Universitat Oberta de Catalunya

uoc.edu

El salto cualitativo de Deep Learning en problemas de clasificación

María López Bautista
Máster en Ingeniería de Telecomunicación
Procesamiento de la señal

Pere Martí Puig
José Antonio Morán Moreno

10/06/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*El salto cualitativo de Deep Learning en problemas de clasificación*
Nombre del autor:	*María López Bautista*
Nombre del consultor/a:	*Pere Martí Puig*
Nombre del PRA:	*José Antonio Morán Moreno*
Fecha de entrega (mes/año):	06/2018
Titulación:	*Máster en Ingeniería de Telecomunicación*
Área del Trabajo Final:	*Procesamiento de la señal*
Idioma del trabajo:	*Castellano*
Palabras clave	*peces, identificación, clasificación, aprendizaje profundo, SVM, redes neuronales convolucionales*

Resumen del Trabajo

Los procesos biológicos están estrechamente relacionados con el comportamiento de los animales, por lo que su observación cada vez adquiere más importancia. El método más común para ello es la grabación de vídeo en escenarios controlados durante un tiempo prolongado.

Para el rastreo y detección de diferentes individuos que cohabitan en un mismo espacio ya existen algoritmos que ofrecen resultados aceptables. Sin embargo, el reto todavía se encuentra en mantener la identidad de cada individuo a lo largo del seguimiento de larga duración que se pretende. Este propósito se convierte en complejo especialmente cuando los peces desaparecen del objetivo de la cámara, se cruzan entre ellos o se difuminan por la calidad obtenida en la imagen.

Así pues, el presente TFM (Trabajo Final de Máster) tiene como objetivo proponer una solución a esta problemática mediante la aplicación de métodos de aprendizaje máquina. Éstos, junto a los algoritmos de seguimiento mencionados, permitirán mejorar la información recabada de los diferentes individuos observados con el fin de poder extraer conclusiones detalladas sobre su conducta.

El trabajo se centra en la aplicación de Deep Learning como algoritmo principal para la clasificación e identificación de los individuos. Además, se implementarán clasificadores con un enfoque más tradicional para poder

observar el salto cualitativo que aportan las redes neuronales al objetivo que nos ocupa.

Abstract (in English, 250 words or less):

Biological processes are closely related to animals behavior, so their observation is getting more important. The most common method for this is video-recording at controlled situations for a long time.

There are already algorithms that offer suitable results for tracking and detection of different individuals that cohabit in the same space. However, long-term tracking while maintaining the identity of animals is still a challenge. Besides, this can be more complicated when fish disappear from the camera's lens, cross each other or are blurred because of the quality of the image.

Therefore, this master's thesis aims to propose a solution to this problem through machine learning methods application. These, with the tracking algorithms, will improve the information collected about the different individuals observed to draw conclusions about their behavior.

The work focuses on the use of Deep Learning as the main algorithm for the classification and identification of fish. Later, it will be compared with other machine learning methods, concluding the qualitative leap that Deep Learning entails.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del trabajo

Diversos estudios biológicos sobre la selección natural han demostrado que el comportamiento de las especies es un factor influyente en su evolución [1] [2] [3]. En relación con ello concluyen que las dinámicas negativas en el tamaño de ciertas poblaciones tienen una estrecha relación con las personalidades que presentan sus individuos. De esta manera, la observación de la conducta de esas especies se convierte en un elemento clave para posibilitar un análisis específico del desarrollo de su población.

En el caso que nos ocupa, se dispone de un grupo de cinco peces cuyo comportamiento está siendo monitorizado a través de cámaras de vídeo con el fin de poder establecer las relaciones pertinentes entre su conducta y su personalidad [4].

Las actuales mejoras tecnológicas han permitido el uso generalizado de dispositivos subacuáticos para recopilación de datos, estimación de parámetros sobre la población y monitorización continua. Sin embargo, el estudio de la personalidad a través de la grabación de vídeo del conjunto de individuos a analizar todavía presenta un desafío importante porque requiere de un reconocimiento individual de cada sujeto. Lo que se pretende es definir la personalidad del pez a partir de sus patrones de uso espacial tras un tiempo de observación prolongado; puesto que su conducta se asociará, a través del consiguiente análisis estadístico, a su movimiento por el tanque en el que está siendo grabado.

Al inicio de este trabajo, el algoritmo que usa las grabaciones para el seguimiento del itinerario que describe cada pez ya ha sido implementado y ejecutado [4], por lo que podemos dar por cerrada la etapa de tracking de los peces. Sin entrar en detalle, destacar que el seguimiento es llevado a cabo mediante filtros de Kalman, con los cuales se obtienen buenos resultados.

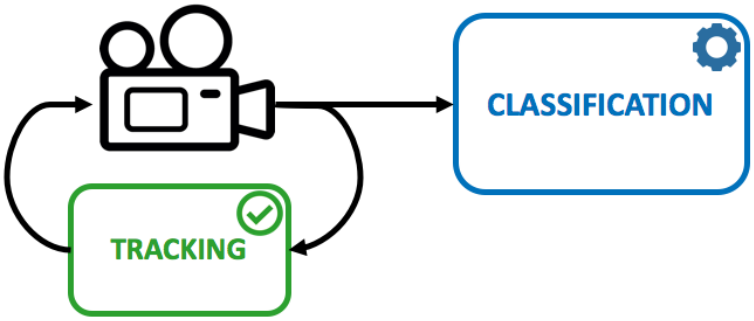

Ilustración 1. Proceso de identificación espacio-temporal del individuo

Sin embargo, para poder completar el estudio del comportamiento de dos o más individuos que cohabitan en un mismo espacio sin cometer errores es necesario ser capaces de identificar cada pez inequívocamente tras varios minutos de grabación. En ese punto es donde se pretende centrar este

trabajo, en la aplicación de diferentes algoritmos que permitan clasificar cada uno de los peces que conforman el grupo de estudio con el menor porcentaje de error posible. Para llevar a cabo dicha clasificación nos vamos a centrar en el empleo de diversos métodos de aprendizaje máquina.

En este contexto, el aprendizaje automático lo entendemos como la rama de la inteligencia artificial que se fundamenta en la creación de software que, en base a cierta cantidad de información que sirve como ejemplo, genera modelos analíticos de forma automática. Estos modelos permitirán, posteriormente, determinar a qué categoría pertenece un nuevo dato en base a patrones o características determinadas que presente la información dada. Una de las perspectivas más interesantes en la actualidad del aprendizaje máquina es el denominado “Deep Learning”. Éste realiza una abstracción de los datos originales de entrada a través del empleo de un gran número de transformaciones no lineales donde, por ejemplo, dada la matriz de píxeles de una imagen el sistema pueda interpretar elementos y extraer características de la escena más allá del valor número de cada píxel.

Así pues, el trabajo se centra en la aplicación, sobre las imágenes disponibles que describen el movimiento de los peces, tanto de Deep Learning a través de diferentes estrategias como de enfoques más tradicionales con el fin de compararlos y poder evidenciar las mejoras que conlleva el empleo de Deep Learning.

1.2 Objetivos del Trabajo

El presente trabajo tiene como objetivo principal el estudio del aprendizaje profundo y su utilidad en problemas de clasificación de imágenes. Este propósito se puede desglosar en los siguientes:

- Investigación sobre las oportunidades de Deep Learning.
- Identificación de individuos dentro de una población de peces.
- Desarrollo y configuración de algoritmos de aprendizaje máquina.
- Comparación de métodos de aprendizaje de diferente naturaleza contrastando los resultados obtenidos con cada uno.
- Generación de base de datos de información clasificada que nos permita entrenar los modelos desarrollados.

Indicar en este punto que los diferentes procesos que se van a implementar para conseguir estos objetivos se llevarán a cabo con la herramienta MATLAB.

1.3 Enfoque y método seguido

Partiendo de las imágenes capturadas por el algoritmo de seguimiento descrito en [4], el primer paso a efectuar es el de clasificarlas para poder obtener una base de datos que conforme el soporte con el que entrenar los modelos que se desarrollarán posteriormente.

Una vez disponible la información, se da paso al periodo de investigación en el que se estudian las diferentes alternativas a emplear como métodos de aprendizaje máquina sobre el problema que nos ocupa. Este análisis recorre desde las opciones más clásicas hasta el aprendizaje profundo y tiene como resultado la decisión de qué métodos se van a aplicar sobre nuestros datos. La elección depende directamente del objetivo principal, que se trata del estudio de Deep Learning y las capacidades que ofrece, tanto positivas como negativas.

Con la decisión tomada, y a partir de un conjunto de información clasificada que se emplea como conjunto de entrenamiento, se procede a desarrollar modelos capaces de extraer patrones de dicha información para poder llevar a cabo predicciones sobre nuevas imágenes capturadas sobre el conjunto de peces.

Por último, se evaluarán los modelos entrenados con una serie de nuevas imágenes y se compararán entre sí mediante los resultados obtenidos.

1.4 Planificación del Trabajo

De cara a realizar la planificación de este proyecto, los dos puntos principales a considerar son los recursos disponibles y la identificación de tareas que realizar. Con ello será posible realizar el diagrama de Gantt en el que se representan las actividades, su duración y las dependencias entre ellas.

Así pues, el trabajo se ha desglosado en los siguientes conjuntos de actividades:

Tarea	Duración	Descripción y dependencias
Definición TFM	35h	Fecha objetivo: 10/03/2018
Análisis previo	20h	Estudio inicial sobre aprendizaje máquina, deep learning y sus diferentes aplicaciones
Propuesta de título	1h	Determinación del título el proyecto
Objetivos	3h	Definición de objetivos principales del proyecto
Índice	3h	Definición del índice del proyecto
Planificación	8h	Preparar la hoja de ruta del proyecto
Estado del Arte	35h	Fecha objetivo: 25/03/2018
Búsqueda bibliográfica	5h	Selección de información sobre las técnicas a utilizar (en qué consiste, proyectos ejemplo…)
Familiarización	12h	Lectura y comprensión de datos recopilados
Redacción	18h	Escribir el estado del arte en la memoria
Implementación	85h	Fecha objetivo: 10/05/2018
GUI de clasificación	12h	Desarrollo de interfaz gráfica que facilite la clasificación manual
Clasificación manual	16h	Ejecución de la clasificación manual de más de 12.000 imágenes
Adecuación entorno	2h	Preparación del entorno de trabajo para iniciar la implementación de los modelos
Implementación DL	30h	Desarrollo de modelos con Deep Learning

Implementación SVM	25h	Desarrollo de modelos que empleen SVM
Ejecución	90h	Fecha objetivo: 01/06/2018
Ejecución DL	40h	Entrenamiento de modelos Deep Learning
Ejecución SVM	30h	Entrenamiento de modelos que empleen SVM
Comparar resultados	20h	Evaluación de los modelos y extracción de resultados para medir su eficacia
Conclusiones	5h	
Documentación	55h	Fecha objetivo: 22/06/2018
Memoria	30h	Redacción el documento final del TFM
Presentación	25h	Generación del documento de exposición del TFM en el formato previamente seleccionado

Tabla 1. Actividades del proyecto

De este listado de tareas se pueden extraer una relación de los hitos relevantes que componen el proyecto a efectuar, los cuales serían:

Tarea	Fecha
Entrega Definición TFM	10/03/2018
Entrega Estado del Arte	23/03/2108
Entrega Implementación	29/04/2018
Entrega Memoria	11/06/2018
Entrega Presentación	20/06/2018

Tabla 2. Hitos del proyecto

Para el cumplimiento de estos hitos y la ejecución de las tareas descritas anteriormente se ha contado con un alumno, el cual se estima dedica una media de 20 horas semanales con el soporte necesario del tutor, Pere Martí-Puig. Esta ocupación se reparte con dos horas diarias de lunes a viernes y 10 horas repartidas en el fin de semana. En la etapa de implementación y ejecución de los modelos se ampliará puntualmente la dedicación.

Finalmente, considerando todas las acotaciones anteriores, el diagrama de Gantt queda tal y como se puede ver en la siguiente página.

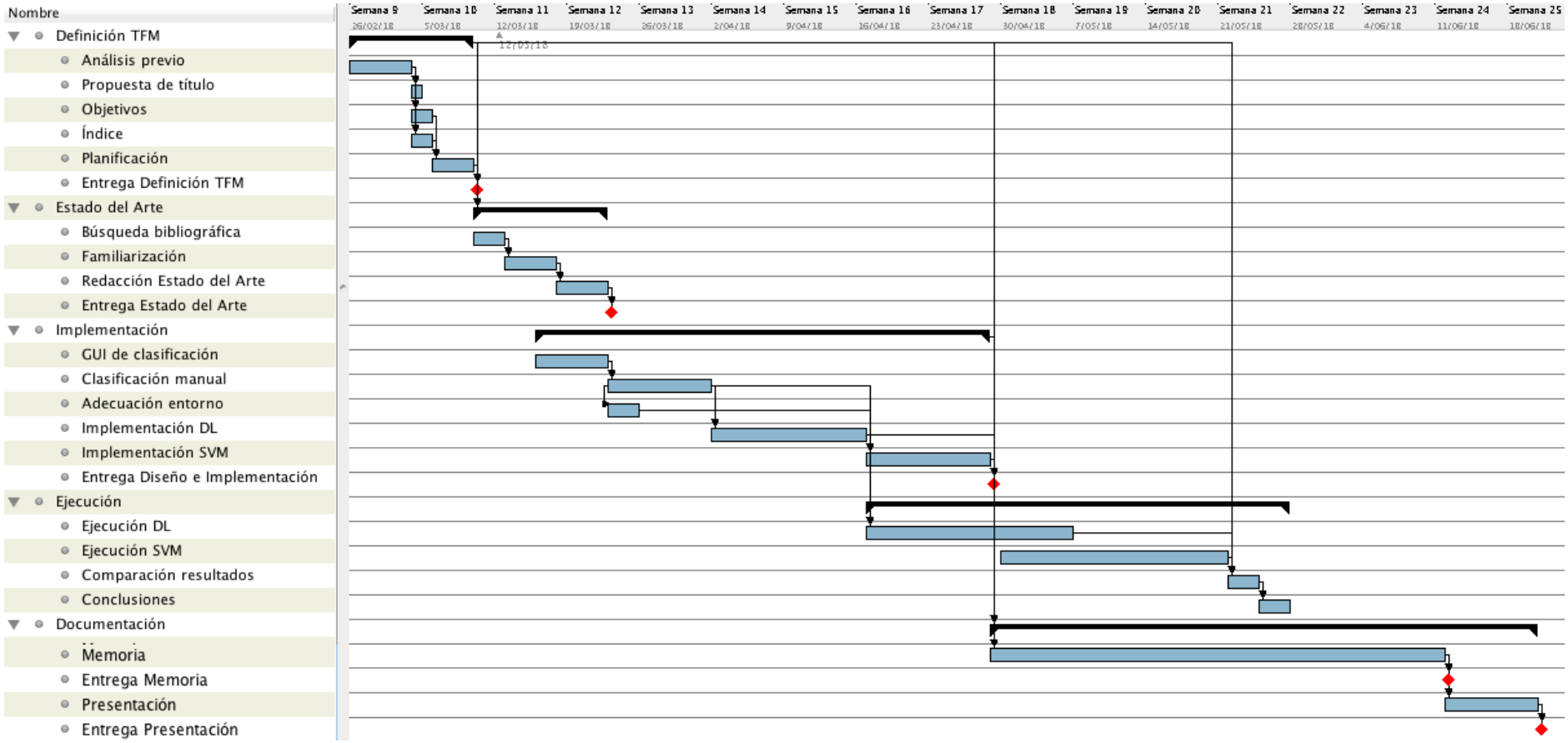

Ilustración 2. Diagrama de Gantt del proyecto

1.5 Breve sumario de productos obtenidos

Los productos obtenidos tras el desarrollo de este proyecto son:

- Sencilla interfaz gráfica de usuario que nos permita llevar a cabo la clasificación manual de imágenes de una forma lo más amigable posible.
- Algoritmo de clasificación mediante Deep Learning utilizando transferencia de aprendizaje.
- Algoritmo de clasificación mediante Deep Learning llevando a cabo un entrenamiento desde cero. Creación de una red neuronal.
- Algoritmo de clasificación que combina Deep Learning con SVM (Support Vector Machine). Emplea Deep Learning para la extracción de características y SVM para la clasificación de imágenes.
- Algoritmo de clasificación que emplee un SVM para clasificar imágenes basándose en características extraídas mediante segmentación del color.

