficientNet, presentes actualmente en el estado del arte del *deep learning*.

  Estas últimas arquitecturas se podrán implementar de dos maneras; o bien como extractores de características, sin entrenarlas para cubrir nuestro problema (*transfer learning*), o se podrán ajustar a nuestros datos para extraer características del problema en cuestión (*fine tuning*); se comprobará cual de estas dos alternativas es mejor de cara a nuestro problema de clasificación.

  Para cada una de estas arquitecturas, se evaluará su rendimiento a la hora de solucionar este problema de clasificación, usando, para ello, diferentes métricas que nos servirán para evaluarlas individualmente y compararlas entre ellas. De esta manera habremos cumplido satifactoriamente el objetivo.

- **Subobjetivo 4. Implementar el algoritmo de clasificación obtenido en una interfaz gráfica.**. Una vez obtenido un modelo que clasifique correctamente los datos (posteriormente comentaremos a qué nos referimos con cuando decimos correctamente), se creará una plataforma web donde se automatice el proceso de clasificación de Diatomas, bajo una interfaz gráfica, por ejemplo, una página *web*. El usuario, tras adjuntar una imagen de una Diatomea, podrá ver de qué clase es ese Diatomea en cuestión, consiguiendo, finalmente, haber automatizado el proceso de clasificación.

Cada uno de estos subobjetivos es igual de relevante para cumplir el objetivo final, por lo que se dispondrá de un apartado en este trabajo con el proceso realizado para cubrir cada uno de ellos. En el siguiente apartado, el apartado de planificación, se establece un marco temporal donde se siguen todos estos objetivos, indicando también las dependencias entre cada uno de ellos.

1.3. Planificación

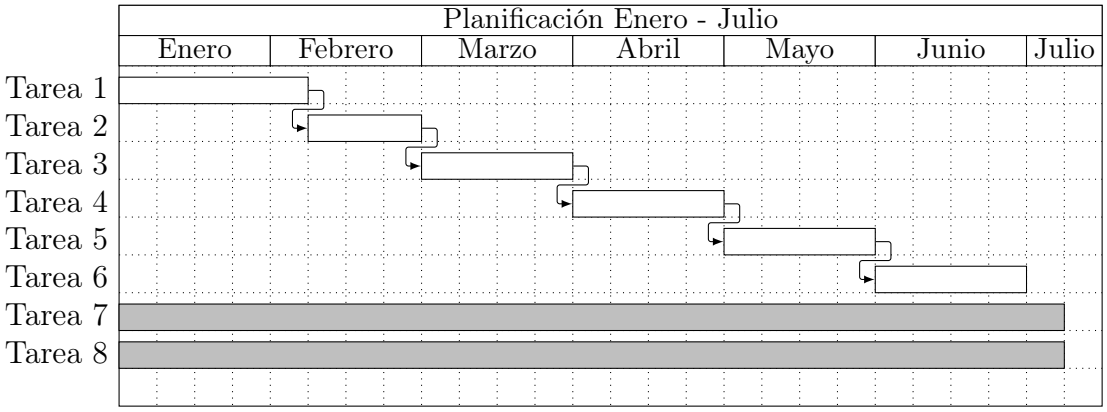

Figura 1.1: Diagrama de Gantt estructurado en semanas para la planificación del trabajo.
Fuente: Elaboración propia

1.4. Estructura de la memoria

La memoria está estructurada en varios bloques, que se explicarán a continuación:

1.4.1. Fundamentos teóricos

En este apartado se explican todos los fundamentos teóricos necesarios para entender el problema en cuestión y toda terminología que se usará para llevar a cabo el proyecto. En primer lugar, se comenzará comentando los conceptos básicos del *machine learning*, y posteriormente se avanzarán con conceptos más especializados, como el *deep learning* o las Redes Neuronales Convolucionales, de manera que cualquier persona que esté interesada en saber qué se esta haciendo pueda entender poco a poco el proceso.

Se recomienda la lectura de manera secuencial para el completo entendimiento de lo que se quiere hacer en la práctica, ya que todos estos términos están relacionados entre sí.