Finalmente, se compararán las ejecuciones de los diferentes algoritmos descritos para extraer un listado de capacidades y limitaciones de ellos.

1.6 Breve descripción de los otros capítulos de la memoria

Tal y como se puede observar en el índice de esta memoria, los puntos recogidos recorren tanto desde un punto de vista teórico como práctico qué es el aprendizaje máquina de forma global, cómo se materializa y concreta el aprendizaje profundo y qué diferencias hay entre la aplicación de diferentes métodos.

Así pues, los siguientes capítulos contendrán:

- Estado del Arte: incluye la definición de aprendizaje máquina y aprendizaje profundo, presentando la utilidad de esos métodos y sus aplicaciones más recientes. Además, se describen las redes neuronales y sus diferentes componentes, en los que se basa Deep Learning.
- GUI de clasificación: en el primer apartado de componente práctico se explica la tipología de los datos disponibles y la interfaz gráfica que se ha desarrollado para facilitar su clasificación.
- Modelos tradicionales: define el algoritmo empleado para caracterizar el color de las imágenes y utilizarlo como característica para entrenar un SVM que servirá de clasificador. Se denomina modelos tradicionales a aquellos que siguen el flujo habitual de un clasificador.
- Modelos basados en Deep Learning: descripción de las diferentes estrategias que se pueden emplear para sacarle partido a las ventajas de Deep Learning. En este apartado se expondrán los tres enfoques con los que se han utilizado las redes neuronales: transferencia del aprendizaje (ajuste de un modelo previamente entrenado), entrenamiento desde cero (creación de una arquitectura de red creada desde cero) y extracción de características (de las capas más profundas de la red).
- Conclusiones: se divide principalmente en dos apartados. El primero es el de Resultados, el cual engloba el resultado obtenido en las ejecuciones

de los diferentes algoritmos desde un punto de vista cuantitativo. El segundo, Conclusiones, pretende analizar los datos anteriores y valorarlos cualitativamente.

2. Estado del Arte

2.1. Aprendizaje Máquina

Los algoritmos de aprendizaje automático permiten a los ordenadores hacer aquello que humanos y animales realizan de forma natural: aprender de la experiencia. Dichos algoritmos giran en torno al uso de métodos computacionales que permiten “aprender” información a partir de los datos sin depender de una ecuación que defina en su totalidad al modelo. Así pues, emplean métodos adaptativos que mejoran su rendimiento a medida que aumenta el número de muestras disponibles para su entrenamiento [5].

El desempeño de los diferentes métodos de aprendizaje máquina se basa en encontrar patrones naturales en los datos proporcionados que permiten generar una base de conocimiento. Más tarde, con ella será posible llevar a cabo predicciones y tomar mejores decisiones.

El uso de estas técnicas debe ser considerado cuando se desee afrontar un problema complejo que implique gestionar una gran cantidad de datos y muchas variables, pero no se disponga de ninguna ecuación que las relacione. Con ello, su empleo está extendido tanto en actividades diarias en las que es necesario tomar decisiones críticas como en otras menos comunes que están viéndose potenciadas gracias al reciente crecimiento del Big Data. Entre todas ellas, se pueden destacar las siguientes:

- El diagnóstico médico: detección de tumores, secuenciación del ADN, descubrimiento de fármacos.
- Predicciones financieras: compra-venta de acciones, calificación del crédito (como medida de riesgo), detección de fraudes.
- Procesamiento de imágenes y visión artificial: detección de movimiento, detección de objetos, reconocimiento facial.
- Control de la producción de energía: previsiones de carga o tasación de su precio.
- Predicciones útiles para el mantenimiento de las industrias aeroespacial, automovilística o de producción masiva.
- Procesamiento de lenguaje natural.
- Previsión de las tendencias de compras de los clientes en base a patrones de su comportamiento.
- Recomendaciones al consumidor de páginas web de contenido audiovisual.

Ilustración 3. Aplicaciones reales del aprendizaje máquina [5]

El enorme desarrollo que está viviendo la tecnología asociada a la Inteligencia Artificial está dando lugar, en los últimos tiempos, a aplicaciones sensacionales. Tal es así que, de acuerdo con Gartner [7], las tecnologías de aprendizaje automático estarán presentes en casi todos los nuevos productos software para 2020.

Una de las áreas donde los avances han sido más apreciables es el reconocimiento de imágenes, campo en el que se centra el presente trabajo. Ya en 2016, según un estudio de O'Really [6] sobre el mercado de la IA, el reconocimiento de imágenes era una de las áreas donde más empresas en EE. UU. estaban invirtiendo, considerando la inversión global en Inteligencia Artificial.

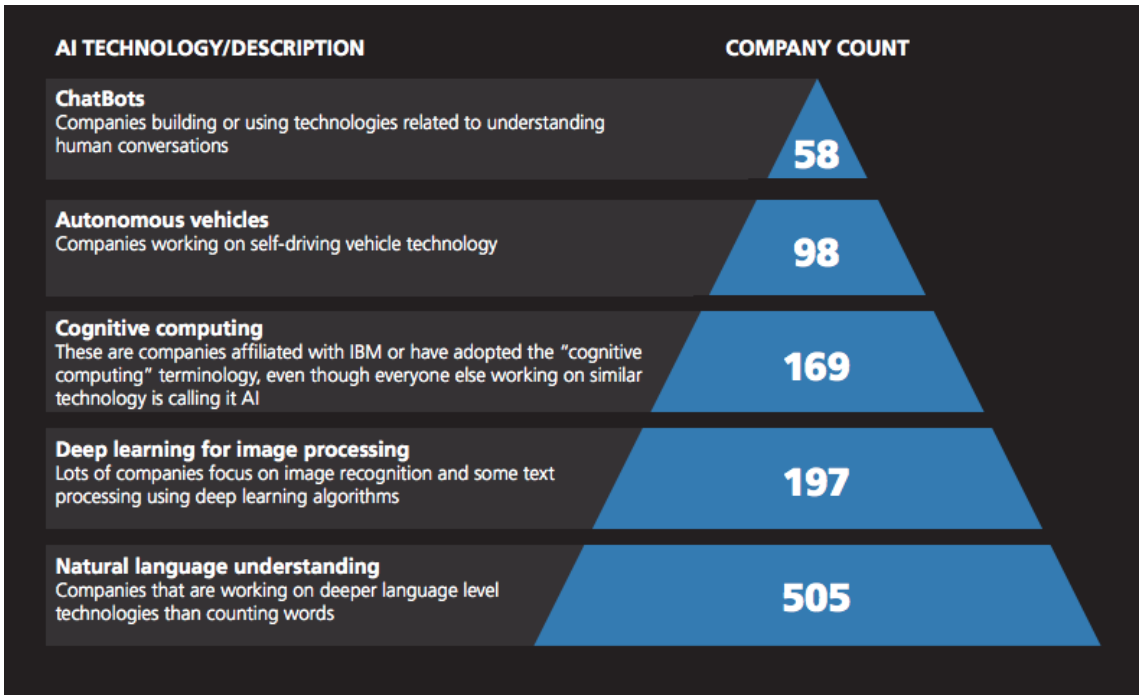

Ilustración 4. Categorías de tecnologías IA e inversión de empresas [6]

Dentro de este campo los casos de uso son variados. Entre los más interesantes se pueden destacar:

- Etiquetado de imágenes. El objetivo de estas aplicaciones es extraer palabras clave de las imágenes para poder indexarlas y realizar posteriores búsquedas.
- Análisis de clientes. Detectando los productos que consume el usuario (por logos, textos, etc.) permite ampliar el conocimiento sobre el mismo.
- Diagnóstico de enfermedades a través del análisis de imágenes médicas en base a comparación con diagnósticos previos.
- Realidad virtual. Gaming o interacción con el medio entre otros.
- Seguridad. Identificación de matrículas, análisis de grabaciones de cámaras de seguridad, extracción de patrones de comportamientos sospechosos, verificación de usuarios basada en el rostro.

Para concluir con este apartado introductorio sobre el aprendizaje máquina, se puede resumir esta rama de la inteligencia artificial como la especialidad que pretende establecer procesos de inducción al conocimiento creando programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos.

2.2. Tipos de aprendizaje máquina

A continuación, se van a presentar los diferentes tipos de aprendizaje automático existentes en función del mecanismo utilizado para aprender [5].

- Aprendizaje supervisado: entrena un modelo con datos de entrada y salida conocidos para que se puedan predecir salidas futuras. Se suele utilizar si se necesita llevar a cabo una predicción o una clasificación.
- Aprendizaje no supervisado: encuentra patrones ocultos o estructuras intrínsecas entre los datos de entrada. Se emplea en los casos en los que se necesita explorar los datos para localizar una buena representación interna, como la división de datos en clusters.

Ilustración 5. Tipos de aprendizaje máquina

2.2.1. Aprendizaje supervisado

El aprendizaje automático supervisado produce una función que establece una correspondencia entre las entradas y salidas deseadas del sistema. Los algoritmos de este tipo de aprendizaje tienen como objetivo la predicción, con lo que toman un conjunto conocido de datos de entrada y

respuesta conocidas para esos datos y entrenan un modelo con objeto de generar pronósticos razonables como respuesta a datos nuevos.

Para desarrollar dichos modelos predictivos el aprendizaje supervisado emplea técnicas de clasificación y regresión.

- Técnicas de clasificación: predicen respuestas discretas. Los modelos de clasificación organizan los datos de entrada en categorías, por lo que se recomienda el uso de estos métodos si los datos disponibles se pueden etiquetar, categorizar o dividir en grupos concretos. Sus aplicaciones más habituales son las imágenes médicas, el reconocimiento de voz y la calificación crediticia. Algunos algoritmos habituales para realizar la clasificación son: SVM, árboles de decisión, k-vecino más cercano, clasificadores bayesianos, análisis discriminantes, regresión logística y redes neuronales.

- Técnicas de regresión: predicen respuestas continuas. Ese tipo de modelos se basan en la evaluación de una o más variables para predecir su comportamiento, por lo que se recomienda su uso si se trabaja con variables reales, como la temperatura o el tiempo que tarda una pieza de equipamiento en fallar. Sus aplicaciones más habituales son la predicción de cargas o el trading algorítmico. Algunos algoritmos habituales de regresión son: modelos lineales, regresión por pasos, redes neuronales o aprendizaje neuro difuso adaptativo.

2.2.2. Aprendizaje no supervisado

El aprendizaje automático no supervisado tiene como objetivo encontrar estructuras intrínsecas en los datos. Se emplea para inferir información a partir de conjuntos de datos de entrada sin respuestas etiquetadas.

El clustering (agrupación) es la técnica más común. Se basa en la exploración de datos con objeto de encontrar patrones ocultos en los mismos. Sus aplicaciones más comunes son el análisis de secuencias genéticas, la investigación de mercados o el reconocimiento de objetos.

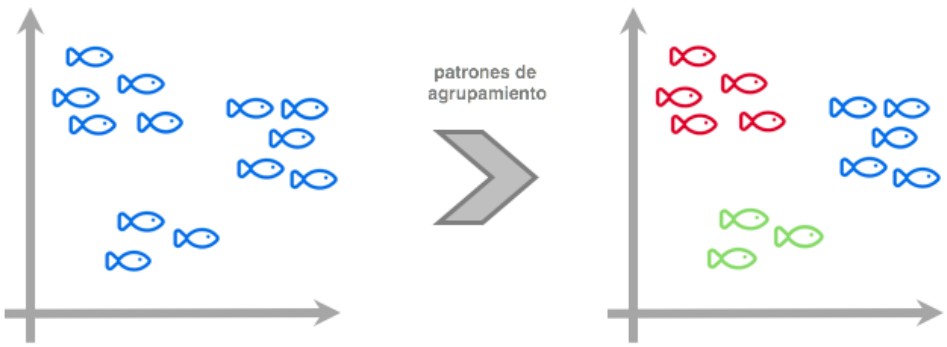

Ilustración 6. Clustering

Algunos algoritmos habituales para realizar clustering son: clustering jerárquico, modelos de mezclas gaussianas, modelos de Markov ocultos, mapas auto organizados o clustering sustractivo.

2.3. Aprendizaje Profundo

A pesar de que el Deep Learning parece una técnica nueva e innovadora, las investigaciones al respecto surgen hace más de 30 años cuando el japonés Kunihiko Fukushima propone un modelo neuronal de entre cinco y seis capas. Sin embargo, la complejidad de su desarrollo y el coste elevado de su investigación provoca que se posponga y se retome con fuerza hace menos de 10 años [9]. Los motivos principales por los que solo ha empezado a resultar útil recientemente son:

- Requiere de grandes cantidades de datos etiquetados.
- Requiere de una potencia de cálculo significativa. Las GPU (Graphics Processing Unit) de alto rendimiento que tienen arquitectura paralela resultan eficientes para la aplicación de técnicas de aprendizaje profundo.

El boom de esta técnica se produce cuando en 2012 un grupo de investigación de la Universidad de Toronto obtiene resultados sorprendentes en una de las principales competiciones de reconocimiento de imágenes: ILSVRC [13], ImageNet Large Scale Visual Recognition Challenge. Desde entonces, su papel en muchas áreas de inteligencia artificial ha aumentado y sus resultados mejoran a técnicas tradicionales en todo aquello que tiene que ver con reconocimiento de patrones, imágenes, voz, etc. El salto de calidad que aporta el Deep Learning a estas tareas viene determinado porque supera con éxito el desafío de la variabilidad intra-clase. Ésta consiste en las diferencias notables que existen entre imágenes que pertenecen a la misma clase. Con diferentes técnicas aplicadas en Deep Learning se consigue minimizarla.