1.4.2. Estado del arte

En esta sección se hablará en primer lugar del proceso que se sigue actualmente para la clasificación de Diatomeas: qué se hace y cómo. Por otro lado, se comentarán las arquitecturas que mejores resultados están obteniendo para la tarea de Clasificación de Imágenes en cuanto a Redes Neuronales Convolucionales se refiere, viendo las últimas mejoras en este apartado.

2 | Fundamentos teóricos

2.1. Machine Learning

El *Machine Learning* es una rama de las Ciencias de la Computación y a su vez de la Inteligencia Artificial. Pese a tener muchas definiciones y ser un concepto relativamente abstracto, una definición que considero bastante apropiada es llamar Machine Learning al desarrollo de sistemas capaces de **cambiar su comportamiento de forma autónoma en base una experiencia**, consiguiendo la capacidad de aprender a partir de unos datos dados. [7]

Con el término aprender nos referimos a desempeñar una tarea de forma correcta, es decir, con un mínimo error durante su ejecución; se podría hacer una analogía con el aprendizaje de un niño durante los primeros años de vida; al principio hay que indicarle varias veces que el objeto A es una cuchara y el objeto B un tenedor pero llega un momento donde será capaz de aprender qué diferencia a la cuchara y al tenedor para poder clasificarlos correctamente.

Este último ejemplo es un ejemplo del **aprendizaje supervisado**, uno de los tipos de aprendizaje más comunes en la práctica del *machine learning* y en concreto aquel con el que trabajaremos: nuestro sistema recibirá como entradas unos **datos (en nuestro caso, imágenes de Diatomeas)** y una etiqueta asociada a cada imagen, indicando **la especie**.

La idea a seguir es que tras un proceso de aprendizaje (se verá cómo más tarde, en particular para la arquitectura usada en este problema), el sistema sea capaz de clasificar **nuevas imágenes de Diatomeas** con las que no ha sido entrenada y que sea capaz de predecir la **la especie a la que pertenece**.

2.2. Deep Learning

Si bien hemos entendido qué es el *machine learning*, el siguiente paso es hablar de uno de los subcampos de este: el *deep learning*, o aprendizaje profundo.

El *deep learning* es una vertiente del *machine learning* que pone el énfasis en aprender capas sucesivas de representaciones de los datos cada vez más significativas; a veces se malinterpreta este concepto pensando que el entendimiento es más profundo pero no es así. Lo único que varía es la idea de **crear capas sucesivas de representaciones**, y todas ellas se aprenden de forma *automática* a partir del entrenamiento, incluida la capa dedicada a clasificar los datos (**el clasificador**), que va al final de la red.

2.2.1. ¿Por qué surgió?

Las técnicas convencionales de *machine learning* necesitan un paso previo antes de su aplicación, y es la **extracción de características**, puesto que lo único que se entrena es

el **clasificador** que usa las características durante el entrenamiento para delimitar una frontera de decisión entre una clase u otra.

Es decir, para saber diferenciar entre un objeto de la clase A y otro de la clase B es necesario **delimitar qué características los diferencian**, y usarlas como datos de entrada para nuestra red. De esta manera, será capaz de establecer una clasificación usando estas variables.

El problema surge cuando tenemos casos como el que estamos estudiando, donde se requiere a un experto en el dominio para delimitar las características principales de cada clase y, por tanto, la selección de características se convierte en un problema más.

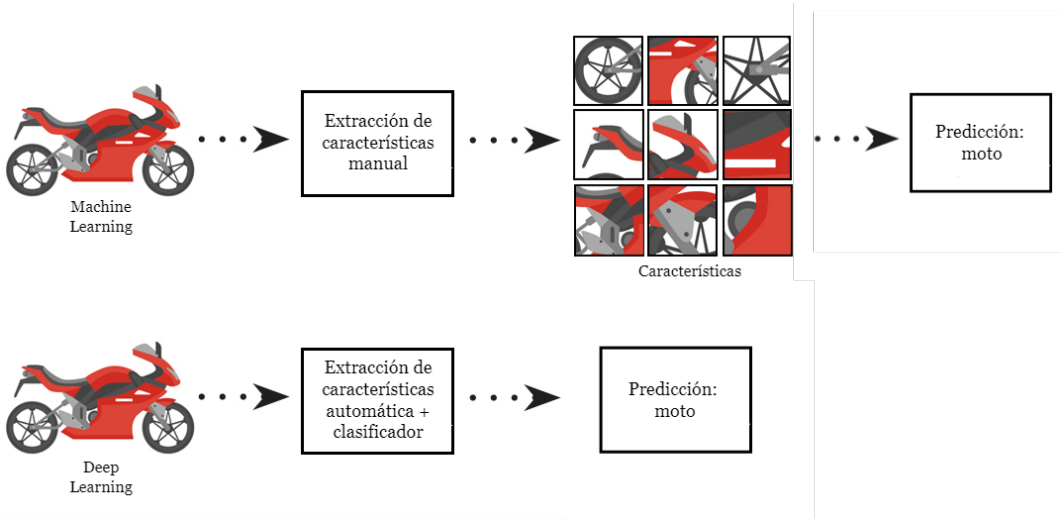

Figura 2.1: Comparativa del proceso de *machine learning* versus el *deep learning*. Fuente: Elaboración propia a partir de [8].

En el *deep learning*, estas características se aprenden de manera automática mediante modelos llamados *redes neuronales*, que son, básicamente, estructuras de capas apiladas una encima de otra, donde cada capa surge a partir de una transformación de la capa anterior.
Lo interesante es que **estas transformaciones de datos se aprenden por la exposición a ejemplos**, sin tener que ser definidas previamente, por lo que, de cierta manera, estamos encontrando **patrones** que poseen los datos y que definirán las características fundamentales de cada tipo de dato.
Al trabajar con datos, es necesario disponer de una gran cantidad de ellos para poder aprender no sólo el cómo clasificarlos, sino extraer las características fundamentales de cada tipo de dato, por lo que los datos juegan un gran rol en el *deep learning*.

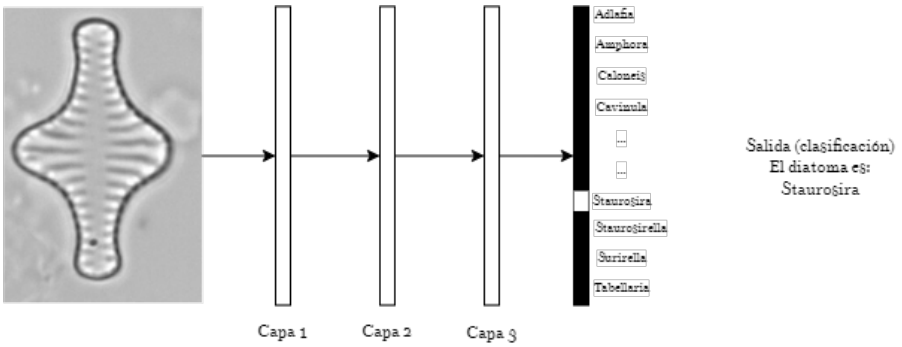

Figura 2.2: Ejemplo gráfico de una red neuronal usada para la clasificación de Diatomas. La salida es la clase estimada por la red.

La idea plasmada en la figura 2.2 es que, para dar la predicción final, la red transforma el dato de entrada (imagen) en representaciones que son cada vez más diferentes de la imagen original y más informativas respecto al resultado final.

2.3. Redes Neuronales Convolucionales para la clasificación

Las Redes Neuronales Convolucionales (llamadas a partir de ahora **CNN**), son un tipo de redes neuronales muy usadas para las tareas de reconocimiento y clasificación de imágenes, debido a que son capaces de detectar una gran cantidad de características de una imagen mediante decenas o cientos de capas ocultas. Cada capa oculta aumenta la complejidad de características de la imagen aprendida.

Por ejemplo, la primera capa oculta podría aprender el cómo detectar bordes o esquinas, mientras que la segunda aprende a detectar formas más complejas propias del objeto que se intenta reconocer.