El aprendizaje profundo se basa en el aprendizaje máquina y no es más que una familia de algoritmos cuyo propósito es simular el comportamiento que lleva a cabo nuestro cerebro para reconocer imágenes, palabras o sonidos. En el cerebro humano, ante un estímulo, ciertas neuronas se activan, evalúan la información que han recibido, reaccionan y se comunican con otras neuronas. En posteriores estímulos añaden nuevos datos a los que ya conocen, evalúan el resultado de las acciones anteriores y corrigen su funcionamiento para tener la mejor reacción posible [8].

Los algoritmos citados son empleados por las redes neuronales artificiales para actuar de forma similar al cerebro, pero simplificando el conjunto de neuronas empleado en distintas capas. Así, cada grupo de neuronas analiza los datos de entrada, los procesa y los entrega a otro grupo de neuronas en forma de datos. Los buenos resultados de este método son posibles por la cantidad de conexiones existentes entre las diferentes capas. El término profundo suele hacer referencia al número de capas ocultas en la red

neuronal. Las redes neuronales tradicionales solo contienen dos o tres capas, mientras que las redes profundas pueden tener hasta 150. [11].

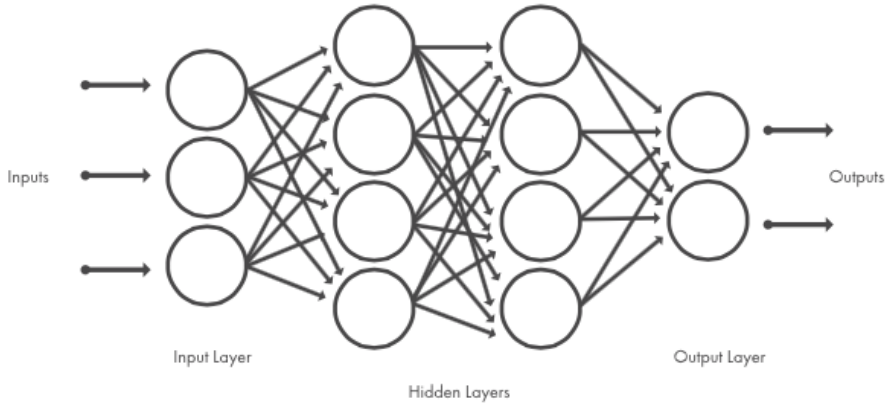

Ilustración 7. Red neuronal artificial [11]

Desde un punto de vista matemático, podemos ver el aprendizaje profundo como un modelo que procesa la información que recibe en la entrada, codificada como números, para hacerla pasar por una serie de operaciones matemáticas (las capas de neuronas) y obtener como salida información codificada que se puede adaptar para llevar a cabo una función deseada concreta. De esta manera, Deep Learning funciona gracias a la arquitectura que forman las diferentes capas y a la rutina que emplea para optimizar dicha arquitectura.

El gran avance de este algoritmo respecto a Machine Learning se basa en que el sistema puede entrenarse a sí mismo para encontrar coherencia en datos aleatorios. A partir de ello, mejora el rendimiento con unos niveles elevados de precisión, consiguiendo sistemas que superan a los propios humanos en tareas de clasificación y detección en imágenes.

Entre las aplicaciones que más están explotando las capacidades de estas técnicas se encuentran [11]:

- Conducción autónoma. Se utiliza tanto para detectar peatones como objetos como señales y semáforos.
- Sector aeroespacial y de defensa. El aprendizaje profundo resulta útil para identificar para explorar áreas con las imágenes capturadas por satélites.
- Investigación médica. La detección de células cancerígenas es una de las áreas donde más partido se le está sacando al aprendizaje profundo.
- Electrónica. Traducción del habla, previsión de comportamientos del usuario, etc.
- Automatización industrial. Se emplea para mejorar la seguridad de los trabajadores.

Sea la aplicación que sea que se le quiere dar al aprendizaje profundo, se pueden utilizar dos posibles enfoques [8]:

- Entrenar una red neuronal desde cero: situación en la que son necesarios muchos datos para entrenar la red, lo que requiere emplear desde días a semanas para ejecutar casos reales. La ventaja es que sus resultados son muy exactos.

- Modificar una red ya entrenada para otros fines: Transfer Learning es la tarea de utilizar el conocimiento previo provisto por una red pre-entrenada para aprender nuevos patrones en nuevos datos. Mediante la modificación de los pesos de las últimas capas de la red se puede adaptar ésta para ser utilizada con datos diferentes a aquellos para los que fue creada. Ese enfoque resulta más rápido y sencillo que el entrenamiento desde cero y es adecuado cuando se dispone de pocas muestras de datos para entrenar [10]. La ventaja de la transferencia del aprendizaje es que la red pre-capacitada ya dispone de un conjunto de características "aprendidas" que puede aplicar en tareas similares. La precisión de este modelo es directamente proporcional a la calidad de la red pre-entrenada.

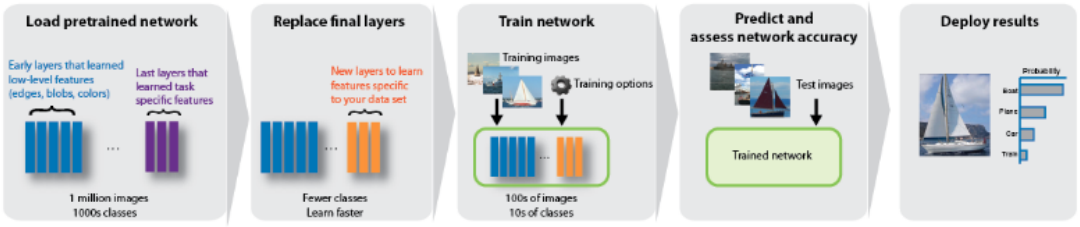

Ilustración 8. Transfer Learning [10]

Por último, antes de pasar a describir las redes neuronales, si bien son éstas las que se usarán en el presente trabajo, es necesario aclarar que no es el único modelo que se emplea para que los ordenadores sean capaces de llevar a cabo aprendizaje profundo. Otros ejemplos son las reglas de asociación, las redes bayesianas, los árboles de decisión o el aprendizaje por refuerzo.

2.3.1. Redes Neuronales

Una red de neuronas, llamada comúnmente red neuronal artificial, es un modelo matemático formado por una serie de operaciones que, para un vector de entrada x, ofrece un vector de salida distinto f(x). Otra forma de verlas, como ya se ha indicado en el apartado anterior, es como un procesador que recibe información entrante codificada como números, hace una serie de operaciones y produce como resultado información saliente igualmente codificada como números.

Para explicar su funcionamiento nos basaremos en la figura 7. Las neuronas se organizan en capas, donde cada una se conecta con todas las de la capa siguiente. En esta arquitectura, cada conexión tiene asociado un peso, por lo que la principal operación que se realiza es una multiplicación entre el valor de la neurona y el de su conexión saliente. De esta manera, las neuronas de las capas siguientes reciben los resultados anteriores sumados en uno y aplican funciones no lineales para producir un nuevo resultado. En

este punto, según la red neuronal utilizada, las funciones serán de uno u otro tipo. La más común es la función sigmoide.

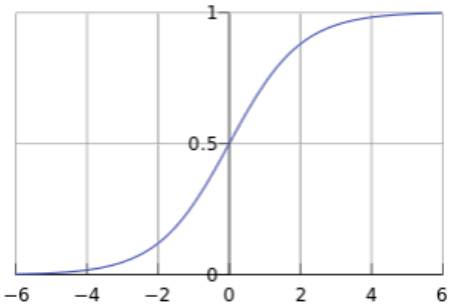

Ilustración 9. Función sigmoide [15]

Existen tres tipos de capas en las que se agrupan las neuronas:

- Capas de entrada: Neuronas que introducen los valores de entrada en la red, por lo que dispondrá de tantas neuronas como datos de entrada. En estas neuronas no se produce procesamiento.
- Capas ocultas: Cálculos intermedios de la red. Cada neurona oculta contiene un peso y un parámetro que son los elementos que transformarán los datos de entrada. El número de capas ocultas es variable.
- Capas de salida: Neuronas cuyos valores corresponden a los valores de salida de la red neuronal. En esta capa cada neurona también tendrá asociado pesos y parámetros.

En la arquitectura descrita, el proceso de aprendizaje de una red neuronal consiste en, basándose en una serie de ejemplos en los que se conoce el valor de salida para la entrada correspondiente, ser capaz de inducir la relación entre entradas y salidas para datos desconocidos. Este proceso se denomina entrenamiento de la red y puede ser visto como el proceso de ajuste de los parámetros de la red. Partiendo de un conjunto de pesos aleatorios, el proceso de aprendizaje busca adaptar el valor de los mismos que permitan a la red conseguir el comportamiento deseado. El proceso de aprendizaje es un proceso iterativo en el cual se refina la solución hasta alcanzar un nivel de operación satisfactorio. De hecho, la mayoría de los métodos de entrenamiento utilizados en redes neuronales consisten en definir una función de error que mida el rendimiento de la red en función de los pesos. A partir de ella, el objetivo es encontrar un conjunto de pesos que la minimicen o maximicen, determinando con ellos el punto óptimo de la red neuronal.

2.3.2. Redes Neuronales Convolucionales

Uno de los tipos más populares de redes neuronales profundas son las conocidas como redes neuronales convolucionales: CNN o ConvNet [11]. Las capas que utilizan estas redes son 2D, por lo que resulta una arquitectura adecuada para procesar datos 2D tales como imágenes.

En general, una red convolucional no es más que una modificación de una red neuronal básica donde la principal diferencia radica en que los pesos son filtros que se convolucionarán directamente con la imagen [16]. La particularidad de este tipo de redes es que cada neurona de una capa no recibe conexiones de todas las neuronas de la capa anterior, sino sólo de algunas. Esto favorece que una neurona se especialice en una región de la lista de códigos de la capa anterior, reduciendo drásticamente el número de pesos y de multiplicaciones necesarias.

En cuanto a las principales diferencias de las CNN con otros modelos de redes neuronales tradicionales destacan dos:

- Las CNN disponen de un número de capas elevado, por lo que será necesario tener en cuenta el papel que va a desempeñar las capacidades de GPU que tenga el dispositivo a utilizar.

- Las CNN eliminan la necesidad de una extracción de características manual, por lo que no es necesario identificar previamente las características a utilizar para clasificar las imágenes. En las CNN las características relevantes no se entrenan, se aprenden mientras la red se entrena con una colección de imágenes. Esta extracción de características automatizada hace que los modelos de aprendizaje profundo sean muy precisos para la clasificación de objetos.

En la imagen a continuación se muestra la estructura básica de un CNN. En ella se puede observar cómo las capas de más alto nivel llevan a cabo la extracción de características mencionada, y las capas más bajas clasifican y deciden la categoría de la imagen de entrada. En ese primer bloque cada capa oculta aumenta la complejidad de las características aprendidas. Por ejemplo, la primera capa podría aprender a detectar bordes, mientras la segunda aprende cómo detectar formas más complejas propias del objeto que se pretende reconocer.

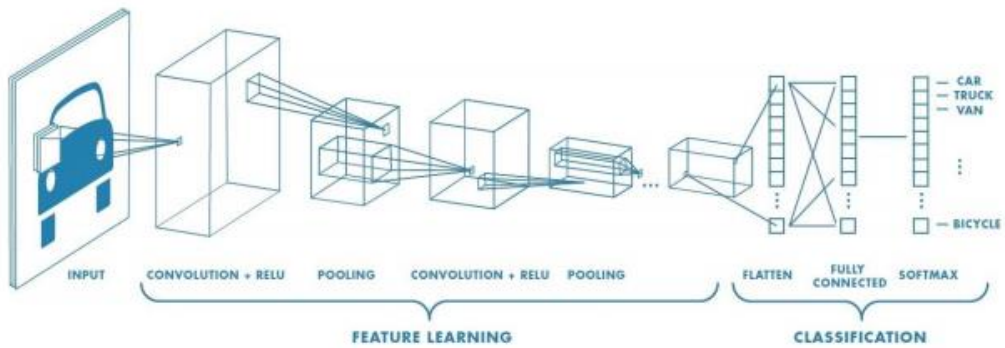

Ilustración 10. Estructura de una CNN [17]

Como se puede ver en la anterior figura, una red convolucional está formada por los siguientes tipos de capas:

- Capa convolucional: capa oculta donde se define, para cada neurona, los bias (o parámetros de ajuste), los filtros y la función asociada. Incluyen la función de ReLU (función de activación o rectificación). Es la capa principal de las CNN y funciona de tal forma que lleva a cabo una convolución entre un filtro definido y el volumen que recibe como datos de entrada. La operación la lleva a cabo deslizando el filtro por cada una de las porciones de interés de la imagen, tras lo que calcula el producto escalar del filtro con la región cubierta y después hace la suma de los valores.

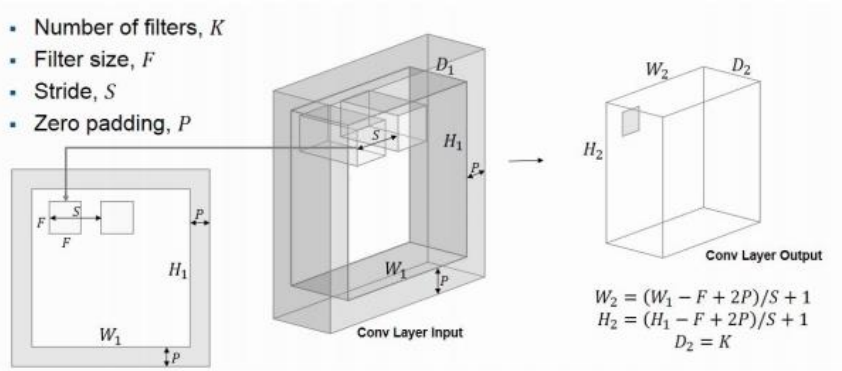

Ilustración 11. Capa de convolución [8]

- Capa de sub-muestreo (pooling): capa opcional que se utiliza para reducir la dimensionalidad de la imagen, lo que permite tanto reducir el coste computacional de convoluciones posteriores y el número de parámetros a utilizar como controlar el sobre-aprendizaje de la red. Los más comunes son max-pooling y average-pooling.

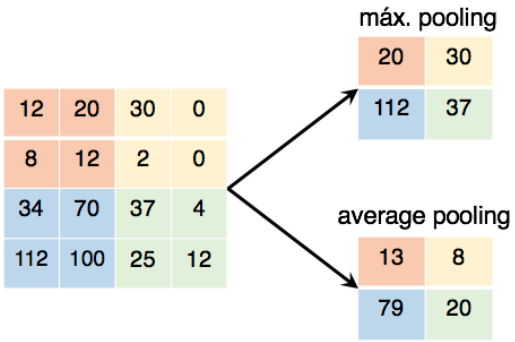

Ilustración 12. Ejemplos de pooling en una CNN

- Capa fully-connected (FCL): red neuronal completa que se conecta a la salida de la última capa convolucional de la red. Es el núcleo de la etapa de clasificación. Requiere de una adaptación intermedia que convierta los datos a un vector, siendo ésa la capa Flatten, tal y como se puede observar en la ilustración 10. Finalmente, la capa Softmax se encarga de asociar una etiqueta a la imagen de entrada aplicando una probabilidad de semejanza.