La estructura fundamental de estas redes se compone de combinaciones de bloques que, conectados entre sí, realizan el proceso de extracción de características y clasificación. Cada bloque se genera de una manera distinta y tiene una función específica en el proceso de aprendizaje, aunque, a alto nivel, el primer conjunto de capas estarán dedicadas a la **extracción de características** y las capas finales serán las que se ocupen de la clasificación de la imagen.
En la parte de extracción de características, como su nombre indica, se extraerán las características de los datos de entrada o de **entrenamiento**, formando los mapas de características (o *feature maps*), los cuales generalmente aumentan en número y cantidad, pero disminuyen en tamaño, obteniendo al final características de alto nivel y mapas de características muy pequeños.
Cuando estos mapas de características han sido extraídos, una red neuronal *fully connected*, es decir, donde todos sus elementos tienen cierto enlace entre sí, se encarga de separar y clasificar cada característica.

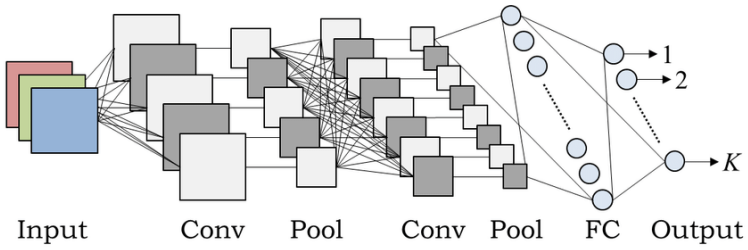

Figura 2.3: Ejemplo gráfico de una CNN, donde se diferencia entre extracción de características y clasificación.
Fuente: [9]

Una vez viendo el esquema de la red, tomando como ejemplo la figura , es importante entender además que hay que tener en cuenta el cómo **las capas se entrenan** y modifican sus valores para adaptarse al problema.
Por tanto, desglosaremos los conceptos teóricos de las CNN en tres partes: **extracción de características, clasificación y entrenamiento.**

2.3.1. Extracción de características

En esta parte se comentan los fundamentos teóricos principales que conforman la parte de extracción de características realizadas por las CNN.

Convolución

Las CNN son capaces de aplicar una gran cantidad de transformaciones a los datos gracias al operador de **convolución**. La convolución es el operador usado para transformar un bloque de entrada (ya sea el bloque de datos inicial u otro creado a lo largo del proceso) en otro bloque de salida con otra estructura.

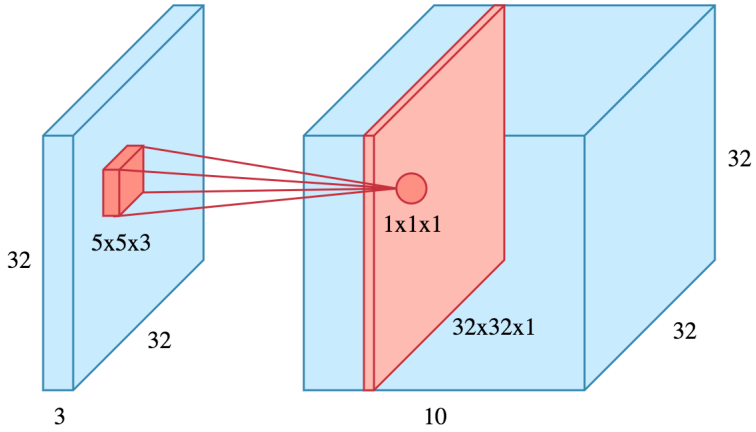

Figura 2.4: Generación de un nuevo mapa de características (**rojo**) a partir de una convolución con un kernel 2D.

Como se aprecia en la figura 2.3, la operación de convolución necesita dos cosas: un bloque de entrada que será transformado (o convolucionado), bloque que llamaremos a partir de ahora **mapa de características** y un **kernel**, que será lo que aplicaremos sobre el bloque de entrada para transformarlo al bloque de salida, que será un mapa de características resultante.

Lo principal que tenemos que entender de la operación de convolución, sin entrar en conceptos matemáticos profundos, es que el **kernel** se **desliza o desplaza** sobre el bloque de características de entrada **completo**, elemento a elemento, y de manera horizontal y vertical, y a lo largo de toda su profundidad, para generar un mapa de características, y que el **kernel** no opera con un solo elemento del bloque de entrada por cada pasada, sino que toma varios de los elementos de este bloque para generar **uno** del siguiente bloque. Es decir, si nos fijamos en la figura 2.3, el **kernel** de tamaño 5x5x3 usaría 25 elementos del mapa de características A para generar un elemento del mapa de características B. Como se ha especificado que la profundidad del kernel es 3 (5x5x**3**), se utilizan 75 elementos, 25 por cada capa del bloque A, para generar un solo elemento del siguiente bloque B.

Otra cosa a destacar es que no todos los datos que se utilizan y combinan para crear un dato del siguiente mapa de características **tienen la misma importancia**. La importancia que cada dato tiene varía en función del **kernel**, que **pondera** cada dato con un cierto valor (generalmente, entre 0 y 1) de manera que cada dato tiene su importancia relativa en la operación de convolución.

Esto hace que los mapas de características varíen dependiendo del **kernel** aplicado, tanto en dimensión como en el valor. Como ya hemos visto matemáticamente, el bloque de características de entrada se transforma en nuevos mapas de características y por tanto, otra representación de los datos diferente; en el proceso de aprendizaje se buscará el conjunto de valores **para los pesos de todas las capas de la red** de manera que la red asigne correctamente las etiquetas a los ejemplos que le hemos proporcionado (es la

única información de la que dispone, los datos). Lo que el ingeniero deberá de especificar son otros parámetros como el tamaño del kernel o el número de mapas de características generados en el siguiente bloque.
A más mapas de características queramos generar, más bloques convolucionales habrá que añadir en la arquitectura, y más pesos (**parámetros**) por aprender.

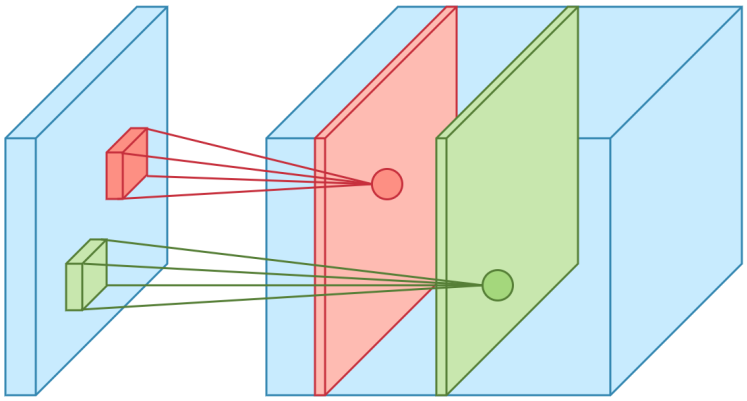

Figura 2.5: Ejemplo anterior pero mostrando el uso de otro **kernel** para generar un mapa de características más.

En la figura 2.4 se puede ver que usa más de un **kernel** para generar el nuevo bloque de características debido a que se ha especificado más de un mapa de características como salida, y por tanto, diferentes representaciones del bloque anterior. Aquí se ve la clara distinción entre **mapa de características** (parte roja y verde del bloque de características) y **el bloque de características al completo** (azul).

Pooling

El *pooling* es una operación que se suele aplicar a bloques de características cuando consideramos que en ese bloque hay información redundante entre los píxeles, es decir, que de cierta manera, los píxeles del entorno tienen bastante información que comparten en común y se podrían resumir.[en] uno solo, **reduciendo el número de parámetros entre bloques** (ya que el bloque se reduce en dimensiones) y aportando cierta **invarianza a las traslaciones**, demanera que las características de un objeto no tengan que estar en un lugar determinado, sino en cualquier lugar de la imagen. [10]

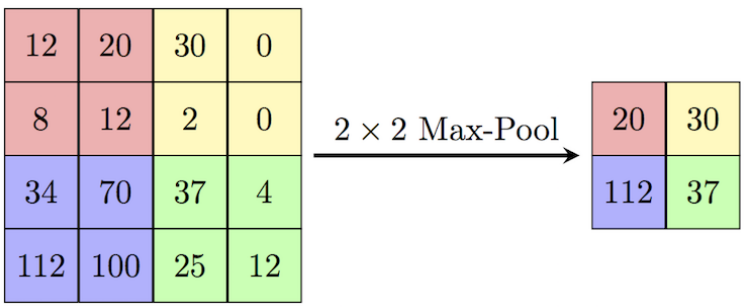

Figura 2.6: Ejemplo de la función *maxpooling* de tamaño 2 (2x2) en una matriz de datos. Como podemos ver, el tamaño del bloque se reduce considerablemente.