2.3.3. Aprendizaje Automático vs Aprendizaje Profundo

En este apartado se quieren reunir y explicar las principales diferencias que existen entre el aprendizaje automático (técnicas tradicionales) y el aprendizaje profundo, para esclarecer qué se puede esperar del resultado de este trabajo [11].

En primer lugar, aunque ya se ha indicado anteriormente en el punto de Redes Neuronales Convolucionales, el flujo a través del cual se extraen las características en ambos modelos es diferente. Un flujo de trabajo de aprendizaje automático empieza con la extracción manual de las características relevantes de las imágenes. Dichas características se utilizan posteriormente para crear un modelo que categoriza los objetos de la imagen. Sin embargo, en un flujo de trabajo de aprendizaje profundo, las características relevantes se extraen directamente de la imagen de forma automática, es decir, se realiza un "aprendizaje completo" basado en datos sin procesar y un objeto a satisfacer, por ejemplo, una tarea de clasificación.

Además de la extracción de características, la modelización es automática en el aprendizaje profundo, mientras que la selección de características y de clasificación es manual en el aprendizaje automático.

Otra diferencia clave es que en los algoritmos de aprendizaje profundo la escala aumenta con los datos; mientras que, en el aprendizaje automático existe convergencia. Éstos llegan a un punto muerto en cierto nivel de rendimiento cuando se agregan más ejemplos y datos de entrenamiento a la red. Asociada a esta diferencia se puede deducir una de las ventajas fundamentales de las redes de aprendizaje profundo: siempre mejoran a medida que aumenta la volumetría de los datos.

Por último, las redes de aprendizaje profundo requieren de una gran cantidad de datos para entrenar el modelo, así como GPUs suficientes para procesar los datos con rapidez. El aprendizaje automático ofrece diversas técnicas que se permiten adecuar el modelo a su aplicación, el tamaño de los datos disponibles y el tipo de problema que se desea resolver. Así pues, a la hora de elegir entre ambos tipos de aprendizaje, aunque el profundo se puede considerar una forma especializada del aprendizaje automático, se debe tener en cuenta tanto las capacidades del dispositivo donde se va a ejecutar el proceso como la cantidad de datos etiquetados.

2.3.4. Futuro del Aprendizaje Profundo

En cuanto a los próximos pasos a dar en la materia del Deep Learning, los desafíos críticos a abordar para avanzar en la clasificación de imágenes y sobre los que ya hay investigaciones en activo serían [13]:

- Aprendizaje no supervisado: la mayoría de métodos de aprendizaje profundo aplicados actualmente están supervisados. Esto conlleva que es necesario tener grandes cantidades de datos de entrenamiento

etiquetados, lo cual conlleva un trabajo manual de revisión que resulta tedioso. Para evitar estas tareas los investigadores ya están dando pasos hacia el aprendizaje profundo no supervisado a través de movimientos intermedios como el aprendizaje semi-supervisado o el aprendizaje a partir de una transferencia rápida y efectiva.

- Imágenes adversas: Se trata de imágenes cuya categoría parece obvia para un ser humano, pero causa fallos masivos en una red profunda. Parte del problema procede de la ausencia de comprensión completa que tenemos de los que sucede dentro de las propias redes. En cualquier caso, ya existen trabajos que enfrentan esta dificultad.

- Acelerar el proceso: Las mejoras en el hardware de los últimos tiempos, sobre todo a nivel de GPU, han facilitado el crecimiento exponencial del aprendizaje profundo. Las redes profundas conllevan tal cantidad de operaciones que requieren de grandes capacidades para ejecutarse, lo que generalmente solo puede llevarse a cabo en dispositivos con una GPU de gama alta. El reto reside en poder emplear redes profundas en dispositivos que no disponen de esas capacidades, como pueden ser los dispositivos móviles. Las investigaciones en esta área están avanzando en la dirección de utilizar arquitecturas que efectúan las convoluciones de manera que consiguen reducir tanto el consumo de memoria como el tiempo de inferencia.

3. Clasificación manual de imágenes

3.1. Naturaleza de los datos

El presente trabajo está basado en las imágenes obtenidas por el experimento descrito en [4]. En él se trabaja con un conjunto de cinco peces Serranus scriba, individuos ampliamente distribuidos en el mar Mediterráneo y cuya población se ve fuertemente alterada por la pesca recreativa por motivos de tamaño y modelo reproductivo. La selección de esta especie para el experimento indicado atendía a la razón de que existen evidencias de que su evolución esté estrechamente relacionada con su comportamiento.

Ilustración 13. Serranus scriba (Fuente: photomazza.com)

Los peces fueron capturados en estado salvaje pescando en aguas costeras (de 5 a 10 metros de profundidad) cerca de la bahía de Andratx, en la costa suroeste de Mallorca. Todos ellos eran adultos. Tras ello, los peces fueron marcados por medio de una pistola de marcado de aguja utilizando colores fluorescentes. Aprovechando el momento de manipulación, que no se prolongó más de 1min, se tomaron medidas de longitud y peso de los individuos (longitud total del cuerpo 15.7±1,9cm, masa media 53±21,5g).

Una vez manipulados fueron introducidos en un tanque, de dimensiones 4mx2mx30cm, el cual se llenó con agua de mar y se equipó con un sistema de filtrado en circuito abierto. Para que el comportamiento de los peces no se viese afectado por las condiciones ambientales, el tanque fue iluminado emulando las condiciones de luz natural existentes en el hábitat original de la especie, generando una iluminación homogénea en todo el depósito. En este sentido, también se incluyeron modos de operación que permitiesen pasar por luz diurna, luz azul de atardecer y oscuridad. Además, la temperatura del agua se ha mantenido en una media de 26.2ºC, para asemejarse a la del mar, durante todo el periodo de captura de las imágenes.

El tanque experimental se dividió en dos áreas: el refugio (bloques de piedra y plantas de plástico) y la arena; y fue aislado del resto del laboratorio para evitar estímulos que provoquen desviaciones en el experimento. Por ello, las cámaras, las cuales han capturado 10 imágenes por segundo, se controlan a distancia.

En el momento de iniciar el presente proyecto, la captura por parte de las cámaras ya se realizaba de forma eficaz mediante un algoritmo de detección y multi-tracking que permite capturar el movimiento de los peces y realizarles un seguimiento activo. Este algoritmo se basa en background subtraction (técnica empleada en el procesamiento de imágenes que extrae la información de interés, del primer plano, para su posterior procesamiento) para segmentar el vídeo y extraer de él las imágenes en las que aparecen los individuos y que se utilizarán en los modelos desarrollados aquí. Una vez obtenidas esas imágenes, el mayor desafío, tal y como se adelantaba en la introducción de esta memoria, es identificar a los individuos de forma inequívoca, para lo que se emplearán algoritmos de clasificación.

3.2. Tipología de datos

Para la aplicación de los algoritmos que se describen en los apartados posteriores se han utilizado las imágenes extraídas de secuencias de vídeo obtenidas de las diferentes cámaras instaladas en el tanque experimental durante un mismo periodo de tiempo. En ellas se puede apreciar el seguimiento a los cinco individuos que han participado en los ensayos. Éstos, como se ha indicado anteriormente, estaban marcados, por lo que utilizando los colores de sus marcas ortogonales los podemos distinguir de la siguiente manera:

Pez con marca verde única

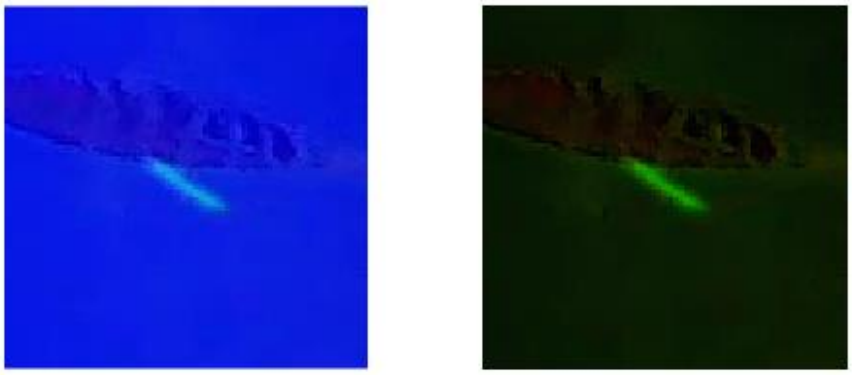

Ilustración 14. Marca verde (Representación RGB - Representación RG)

Pez con marca roja única

Ilustración 15. Marca roja (Representación RGB - Representación RG)

Pez con marca doble roja

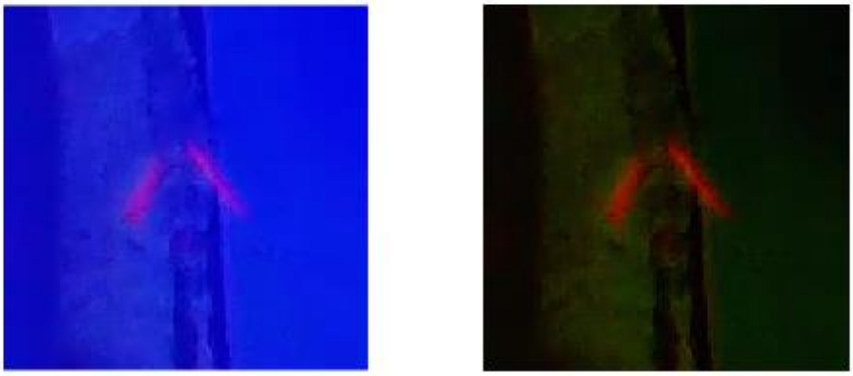

Ilustración 16. Doble marca roja (Representación RGB - Representación RG)

Pez con marca doble verde

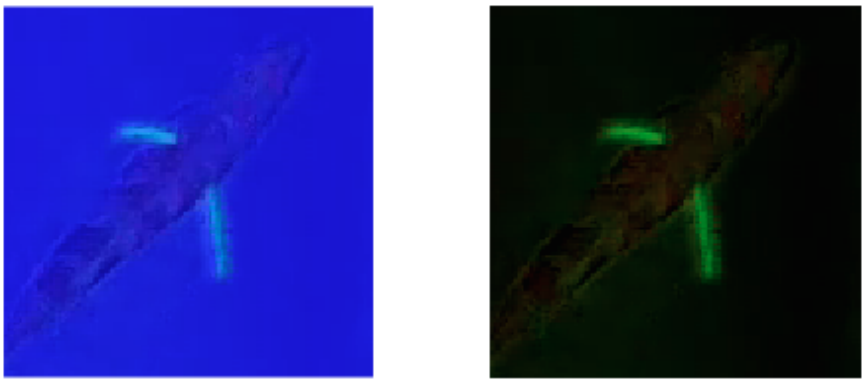

Ilustración 17. Doble marca verde (Representación RGB - Representación RG)

Pez con marca verde y marca roja

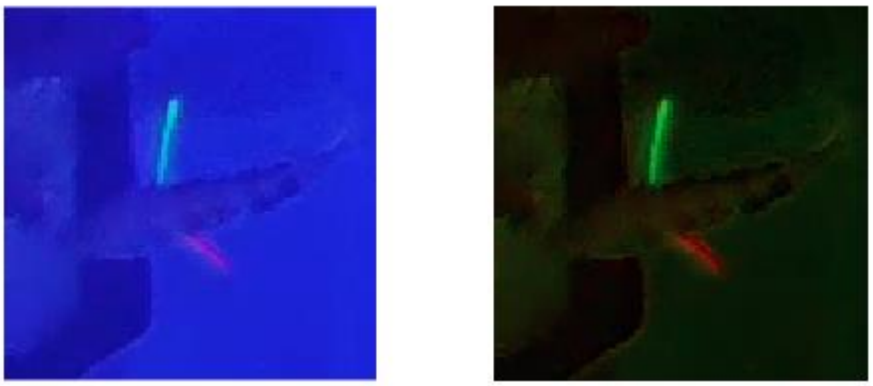

Ilustración 18. Marcas verde y roja (Representación RGB - Representación RG)

Sin embargo, no todas las imágenes obtenidas nos permiten llevar a cabo una clasificación, sino que algunas de ellas presentan situaciones que resultarán muy problemáticas a la hora de identificar al individuo. Éstas serán

descartadas tras aplicar el proceso de clasificación manual. Los dos principales motivos de estos inconvenientes son:

- Obtención de imágenes difuminadas: provocado por las capacidades de la cámara para captar al pez cuando éste se está moviendo a alta velocidad.

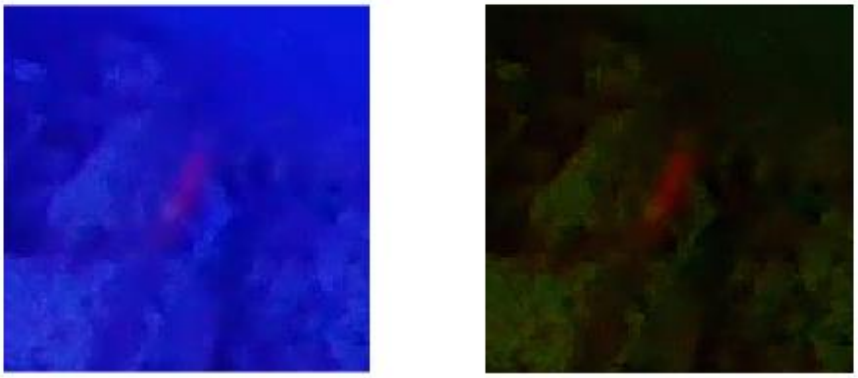

Ilustración 19. Ejemplo de imagen difuminada (Representaciones RGB - RG)

- Superposición de peces: los cruces de los peces en su recorrido provocan que una misma cámara los capte a los dos en la misma imagen, la cual no podrá ser clasificada con las etiquetas indicadas.

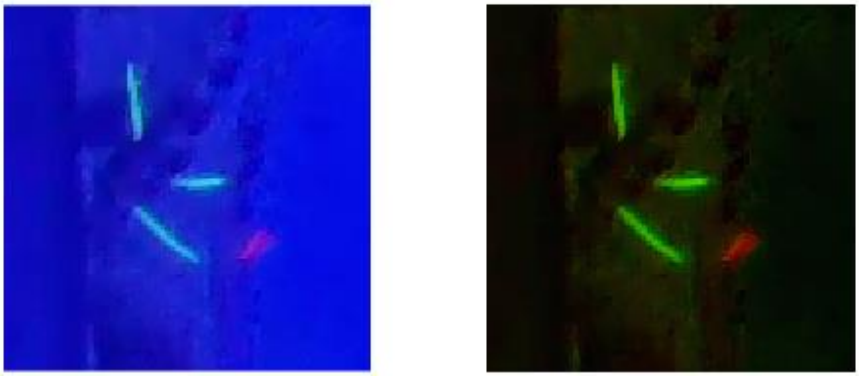

Ilustración 20. Ejemplo de superposición de peces (Representaciones RGB - RG)

Además, habrá imágenes generadas cuando el individuo se oculta en la zona de refugio y que, por tanto, no muestran al pez. Éstas también se descartarán.

Ilustración 21. Ejemplo de pez ocultándose (Representaciones RGB - RG)

3.3. Clasificación de imágenes

Por último, antes de pasar a ejecutar los algoritmos de aprendizaje desarrollados, era necesario etiquetar cada imagen para poder utilizarlas tanto como entrenamiento para los modelos como test para evaluar los mismos. La complejidad de esta tarea es que debe ser un humano quien, una a una, clasifique esas imágenes etiquetándolas en función de su criterio visual, por lo que resulta una tarea lenta y ardua.

Así pues, para facilitar esta labor, se desarrolló una interfaz gráfica con MATLAB que mediante la presentación de la imagen ofrezca diferentes botones con los que clasificar de forma ágil la imagen. Además, como muchas secuencias se van repitiendo, en determinados momentos de la clasificación ésta se vuelve más veloz.

La interfaz se compone de dos partes: la presentación y la botonera. En la primera, mediante un texto, se identifican las imágenes extraídas de la secuencia grabada y se expone la propia imagen tanto en RGB (composición de color Red-Green-Blue) como solo con sus componentes R (red) y G (green). En cuanto a la botonera, tiene cinco botones que se corresponden con las categorías que se desean extraer, uno que agrupará a todas las imágenes descartadas por los motivos descritos en el anterior apartado y, por último, un botón de pausa, para llevar a cabo tareas de debug y un botón de SaveClose para poder llevar a cabo la clasificación de todas las imágenes en varias ejecuciones sin perder la información.

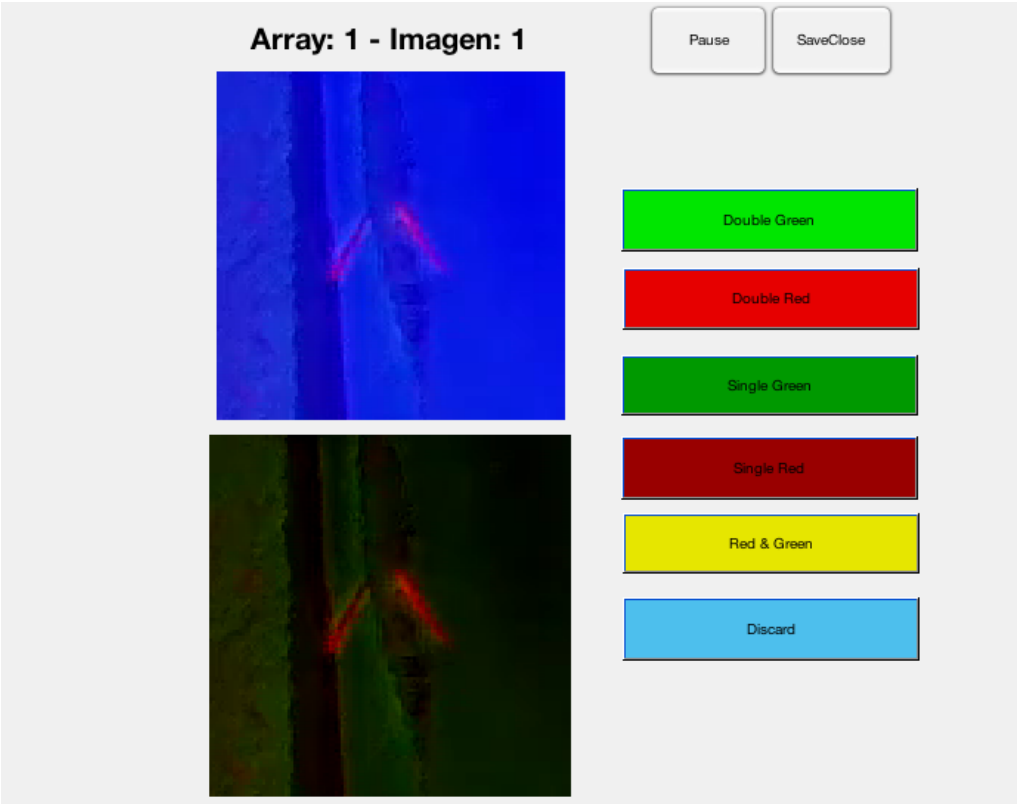

Ilustración 22. Interfaz gráfica para la clasificación manual

La funcionalidad es sencilla: se recorren mediante un bucle todas las imágenes disponibles para clasificar y se crean seis arrays donde ir introduciéndolas mediante diversos índices según el botón pulsado.

Tras aplicar este procedimiento manual, el volumen de imágenes inicial (40.032) se ha repartido de la siguiente forma en los arrays resultado:

- Pez con marca verde única: 3.692.
- Pez con marca roja única: 4.583.
- Pez con doble marca verde: 4.737.
- Pez con doble marca roja: 7.711.
- Pez con marcas roja y verde: 5.851.
- Imágenes descartadas: 13.458.

La última problemática que presentan las imágenes es la desigualdad en su tamaño. Existe un pequeño porcentaje de imágenes que son recortadas por el propio algoritmo de captura debido a encontrarse enfocando al borde del tanque. Éstas también han sido descartadas para su uso en los modelos de clasificación.

Por último, para poder trabajar con una distribución uniforme en todas las categorías, se han seleccionado 3.481 imágenes de cada una para utilizarlas como INPUT en los procesos de aprendizaje automático.

3.3.1. Estructuración de imágenes

Antes de pasar a exponer los diferentes modelos de clasificación desarrollados, es necesario disponer las imágenes de forma que su carga y su uso por parte de dichos modelos sean lo más ágil y sencillo posible. Para ello, en este punto se adelanta que en todos ellos se usarán las estructuras de MATLAB `imageDatastore` para administrar la colección de imágenes necesarias para el entrenamiento y la evaluación del modelo. Un almacén de imágenes de este tipo permite administrar datos de gran tamaño, incluidos los que no caben en la memoria, y leer lotes de imágenes de manera eficiente durante el entrenamiento de una red neuronal convolucional. Además, el imageDatastore permite etiquetar automáticamente las imágenes en función de los nombres de las carpetas donde están ubicadas.

En definitiva, a lo que se quería llegar con este apartado es a explicar la funcionalidad del script entregado: `images_structure.m`. El objetivo de sus comandos es recoger la estructura que resulta del clasificador manual y distribuirlas en carpetas según su categoría, de tal forma que queden dispuestas para crear el `imageDatastore` con un solo comando.

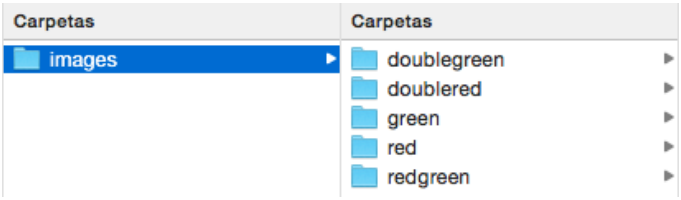

Ilustración 23. Distribución de imágenes requerida por imageDatastore

4. Modelos tradicionales

4.1. Modelo basado en el color aplicando SVM

El primer clasificador empleado se basa en dos elementos: el flujo de trabajo típico del aprendizaje máquina y el empleo de las máquinas de vector soporte.

En primer lugar, el flujo de trabajo propio de un clasificador [18] consta de las siguientes etapas:

- La carga de datos.
- El pre-procesado de los datos: preparación de un formato uniforme.
- Extracción de características.
- Entrenamiento del modelo.
- Iteraciones para mejorar el modelo.
- Integración del modelo en un entorno productivo.

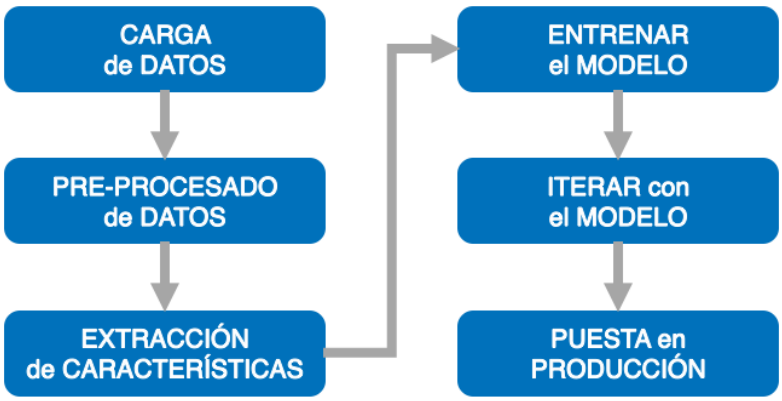

Ilustración 24. Flujo de trabajo de un clasificador

En este proceso, la mayoría de desafíos están relacionados con la selección adecuada de la información extraíble de los datos. Los conjuntos de datos pueden ser complejos y estar dispuestos en una variedad de formatos. Por ello, el pre-procesado de los mismos y la selección de las características apropiadas son pasos importantes en este flujo de trabajo sistemático del aprendizaje máquina. Los modelos altamente flexibles tienden a sobre ajustar los datos al modelar variaciones menores que podrían ser ruido, mientras que los modelos simples pueden asumir demasiado.

En el caso que nos ocupa, las imágenes ya han sido adecuadas a un mismo formato para su tratamiento, por lo que para este primer modelo se da por hecho el pre-procesado necesario de los datos. Además, como ya se indicó anteriormente, los individuos están marcados con combinaciones de colores, por lo que se seleccionará esta característica como eje del clasificador.

4.1.1. Extracción de características

El empleo del color como característica para entrenar el clasificador se llevará a cabo de una forma muy particular. La propiedad a utilizar en el clasificador posterior será la cantidad de regiones o áreas rojas y verdes existentes en cada imagen. Para poder obtener esta información se ejecutarán tres pasos:

1. Segmentación de imágenes: tiene como principal objetivo transformar la representación de la imagen en otra más útil y fácil de analizar. Este proceso servirá para identificar los pixeles verdes y rojos de cada una de las imágenes. Se han desarrollado dos métodos diferentes de segmentación cuyos resultados se compararán posteriormente.
2. Agrupación de píxeles: para poder definir áreas de un mismo color se lleva a cabo una erosión o relleno por difusión de la imagen que permite unir píxeles cercanos de cada color.
3. Conteo de regiones.

4.1.1.1. Segmentación mediante ‘Color Thresholder”

MATLAB pone a disposición del usuario la aplicación ‘Color Thresholder’, la cual ha sido empleada como primera alternativa para la segmentación de las imágenes. La aplicación consiste en una interfaz donde se carga una imagen en un formato de color seleccionado (RGB en este caso) y que permite regular los valores de los histogramas para acotar las regiones que contienen píxeles cuyos valores están en esa selección.

En la siguiente imagen se muestra un ejemplo de segmentación en la que moviendo los ejes de los histogramas se ha llegado a acotar de la mejor forma posible la marca roja del pez.

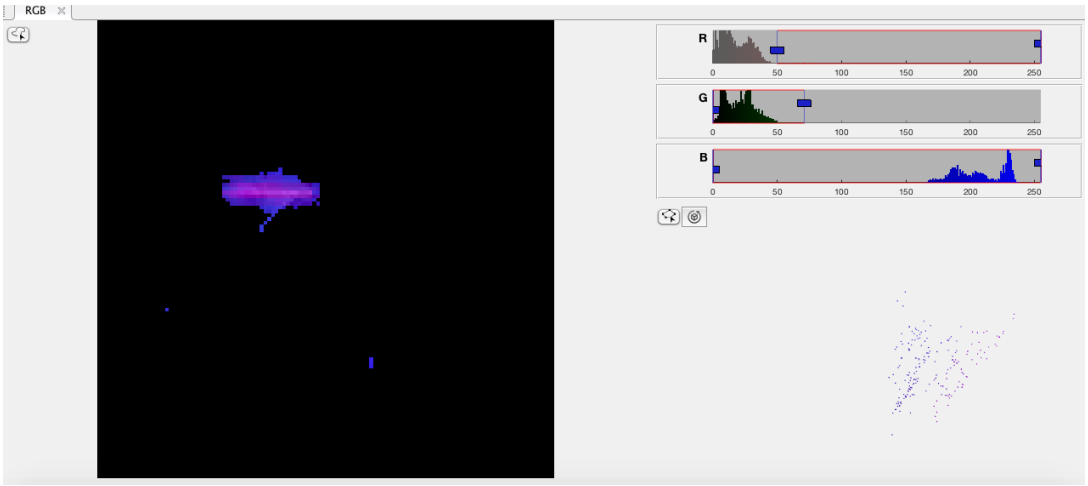

Ilustración 25. Ejemplo de uso de la app 'Color Thresholder"

Tras llevar a cabo esta segmentación sobre una selección representativa del conjunto global de imágenes, se han podido escoger los límites a emplear para acotar los píxeles que consideraremos ‘verdes’ y aquellos que

consideraremos ‘rojos’, tratándose ambos de excluyentes. En este punto se considera necesario aclarar que la componente azul no será utilizará por no aportar ningún valor a nuestro cometido. La selección de dichos límites ha obedecido a la mayor reducción de ruido posible.

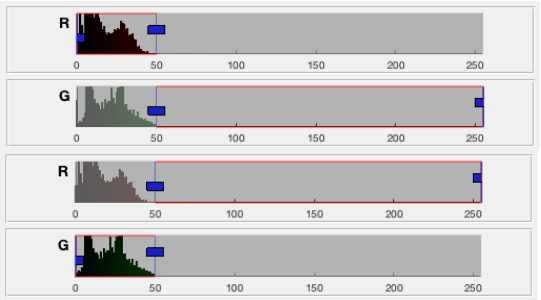

Ilustración 26. Límites seleccionados en el histograma RGB para la segmentación por color

Así pues, el objetivo principal de la función `colour_segment.m` es esta segmentación por color basada en la aplicación mencionada. Sin embargo, antes de devolver a la función principal de este modelo las representaciones binarias de rojo y verde, también lleva a cabo el relleno por difusión de las imágenes (floodfill).

El algoritmo de relleno consigue la agrupación de píxeles que se ha tratado en el apartado anterior. Para imágenes binarias, cambia los píxeles de fondo (0s) por píxeles de primer plano (1s) cuando están rodeados por semejantes. Esta operación termina cuando alcanza los límites del objeto.

A continuación, se muestra un ejemplo de imagen segmentada mediante la función `colour_segment.m`.