En la figura podemos ver un ejemplo de *max pooling*, una función de *pooling* que toma el valor máximo de un cuadrado de lado l; otras funciones conocidas son el L_p *pooling, mixed pooling, Stochastic Pooling...* (más en [11]).

Activación

La función de activación se aplica a todos los elementos de mapas de características, y es aquella que permite principalmente el aprendizaje, es decir, el ajuste de los pesos.

La función de activación decide si una neurona debería ser activada o no (si vale 0 o más de cero), y por tanto, **delimita** los valores de los elementos del bloque convolucional.

Para entender mejor lo que hace, tomemos un ejemplo usando la función de activación más conocida y sencilla, la **ReLU**. Esta función transforma la entrada tal que así:

$$x = \begin{cases} \text{x} & \text{si x} \geq 0 \\ 0 & \text{en otro caso} \end{cases}$$

Es decir, transforma todas las entradas en valores no negativos. Si algún valor es negativo, lo transforma a cero, y el resto de valores positivos mantienen su valor. Esta simple activación hace que el proceso de aprendizaje tenga efecto y se puedan aprender patrones en las imágenes.
Un ejemplo gráfico aplicado a un bloque convolucional es el siguiente, mostrado en la figura 2.2:

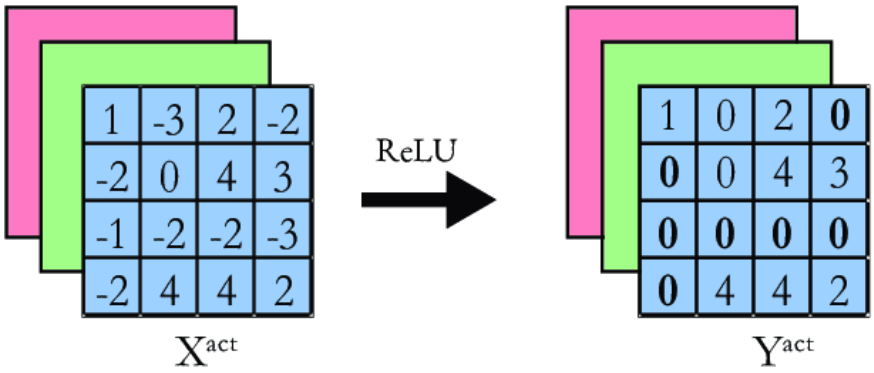

Figura 2.7: Ejemplo gráfico de una función de activación aplicada a un mapa de características.
La salida es otro mapa con la función ya aplicada. Fuente: [12]

Si no hubiese función de activación, en el proceso de aprendizaje estaríamos **únicamente transformando la imagen de entrada una y otra vez**, sin **cribar o filtrar** información de un bloque a otro. Aplicando una función de activación, es decir, funciones **no lineales**, evitamos esto, haciendo que cada bloque tenga información distinta al anterior.

Se ha presentado la función de activación **ReLU**, pero hay muchas otras funciones no lineales de activación como la **sigmoide, hiperbólica o variantes de la ReLU, como la *Leaky ReLU*** (más en[11]); lo importante aquí es entender hasta ahora que la convolución transforma los datos y la función de activación nos criba parte de ellos, y que **en la gran mayoría de los casos**, cada bloque convolucional que se genera va acompañado de una función de activación, antes de volver a convolucionar.

Más tarde veremos la función de activación *softmax*, utilizada en nuestro problema para poder transformar la salida de la red neuronal en una predicción sencilla y fácil de entender.

Regularización

Para poder definir bien este concepto, necesitamos definir bien previamente el concepto de *overfitting*. El *overfitting* es un problema muy común en las CNN, debido a la cantidad de pámetros que estos modelos tienen, y ocurre cuando estas arquitecturas son capaces de ajustar bien los datos de entrenamiento, pero son muy pobres a la hora de predecir sobre datos nuevos, debido a que los **pesos** de la red se han sobreajustado para adaptarse a los datos de entrenamiento de una manera demasiado precisa. **En este momento, ya no estamos aprendiendo, sino memorizando literalmente los datos que tenemos** y la capacidad de predicción es mucho peor, debido a que los datos nuevos difieren de aquellos con los que hemos entrenado.