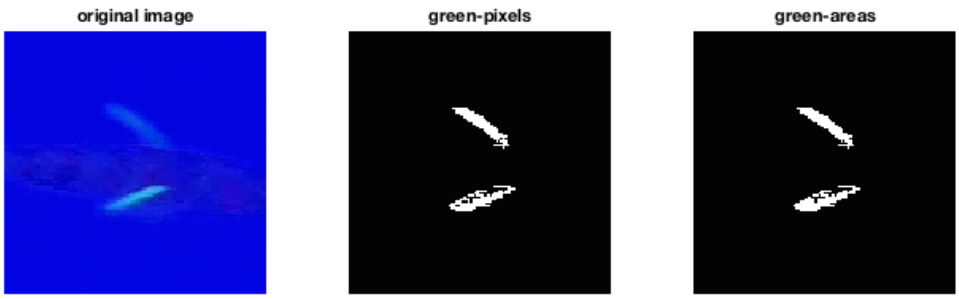

Ilustración 27. Ejemplo de segmentación y relleno por difusión

4.1.1.2. Segmentación basada en el valor de la luminancia

La segunda alternativa desarrollada para obtener las imágenes segmentadas en función del color está basada en el concepto de luminancia. La luminancia es la cuantificación del brillo de un objeto, característica que puede producir que un objeto con tonalidad diferente se observe similar por esa sensación lumínica. El valor de la misma en cada pixel en una imagen representada en RGB se puede obtener a través de la siguiente fórmula:

$$Y = 0.2989 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

El objetivo es utilizar el formato YCbCr y aplicar los umbrales a la crominancia diferencia de rojo y a su equivalente en verde, aunque ésta no sea utiliza en el formato antes indicado. Estas componentes permiten eliminar los colores que surgen de la combinación de R, G y B y limitarnos a evaluar los colores puros rojo y verde. Para entender este concepto visualmente es un buen apoyo la siguiente imagen, donde se muestra como varía el plano de crominancias (en este caso Cb y Cr, aunque se utilizarán Cr y Cg en el modelo) según los valores de luminancia.

Ilustración 28. Plano Cb-Cr con diferentes valores de Y (Fuente: wikipedia)

Una vez aplicados los umbrales sobre las componentes C_R y C_G se utilizará la operación de cierre para la generación de las regiones. Mientras que en la segmentación en base a la aplicación 'Color Thresholder' se ha utilizado el algoritmo de relleno, en este caso se utiliza la combinación de dilatación y erosión de la imagen con un mismo elemento estructurante. Éstas son dos de las operaciones fundamentales en el área de la morfología matemática y se aplican a imágenes binarias. En primer lugar, la dilatación permite que pasen a formar parte del objeto aquellos píxeles o pequeños grupos de ellos que sabemos que deberían pertenecer al objeto pero que han quedado aislados por diferentes motivos. En lo referente a la erosión, su efecto básico es el erosionar los límites de las regiones de los píxeles del primer plano (1s), con el resultado de que las áreas con estos píxeles reducen su tamaño y el fondo se hace más grande (0s). La consecuencia directa de estas operaciones es eliminar ruido y dejar como primer plano áreas compactas.

A continuación, se muestra un ejemplo de imagen segmentada mediante la función `luminance_segment.m`.

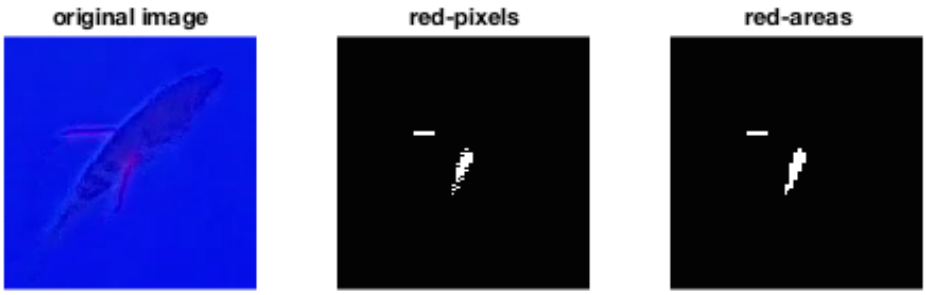

Ilustración 29. Ejemplo de segmentación y closing

Por último, indicar que la elección de umbrales para esta segmentación se ha hecho iterando el modelo para concluir con qué valores se obtenían mejores resultados.

		UMBRAL ROJO				
		1	2	3	4	5
UMBRAL VERDE	1	0.9017	0.9040	**0.9079**	0.9046	0.9000
	2	0.9034	0.9050	0.8987	0.9033	0.8973
	3	0.8971	0.9000	0.8950	0.8954	0.8831
	4	0.8908	0.8841	0.8929	0.8952	0.8780
	5	0.8862	0.8866	0.8830	0.8814	0.8785

Tabla 3. Selección de umbrales para `luminance_segment`

4.1.1.3. Conteo de regiones

Tras segmentar las imágenes y agrupar los píxeles de primer plano, de una u otra forma, el último paso para obtener la característica a emplear en la máquina de vector soporte es calcular el número de áreas de cada color existente en cada imagen. Para ello es necesario emplear la función `count_areas.m`. Ésta se basa en las funciones de la Image Processing Toolbox de MATLAB: bwlabel y regionprops.

- bwlabel: etiqueta componentes conectados en una imagen binaria. El primer valor que devuelve es una matriz que contiene esas etiquetas, el segundo es el número de regiones conectadas encontradas en la imagen.
- regionprops: a partir de la matriz de etiquetas es capaz de extraer las propiedades solicitadas de cada una de las regiones encontradas.

Una vez obtenidas todas las regiones y sus propiedades se considerarán efectivas para nuestro fin aquellas que sean mayores a un determinado umbral, cuyo valor óptimo se calculará en las ejecuciones posteriores de este modelo.

4.1.2. Máquinas de Vector Soporte (SVM)

Las SVM fueron originalmente ideadas para la resolución de problemas de clasificación binaria en los que las clases eran linealmente separables. Su objetivo final es construir un hiperplano que separe de forma óptima los datos de las dos clases que comparten espacio. La selección óptima del hiperplano pasa por elegir el que mejor represente el límite entre los dos conjuntos, es decir, el que maximiza el margen entre las muestras de cada clase respecto a la frontera de separación [19]. A partir de esta situación, surgen otras en la que, debido a la distribución de los datos, no es posible separar linealmente los datos, dando lugar a nuevas versiones de SVM. Cuando no es posible la separación de las clases por el solapamiento de las mismas la solución pasa por encontrar el hiperplano que cometa el menor número de errores posibles. E, incluso, en ocasiones, la solución lineal no proporciona un buen resultado, por lo que se recurre a la proyección de los datos a un espacio de mayor dimensión donde la solución lineal sí es válida.

En el caso que nos ocupa, al tratarse de una clasificación múltiple, se recurre al modelo ECOC (Error-Correcting Output Codes), el cual reduce el aprendizaje multi-clase al empleo de múltiples clasificadores binarios SVM. La función de MATLAB que nos permite entrenar un modelo SVM de estas características es `fitcecoc` [20]. Con este modelo será posible llevar a cabo una predicción de las imágenes reservadas para el test y con ello obtener el resultado de precisión que proporciona el proceso.

El algoritmo se puede ejecutar a través de `areas_svm.m`.

4.2. Modelo basado en el clasificador de imágenes de MATLAB

El segundo procedimiento basado en SVM que se va a emplear para comparar Deep Learning con lo que hemos llamado "modelos tradicionales" se basa en la clase de MATLAB `imageCategoryClassifier`. Este objeto contiene un clasificador SVM entrenado para reconocer diferentes categorías de imágenes.

Para poder verificar el rendimiento de este clasificador, se ha implementado el ejemplo que MATLAB proporciona en su definición [21], el cual sigue los pasos:

1) Carga de imágenes y separación de las mismas en conjuntos de entrenamiento y test.
2) Extracción de características. Usa la función `bagOfFeatures`, la cual recibe un conjunto de imágenes y devuelve características de las mismas tras aplicar diversos métodos (matricial, K-means, etc.).
3) Entrenamiento del clasificador mediante la función `trainImageCategoryClassifier`, el cual requiere de las imágenes y las características extraídas.
4) Evaluación del modelo.

El algoritmo se puede ejecutar a través de `svm.m`.

5. Deep Learning

5.1. Transferencia de aprendizaje: AlexNet

Tal y como se exponía en el apartado 2.3, una de las estrategias que se puede seguir para aplicar Deep Learning es la transferencia de aprendizaje. Consiste en tomar una red pre-entrenada y usarla como punto de partida para aprender una nueva tarea. La ventaja de este enfoque es que la red preestablecida ya ha aprendido un amplio conjunto de características que podrán ser aplicadas con fines similares. Para desarrollar este planteamiento se ha seleccionado AlexNet.

AlexNet es una red neuronal convolucional pre-entrenada que compitió en ImageNet en 2012, un desafío a gran escala de reconocimiento visual en el que consiguió un error del 15.3%, 10.8 puntos por encima del otro finalista. En MATLAB, `net = alexnet` proporciona un modelo entrenado por un subconjunto de la base de datos que se utilizó en el desafío. Este modelo está entrenado por más de un millón de imágenes y puede clasificar objetos en hasta 1000 categorías, por lo que contiene características enriquecidas para una amplia gama de imágenes.

La red contiene ochos capas: las cinco primeras son convolucionales, mientras que las tres son FCL [22]. Si en MATLAB se ejecuta el comando `net.Layers` se puede observar el contenido de sus capas, en las que se explican las particularidades de cada una. Su arquitectura se muestra en la siguiente imagen.

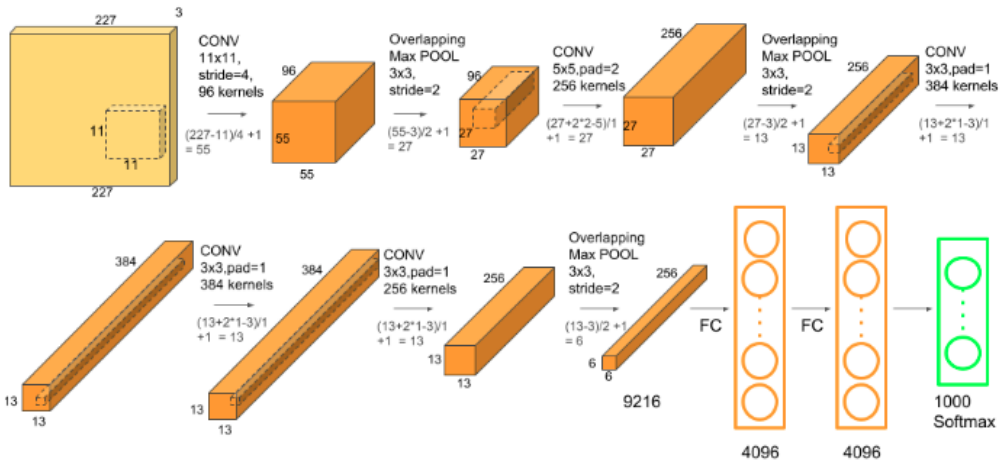

Ilustración 30. Arquitectura de AlexNet [24]

Para entender cómo se extraen características a través de esta red neuronal, es conveniente ver cómo secuencialmente, a lo largo de esta estructura, la imagen sufrirá una serie de transformaciones que permitirán reducir el tamaño de los datos que se manejan y representar la imagen con un conjunto acotado de información que la caracterice. En el caso de las capas de convolución la transformación que se producirá sigue el siguiente patrón [24]:

$$O = \frac{I - K + 2P}{S} + 1$$

donde:

- O = tamaño de salida
- I = tamaño de entrada
- K = tamaño de kernels
- P = tamaño del padding (relleno de ceros)
- S = tamaño del paso de convolución (stride)

Lo que conlleva que, por ejemplo, la primera capa de convolución, en la que se introducen las imágenes con tamaño 227x227x3, tenga una salida de 55x55x96:

$$O = \frac{227 - 11 + 2 \cdot 0}{4} + 1 = 55$$

En cuanto a las capas de pooling, la conversión viene dada por:

$$O = \frac{I - P_S}{S} + 1$$

donde:

- O = tamaño de salida
- I = tamaño de entrada
- P_S = tamaño del filtro de pooling
- S = tamaño del paso de convolución (stride)

Así pues, en la primera capa de maxPool que nos encontramos, la transformación será de 55x55x96 a 27x27x96:

$$O = \frac{55 - 3}{2} + 1 = 27$$

5.1.1. Modelo desarrollado con transferencia de aprendizaje

Una vez cargado el modelo de AlexNet, los pasos para construir su adaptación a nuestros objetivos son los siguientes [23]:

1) Creación del objeto imageDatastore donde se almacenan las imágenes con sus etiquetas asociadas.

2) Dividir los datos en conjuntos de entrenamiento y test. El primero usará el 70% de las imágenes y el segundo el 30% restante.

3) Adaptación de las características de la imagen a lo que requiere la capa de entrada de la red.
   - Reajustar el tamaño para que coincida con el tamaño que espera la red: 227x227x3.
   - Voltear aleatoriamente las imágenes de entrenamiento sobre el eje vertical y, de forma aleatoria también, trasladarlas 30 píxeles horizontalmente y verticalmente. Estas operaciones se llevan a cabo para evitar que la red se sobreajuste y memorice los detalles de las imágenes de entrenamiento.

4) Reemplazar dos de las últimas capas de la red:
   `23: 'fc8'    Fully Connected → Fully Connected para 5 categorías`
   `25: 'output' Classif. Output → Classification Layer`

5) Entrenar la red. En el aprendizaje de transferencia se mantienen los pesos de las primeras capas de la red preestablecida. Los parámetros para configurar son los siguientes [25]:
   - MiniBatchSize: Tamaño de lote. Cantidad de imágenes de entrenamiento que se utilizan en cada iteración para evaluar el gradiente de la función de pérdida y optimizar los pesos de la red.
   - MaxEpochs: Un 'epoch' es un ciclo de entrenamiento completo en todo el conjunto de datos de entrenamiento. Para la transferencia de aprendizaje no hace falta que este valor sea elevado.
   - InitialLearnRate: Un valor bajo ralentiza el aprendizaje de las capas transferidas, pero garantizar que no se alcance un resultado subóptimo.

6) Clasificación de imágenes de test

7) Cálculo de precisión

El script que sigue estos pasos es `transfer_alexnet.m`.

5.2. Extracción de características: AlexNet

Existe una segunda alternativa de utilizar AlexNet para mejorar la precisión de clasificación que es utilizar la red para extraer las características aprendidas de las imágenes y, con éstas, entrenar un clasificador. Esta estrategia es la forma más fácil y rápida de usar las capacidades de una red neuronal profunda. Esto se debe a que la extracción de características no requiere de iteraciones y solo necesita recorrer una vez los datos, de manera que es un buen punto de partida si no se dispone de suficiente GPU para acelerar el entrenamiento de la red [27].

5.2.1. Modelo desarrollado en base a características aprendidas

El algoritmo basado en este enfoque de reutilización de AlexNet consta de dos bloques diferenciados. El primero es la propia extracción de características de la red neuronal, el segundo es el entrenamiento de una SVM usando la función `fitcecoc` de MATLAB. El primer paso, por tanto, será cargar la red preestablecida y, a partir de ella, ejecutar las siguientes tareas:

1) Creación del objeto imageDatastore donde se almacenan las imágenes con sus etiquetas asociadas.

2) Dividir los datos en conjuntos de entrenamiento y test. El primero usará el 70% de las imágenes y el segundo el 30% restante.

3) Adaptación del tamaño de las imágenes al requerido por la red neuronal.

4) Extracción de características. La red neuronal construye una representación jerárquica de las imágenes de entrada. En ella las capas más profundas contienen características de mayor nivel, construidas con las características de nivel inferior de las capas anteriores. Así pues, para operar con esas características de mayor nivel se toma la salida de la capa 'fc7', penúltima FCL. Para ello se utiliza el comando `activations`.

5) Entrenamiento del clasificador SVM

6) Clasificación de imágenes de test

7) Cálculo de precisión

La función a ejecutar para resolver este modelo es `alexnetFE_svm.m`.

5.3. Creación de red neuronal

La última estrategia a implementar para comprobar lo que puede aportar Deep Learning en el escenario que nos ocupa es la de entrenar una red neuronal desde cero, creada especialmente para este propósito.

La única diferencia de este proceso y la transferencia de aprendizaje de AlexNet es que, para definir la arquitectura de la red neuronal, se habrá de definir la estructura de capas (layers) que en la transferencia se extraían de la red preestablecida. Por lo demás, las opciones de entrenamiento y el entrenamiento en sí se lleva a cabo con los mismos comandos que en el dicho modelo [28].

Así pues, la arquitectura desarrollada es la típica de una CNN. Para empezar, la primera capa de ese modelo debe ser la de entrada de imágenes. Para definirla es necesario indicar el tamaño que se esperará de las imágenes (ancho y alto) y el número de canales de las mismas, que en el caso de las imágenes RGB con las que estamos trabajando será 3.

Una vez que la red dispone de los datos con los que trabajar, se inician las etapas donde la operación más importante llevada a cabo es la de convolución. Este bloque está formado por cuatro tipos de capas y se repetirá, en nuestro modelo, hasta en cinco ocasiones. Las capas por las que está formado son:

- Capa de convolución: esta capa se encarga, en primer lugar, de agregar un relleno de ceros alrededor de los datos de entrada, con el fin de que no pierda valor la información almacenada en los bordes. Tras ello, ejecuta el filtro de tamaño indicado en el primer parámetro de la función (en nuestro caso será 3x3). El segundo parámetro marca el número de filtros, es decir, el número de neuronas que se conectan a la misma región de entrada. En las cinco capas presentes de convolución en la red neuronal creada el tamaño del filtro irá aumentando para conseguir que

gradualmente el número de datos con los que se trabaje sea menor y, por tanto, más representativo y manejable.

- Capa de normalización: facilitan la optimización del entrenamiento de la red normalizando, por lotes, los gradientes que se propagan por ella. Esta capa se incluye entre la convolucional y la capa no lineal a la que precede para acelerar el entrenamiento de la red y reducir la sensibilidad de la misma a los valores iniciales.
- ReLU: función de activación no lineal.
- Capa de agrupación: por último, el pooling se utiliza para muestrear y reducir así el tamaño espacial del mapa de características, eliminando la información redundante. Esta capa permite que, en las capas convolucionales posteriores, aumentar la cantidad de filtros sin la necesidad de aumentar los cálculos. En nuestro caso se utilizará maxPooling con un filtro de 2x2, de manera que, manteniendo un número de cálculos constante, cada capa convolucional duplicará el tamaño del filtro que utiliza.

Por último, tras las cinco etapas citadas, se incluye el cierre a la arquitectura, compuesto por:

- Capa completamente conectada: en ella todas las neuronas se conectan a todas las neuronas de la capa anterior. Con ello se consigue combinar todas las características extraídas de capas anteriores para identificar los patrones más importantes de las imágenes. La salida de esta capa permitirá clasificar las imágenes, lo que tiene relación con el parámetro OutputSize que recibe la función, el cual deberá ser igual al número de clases con el que se está trabajando (en este caso, 5).
- Capa Softmax: sirve para normalizar la salida de la capa completamente conectada. La salida de esta capa consiste en números positivos que suman uno, lo que la siguiente capa utilizará como probabilidades de clasificación.
- Capa de Clasificación: emplea las probabilidades que devuelve la función de activación de la capa Softmax para asignar la clase correspondiente y calcular la pérdida.

La estrategia seguida en la construcción de todas estas capas se apoya en el objetivo de reducir los tiempos de ejecución del enfoque de transferencia de aprendizaje. Para ello se han planteado dos tácticas, ambas relacionadas con la cantidad de datos con las que trata la red, pretendiendo reducirla para agilizar las operaciones realizadas y, con ello, el tiempo de ejecución. La primera manera de conseguirlo es emplear un tamaño de imagen menor al original, mientras que la segunda consiste en aumentar en cada bloque convolucional el tamaño del filtro para que la reducción de datos sea mayor de forma gradual.

Finalmente, el modelo se compone de los siguientes pasos:

1) Creación del objeto imageDatastore donde se almacenan las imágenes con sus etiquetas asociadas.

2) Dividir los datos en conjuntos de entrenamiento y test. El primero usará el 70% de las imágenes y el segundo el 30% restante.

3) Adaptación del tamaño de las imágenes al requerido por la red neuronal. Mientras que en AlexNet se usaban imágenes de 227x227 en RGB, en esta arquitectura creada ad hoc se adaptarán las imágenes a un tamaño de 100x100. Esta reducción se considera mínimo, por lo que no afectará al resultado, y permite al modelo ser más ágil y rápido en sus cálculos al estar tratando menos información.

4) Definición de la arquitectura de la red descrita anteriormente.

5) Entrenamiento de la red. Se utilizan los mismos parámetros que en el modelo de transferencia de aprendizaje. Entre otros motivos, para poder comparar ambos procesos en igualdad de condiciones. Los parámetros a configurar son los siguientes [25]:
   - MiniBatchSize: Cantidad de imágenes utilizadas en cada iteración para la operación de optimización de pesos.
   - MaxEpochs: Número de ciclos de entrenamiento.
   - InitialLearnRate: Tasa que regula la velocidad de aprendizaje de la red.

6) Clasificación de imágenes de test

7) Cálculo de precisión

Este proceso se puede encontrar en el entregable `DP_network.m`.

Para terminar, el siguiente diagrama ilustra las diferencias entre los diferentes modelos desarrollados de Deep Learning.

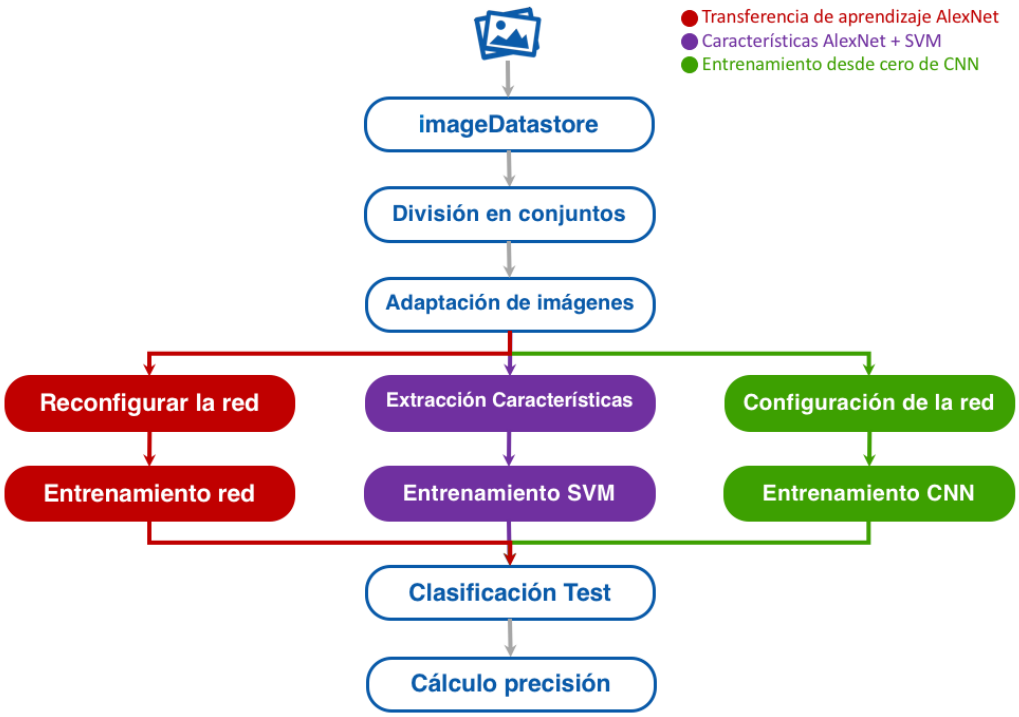

Ilustración 31. Diagrama resumen de los diferentes enfoques de Deep Learning

6. Resultados

En este apartado de la memoria del trabajo se describen las diferentes ejecuciones que se han llevado a cabo de todos los modelos desarrollados, con el fin de comparar el rendimiento de los mismos en términos de tiempo de ejecución y, sobre todo, de precisión. Además, para poder conocer en cada modelo si los resultados mejoran o no a mayor número de muestras empleadas en el entrenamiento, se harán ejecuciones con la totalidad de los datos disponibles (3.481 imágenes de cada categoría) y con un subconjunto en el que se han seleccionado 1.000 imágenes de cada clase de forma aleatoria.

En cuanto a la comparación respecto al tiempo de ejecución, se considera necesario resaltar que se han ejecutado los modelos con equipos de diferentes características, pero los datos recopilados en la memoria se corresponden con los lanzamientos hechos desde un ordenador de estas características:

Ilustración 32. Propiedades del equipo utilizado para las ejecuciones

Por último, también resulta conveniente indicar que, para poder ejecutar las funciones desarrolladas es necesario tener disponibles tanto el modelo de AlexNet como las siguientes librerías de MALTAB:

- Image Processing Toolbox.
- Statistics and Machine Learning Toolbox.
- Computer Vision System Toolbox.
- Neural Network Toolbox.

6.1. Ejecución de SVM basada en el color

Para la ejecución de la máquina de vector soporte que utiliza el conteo de regiones como característica se lanzará la función `areas_svm`, la cual requiere de tres parámetros:

- Directorio del que extraer el conjunto de imágenes a utilizar.
- Tipo de segmentación a emplear. Basada en color RGB o en luminancia.
- Umbral: área a partir de la cual una región se considera suficientemente grande para contar en el total calculado.

Los resultados de precisión obtenidos son los recogidos en la tabla 4. Sin embargo, para que la comparación resulte, de forma visual, más sencilla, se recomienda visualizar los gráficos que se muestran tras ella. En ellos se ha recurrido a las columnas agrupadas para representar conjuntamente el

resultado que, para cada umbral, se obtiene utilizando uno u otro conjunto de imágenes.

Por último, aclarar que tanto estos resultados como los que les siguen en apartados siguientes son una media del rendimiento de los procesos en varias ejecuciones, con el fin de evitar mostrar sub-óptimos locales al emplear una única ejecución.

	COLOR THRESHOLDER		LUMINANCIA	
	$IMAG_{1000}$	$IMAG_{3481}$	$IMAG_{1000}$	$IMAG_{3481}$
0	0.4947	0.4593	0.8873	0.8987
5	0.8047	0.6825	**0.9013**	**0.9137**
10	0.84	0.8378	0.8787	0.9010
15	**0.8513**	**0.8532**	0.886	0.8784
20	0.842	0.8484	0.8587	0.8698
25	0.8393	0.8367	0.824	0.8513
30	0.8347	0.8369	0.8293	0.8423
35	0.828	0.8294	0.824	0.8239
40	0.8187	0.8172	0.7953	0.8141
45	0.804	0.8115	0.7853	0.8038
50	0.8107	0.8017	0.7827	0.7928

Tabla 4. Resultados de SVM basada en el color según el umbral

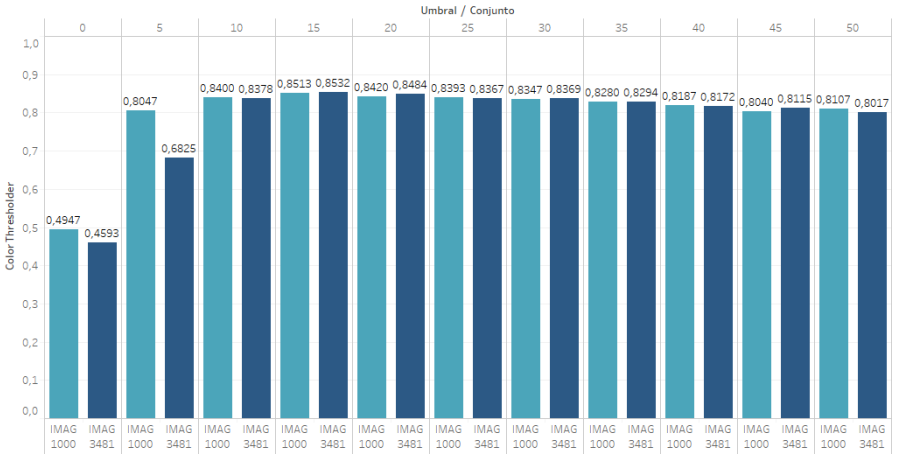

Ilustración 33. Resultado SVM basado en el umbral de color

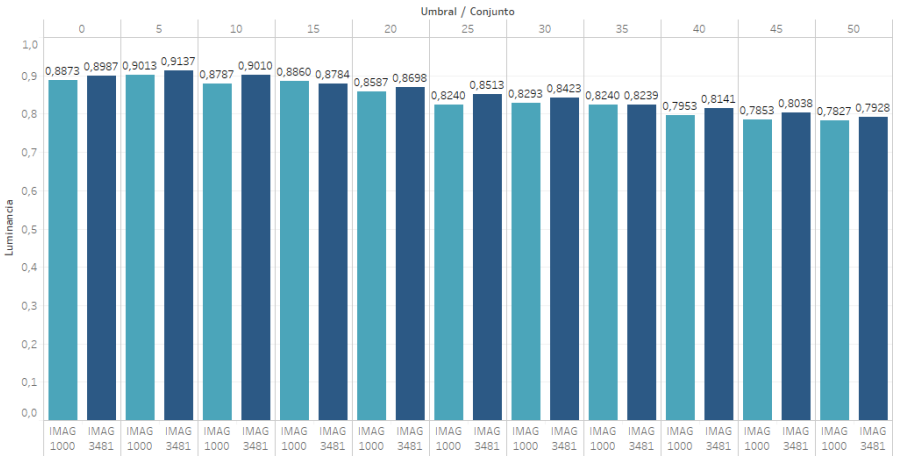

Ilustración 34. Resultados de SVM basada en luminancia

Las conclusiones extraídas de estos gráficos son:

- Debido a la forma de segmentar de ambos métodos, las áreas resultantes en el proceso basado en luminancia son menores que las calculadas por los umbrales del color. La consecuencia de eso es que el tamaño de área para el que se encuentran mejores resultados es menor en el primero método citado que en el segundo.
- En cuanto a los conjuntos de muestras, se aprecia cómo los resultados mejoran al disponer de más imágenes para el entrenamiento del modelo. Así, mientras que para el método que utiliza segmentación basada en el umbral de color dicha mejora es menor, para el que está basado en la luminancia aumenta el rendimiento en más de un 1%.
- En términos generales, nos ofrece mayor rendimiento la segmentación que utiliza la luminancia. En concreto, para los puntos óptimos de ambos modelos, consigue más de un 5% de mejora respecto a los resultados presentados por la segmentación que se apoya en umbrales de color.