Podemos pensar que, cuanta mayor es la complejidad de nuestro modelo (por ejemplo, un polinomio de grado 4 en comparación de un polinomio de grado 15), ajustaremos mejor nuestros datos ya que tenemos más grados de libertad, pero debido a sobreajustar estos datos el polinomio no representa la función desconocida que intentamos aproximar y que nunca conoceremos **(y que por tanto puede ser más simple que la de nuestro modelo)**.

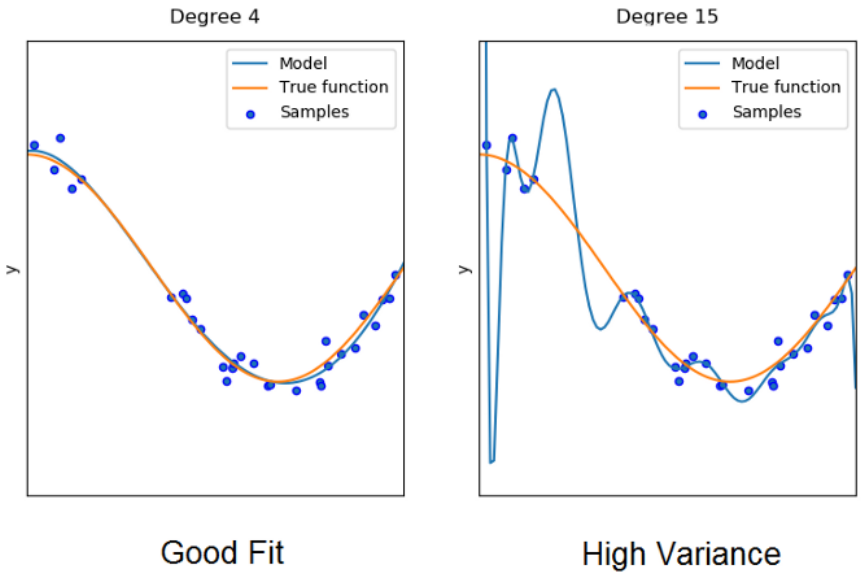

Figura 2.8: Ejemplo de sobreajuste o *overfitting* con un polinomio de grado 4 en comparación a uno de grado 15. Fuente:

En el gráfico anterior, pese a que se dibuja la función objetivo, en la realidad no será así, y por tanto la elección de la complejidad de nuestro modelo es un factor muy importante. Es ahí donde entra la regularización.
En realidad, el concepto de **regularización** es simple: **penalizar** la complejidad de nuestro modelo.

2.3.2. Clasificación

Capa totalmente conectada o *fully connected*

Como se ha visto en todas las operaciones anteriores, las neuronas de cada bloque convolucional no están totalmente conectadas, sino que únicamente la salida de un bloque depende del bloque anterior, pero no de toda la red.

En esta capa, al igual que en la convolucional, existe una ecuación para la salida de neuronas de manera que **cada neurona depende matemáticamente de las demás, estando, como el nombre de esta capa indica, totalmente conectadas** (más en [10]).

Si bien esta capa se puede aplicar en cualquier lugar de la red, su lugar más apropiado es al final de ella, debido a que es muy útil en las tareas de clasificación, debido a la función de activación *softmax*, que, aplicada sobre esta capa, permite obtener una clasificación final de nuestra red adaptada a nuestro problema.

Función de activación *softmax*

La función de activación *softmax* es comúnmente usada en problemas de clasificación múltiple, es decir, problemas donde hay más de dos clases.

Es útil porque transforma un vector de entrada en un vector de probabilidades, donde todos los elementos tienen un valor entre 0 y 1 y, además, la suma de todos los elementos del vector es igual a 1. Esto se consigue aplicando a i del vector de entrada se transforma utilizando la siguiente fórmula:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K$$

Viéndolo de otra forma, se está midiendo la **contribución** del valor de cada elemento del vector al total de éste en tanto por uno. Un resultado de un vector tras aplicar softmax puede ser el siguiente:

Figura 2.9: Softmax Fuente:
https://towardsdatascience.com/deep-learning-concepts-part-1-ea0b14b234c8

Si añadimos un vector de dimensión i al final de nuestra red, donde i es exactamente el número de clases que nuestro problema tiene y queremos diferenciar en la clasificación, tras aplicar esta función tendremos un excelente clasificador, ya que nos devolverá la probabilidad de la imagen de pertenecer a esa clase i.
Por tanto, la red etiquetará a la imagen como clase i, siendo i la posición del vector con mayor probabilidad.

2.3.3. Visualizando las *convnets*

2.4. Entrenamiento de la red

2.4.1. Función de pérdida

2.4.2. Backpropagation

2.5. Transfer Learning y Fine Tuning

2.5.1. Transfer learning (extracción de características fija)

2.5.2. Fine-Tuning

2.5.3. Reentrenamiento completo

3 | Estado del arte

3.1. Clasificación de diatomas

3.1.1. Extracción manual de características

3.1.2. Extracción automática de características

3.2. Redes Neuronales Convolucionales

3.2.1. EfficientNet

3.2.2. EfficientNetV2

3.3. *Frameworks*

4 | Creación del *dataset*

4.1. Obtención de los datos

4.2. Análisis del conjunto de datos

4.3. Preprocesado

4.3.1. Recorte y ajuste

4.3.2. Data Augmentation

4.4. Balanceo de datos

4.5. Creación del *dataframe*

5 | Diseño e implementación

5.1. Diseño de modelo *ad-hoc*

5.2. Extracción fija de características

5.3. Fine-tuning

6 | Análisis de resultados

6.1. Comparativa de arquitecturas

6.2. Visualización del aprendizaje

Bibliografía

[1] Eduardo Lobo y col. «Diatoms as Bioindicators in Rivers». En: jun. de 2016, págs. 245-271. ISBN: 978-3-319-31983-4. DOI: 10.1007/978-3-319-31984-1_11.

[2] Luc Ector y Frédéric Rimet. «Using bioindicators to assess rivers in Europe: An overview». En: ene. de 2005, págs. 7-19. DOI: 10.1007/3-540-26894-4_2.

[3] Saúl Blanco y col. «Comparison of Biotic Indices for Water Quality Diagnosis in the Duero Basin (Spain)». En: *Archiv für Hydrobiologie. Supplementband. Large rivers* 17 (ene. de 2007), págs. 267-286.

[4] British Standards Institute Staff. *Water Quality. Guidance Standard for the Identification, Enumeration and Interpretation of Benthic Diatom Samples from Running Waters*. B S I Standards, 2004. ISBN: 9780580442476. URL: https://books.google.es/books?id=un5SPQAACAAJ.

[5] David G. Mann. «The species concept in diatoms». En: *Phycologia* 38 (1999), págs. 437-495.

[6] Carlos Sánchez-Bueno, Gabriel Cristóbal y Gloria Bueno. «Diatom identification including life cycle stages through morphological and texture descriptors». En: (2019). DOI: 10.7717/peerj.6770.

[7] Nils J. Nilsson. *Introduction to Machine Learning: An Early Draft of a Proposed Textbook*. 1998. URL: http://robotics.stanford.edu/people/nilsson/mlbook.html.

[8] Edpresso. *What is feature extraction?* 2019. URL: https://www.educative.io/edpresso/what-is-feature-extraction (visitado 30-09-2010).

[9] Akinori Hidaka y Takio Kurita. «Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks». En: vol. 2017. Dic. de 2017, págs. 160-167. DOI: 10.5687/sss.2017.160.

[10] Mete Ahishali y col. «Classification of polarimetric SAR images using compact convolutional neural networks». En: *GIScience & Remote Sensing* 58.1 (2021), págs. 28-47. DOI: 10.1080/15481603.2020.1853948. eprint: https://doi.org/10.1080/15481603.2020.1853948. URL: https://doi.org/10.1080/15481603.2020.1853948.

[11] Jiuxiang Gu y col. *Recent Advances in Convolutional Neural Networks*. 2015. DOI: 10.48550/ARXIV.1512.07108. URL: https://arxiv.org/abs/1512.07108.

[12] Kamel Abdelouahab. «Reconfigurable hardware acceleration of CNNs on FPGA-based smart cameras». Dic. de 2018.