Por último, en cuanto al tiempo de ejecución, se han empleado medias de todas las ejecuciones necesarias para extraer los anteriores resultados y, como se puede observar en la siguiente tabla, cuando se emplean pocas muestras las diferencias entre los dos procesos no son significativas, pero se empiezan a hacer latentes al emplear el conjunto de 3.481 muestras. Además, si se calcula una relación sencilla se puede deducir que, al utilizar más imágenes, el tiempo empleado en estos métodos no es directamente proporcional a la ampliación del conjunto, sino que es mayor.

	COLOR THRESHOLDER	LUMINANCIA
$IMAG_{1000}$	43''	42''
$IMAG_{3481}$	3' 46''	2' 55''

Tabla 5. Tiempo medio de ejecución de SVM basada en el color

6.2. Ejecución de clasificador de imágenes de MATLAB

El modelo que utiliza el clasificador de imágenes de MATLAB solo requiere del nombre del directorio donde se encuentran las imágenes que utilizará para entrenamiento y evaluación del proceso.

Así pues, los resultados medios obtenidos con este modelo, tanto en términos de precisión como de tiempos de ejecución, utilizando los dos conjuntos de imágenes disponibles, son los siguientes:

	Precisión	Tiempo medio
$IMAG_{1000}$	0.8464	30' 25''
$IMAG_{3481}$	0.806	31h 16'

Tabla 6. Resultados de precisión y tiempo del clasificador de imágenes de MATLAB

En base a esta información podemos deducir dos cosas. La primera y más evidente, que el tiempo de ejecución de este modelo cuando el conjunto de

imágenes disponibles es muy grande se dispara exponencialmente, haciendo inviable su uso en una aplicación de monitorización continua. La segunda, que se produce un efecto de sobreajuste (overfitting) en el modelo cuando se utiliza el conjunto de más imágenes. Éste consiste en que el modelo se adapta a las imágenes de entrenamiento de una forma muy ajustada, provocando el empeoramiento progresivo de sus resultados ante nuevos datos de entrada.

6.3. Ejecución de la transferencia de aprendizaje (AlexNet)

Tal y como se expuso en la descripción del proceso de transferencia de aprendizaje basado en la red AlexNet, existen varios parámetros a configurar para el entrenamiento de la red. El impacto de los mismos se pretende medir a través de diferentes ejecuciones del modelo. De esta manera, es necesario facilitarle a la función `transfer_alexnet` la siguiente información:

- Directorio del que extraer el conjunto de imágenes a utilizar.
- Tamaño del lote de imágenes: Número de imágenes que se utilizan en cada iteración para optimizar la red.
- Número de iteraciones. Número de ciclos de entrenamiento. Su valor variará para cada ejecución.
- Tasa de entrenamiento inicial. Será un valor fijo (0.001) utilizado para todas las ejecuciones.

La operativa de ejecución se ha basado en dos barridos. En primer lugar, para un valor fijo de iteraciones se han realizado ejecuciones para diferentes valores del tamaño de lote de imágenes, obteniendo los resultados para los dos conjuntos de imágenes disponibles (tabla 6). Una vez extraídos estos datos, se selecciona el valor de tamaño de lote con el que se obtienen de los mejores ratios (64) y se lleva a cabo el barrido para diferentes valores de número de iteraciones. Los resultados de esta segunda serie de ejecuciones se muestran en la tabla 7 para ambos conjuntos de datos de entrada.

	$IMAG_{1000}$	$IMAG_{3481}$
16	0.9567	0.9833
32	0.9647	0.9856
64	0.9793	0.9864
128	0.9673	0.9862
256	0.96	0.9849

Tabla 7. Resultados transferencia de aprendizaje según el tamaño del lote (epochs=5)

	$IMAG_{1000}$	$IMAG_{3481}$
5	0.9793	0.9864
10	0.9873	0.992
15	0.9813	0.9904
20	0.9827	0.9914
25	0.982	0.9816

Tabla 8. Resultados transferencia de aprendizaje según el número de iteraciones (lote=64)

Las principales conclusiones que podemos sacar en base a estos datos son:

- Cuanta más información dispone la red, mejores resultados se obtienen. Es por ello por lo que con el conjunto de 3.481 imágenes se logran mejores ratios de precisión.
- Al principio, cuando se aumenta el tamaño del lote para un mismo número de iteraciones, la precisión de la clasificación mejora. Esto se debe a que aumenta la cantidad de análisis efectuados sobre las imágenes de entrenamiento durante la etapa de aprendizaje. Sin embargo, llega un punto en el que la precisión vuelve a bajar por el hecho de emplear demasiadas imágenes para esa optimización.
- En cuanto al impacto de variar el número de iteraciones, un mayor número de ellas aumenta las probabilidades de éxito de la clasificación hasta un punto en el que se produce la caída del rendimiento. En teoría, aunque no se ha podido demostrar empíricamente, esta caída llegará un momento en que será abrupta, generándose el efecto conocido como de las "neuronas muertas". Éste consiste en que, debido a valores anómalos en los pesos de la red o a una tasa de aprendizaje inadecuada, las neuronas calculan siempre el mismo valor de salida independientemente del valor de entrada [29].

En cuanto a los tiempos que este proceso tarda en ejecutarse, las siguientes tablas son reflejo de las anteriores, pero sustituyendo el valor que representan: tiempo por precisión.

	$IMAG_{1000}$	$IMAG_{3481}$
16	58' 54''	3h 17' 1''
32	47' 4''	2h 43' 17''
64	42' 29''	2h 28' 41''
128	43' 21''	3h 2' 5''
256	1h 3' 9''	2h 56' 37''

Tabla 9. Tiempos de ejecución de la transferencia de aprendizaje según el tamaño del lote (epoch=5)

	$IMAG_{1000}$	$IMAG_{3481}$
5	42' 29''	2h 28' 41''
10	1h 28' 55''	5h 9' 51''
15	2h 11' 59''	7h 45' 8''
20	3h 1' 18''	10h 5' 15''
25	3h 43' 18''	12h 31' 47''

Tabla 10. Tiempos de ejecución de la transferencia de aprendizaje según el número de iteraciones (lote=64)

Al respecto comentar que el comportamiento es el esperado cuando se aumenta el número de iteraciones, puesto que a mayor número de ellas se advierte un crecimiento aproximadamente proporcional del tiempo de ejecución. Sin embargo, cuando el número de imágenes empleadas en la optimización aumenta, ese incremento no se ve reflejado en los tiempos de ejecución, sino que estos son más similares entre sí y la variación es

irregular. Esto se debe a que la optimización se realiza con análisis en bloque, no de forma secuencial como con las iteraciones. Por último, se repite el patrón de que, a mayor número de imágenes disponibles como entrada, los tiempos aumentan de una forma exponencial y no directamente proporcional.

6.4. Ejecución de SVM basado en las características extraídas de AlexNet

Como ya se explicó en el apartado en el que se presentaba este modelo, se puede utilizar una red neuronal para extraer la salida de una de sus capas y utilizarla como las características que representan al conjunto de cara entrenar una SVM. Este enfoque no aplica iteraciones, lo que conlleva dos consecuencias. La primera, que para ejecutarlo será suficiente con indicarle el nombre del directorio del cual tomar las imágenes a utilizar. La segunda, que no requiere de ningún tipo de parametrización para llevarse a cabo, por lo que su ejecución no es configurable y, a pesar de ello, ofrece unos resultados muy positivos tanto en términos de precisión como de tiempos de ejecución.

	Precisión	Tiempo medio
$IMAG_{1000}$	0.9596	4’ 2’’
$IMAG_{3481}$	0.9752	14’ 15’’

Tabla 11. Resultados para la extracción de características de una red pre-entrenada

Resultados que permiten confirmar el, a priori, buen resultado que ofrece esta alternativa y, de nuevo, ratificar cómo a mayor número de imágenes disponibles a la entrada se obtiene mejor rendimiento.

6.5. Ejecución de CNN creada desde cero

El último de los procesos a ejecutar es la red neuronal configurada desde cero para la tarea particular de clasificar las imágenes capturadas de los peces. Debido a que el resultado será directamente facilitado por la propia red, necesitamos, como en el caso de la transferencia de aprendizaje, facilitarle al modelo una serie de parámetros, los cuales se repiten:

- Directorio del que extraer el conjunto de imágenes a utilizar.
- Tamaño del lote de imágenes: Número de imágenes que se utilizan en cada iteración para optimizar la red.
- Número de iteraciones. Número de ciclos de entrenamiento. Su valor variará para cada ejecución.
- Tasa de entrenamiento inicial. Será un valor fijo (0.001) utilizado para todas las ejecuciones.

También de nuevo la operativa consistirá en realizar un barrido con diferentes valores de tamaño de lote de imágenes, seleccionar aquel con el que mejores resultados se obtengan y, finalmente, efectuar ejecuciones variando el valor del número de iteraciones. El rendimiento, medido tanto por

la precisión como por el tiempo, se recoge en las siguientes cuatro tablas. Indicar que el tamaño de lote seleccionado en esta ocasión ha sido de 32.

	$IMAG_{1000}$	$IMAG_{3481}$
16	0.9693	0.9816
32	0.9847	0.9879
64	0.9833	0.9847
128	0.98	0.9852
256	0.968	0.987

Tabla 12. Resultados CNN desde cero según el tamaño del lote (epochs=5)

	$IMAG_{1000}$	$IMAG_{3481}$
5	0.9847	0.9879
10	0.984	0.9923
15	0.9753	0.9904
20	0.9847	0.9908
25	0.9847	0.9906

Tabla 13. Resultados CNN desde cero según el número de iteraciones (lote=32)

	$IMAG_{1000}$	$IMAG_{3481}$
16	10’ 46’’	35’ 55’’
32	9’ 13’’	32’ 57’’
64	8’ 51’’	31’ 40’’
128	8’ 56’’	31’ 22’’
256	9’ 20’’	33’ 12’’

Tabla 14. Tiempos de ejecución de CNN desde cero según el tamaño del lote (epoch=5)

	$IMAG_{1000}$	$IMAG_{3481}$
5	9’ 13’’	32’ 57’’
10	18’ 31’’	1h 5’ 32’’
15	27’ 46’’	1h 35’ 11’’
20	34’ 44’’	2h 3’ 59’’
25	43’ 43’’	2h 43’ 37’’

Tabla 15. Tiempos de ejecución de CNN desde cero según el número de iteraciones (lote=32)

Teniendo en cuenta que el proceso es similar al del caso de la transferencia de aprendizaje, pero creando las capas en lugar de extrayéndolas de una red existente, parece lógico que de la observación de estos datos podamos sacar prácticamente las mismas conclusiones:

- A mayor número de imágenes disponibles, mejor resultado.
- Mayor número de iteraciones o tamaño del lote de imágenes para optimizar no implica mejores resultados. Existe un punto de equilibrio a partir del cual la precisión empieza a disminuir.
- El tiempo de ejecución crece directamente proporcional al número de iteraciones empleadas, pero tiene un comportamiento irregular con la variación del tamaño del lote de imágenes.

7. Conclusiones

7.1. Comparación de resultados

A continuación, se recopilan las principales conclusiones de este trabajo. Se combinan tanto las extraídas en los anteriores puntos como las que son resultado de la comparativa entre los diferentes modelos. Para facilitar su comprensión la siguiente tabla agrupa los valores más significativos de las ejecuciones llevadas a cabo en cada proceso. Es importante resaltar que de los procesos que requerían parametrización se ha seleccionado el mejor rendimiento obtenido en cuanto a precisión. Si bien, no tienen por qué ser los más rápidos, por lo que dependiendo de qué se priorice en la puesta en producción se podría tener que cambiar este criterio.

- Transferencia de aprendizaje a partir de AlexNet: el mejor resultado alcanzado se da cuando el número de iteraciones es 10 y el tamaño de lote de imágenes para la optimización de la red es 64.
- Red neuronal entrenada desde cero: el punto óptimo de rendimiento se cumple cuando el número de iteraciones es 10 y el tamaño de lote de imágenes para la optimización de la red es 32.

	PRECISIÓN		TIEMPO	
	$IMAG_{1000}$	$IMAG_{3481}$	$IMAG_{1000}$	$IMAG_{3481}$
SVM_{COLOR_THRES}	0.8513	0.8532	43''	3' 46''
$SVM_{LUMINANCIA}$	0.9013	0.9137	42''	2' 55''
$CLASIF_{MATLAB}$	0.8464	0.8060	30' 25''	31h 16''
$ALEXNET_{TRANSFER}$	0.9873	0.9920	1h 28' 55''	5h 9' 51''
$ALEXNET_{FEATURES}$	0.9596	0.9752	4' 2''	14' 15''
CNN propia	0.9840	0.9923	18' 31''	1h 5' 32''

Tabla 16. Comparativa de resultados de los diferentes modelos

7.2. Conclusiones

- La primera observación a señalar es la confirmación de la viabilidad del empleo de las redes neuronales convolucionales para crear un sistema de clasificación de imágenes. Su uso permite crear un modelo capaz de "aprender" las características de las diferentes categorías, sin necesidad de tener que articular la extracción de características. Es suficiente con disponer de un conjunto de datos y una arquitectura de red apropiada para el caso.

- Otro punto importante que se ha podido confirmar con las ejecuciones realizadas es la importancia que tiene el hecho de disponer de un amplio conjunto de imágenes. En todos los procesos se puede observar que, a más información disponible en la entrada, mayor conocimiento adquiere el modelo y mejores resultados se obtienen. La excepción la presenta el clasificador de Matlab para el que, debido a que se produce sobre-entrenamiento, utilizar más imágenes supone empeorar el rendimiento.

Relacionado con esta mayor disponibilidad de imágenes, también hemos de tener en cuenta que conlleva un mayor tiempo de entrenamiento.

- En cuanto a la comparación entre sí de los diferentes procesos, en términos generales, se observa con facilidad cómo los modelos que emplean Deep Learning mejoran los resultados respecto a los métodos que siguen el flujo habitual del aprendizaje automático. Según la comparación realizada, ese incremento del ratio varía, pero puede llegar a ser de más del 10% según qué modelos compares.

- Por otra parte, es necesario dejar constancia que, aunque sus ratios de precisión son mejores, las máquinas vector soporte que utilizan, de una u otra forma, la segmentación del color como característica, presentan unos tiempos de ejecución óptimos que tendrán que ser tenidos en cuenta en ciertas aplicaciones si no se consiguen optimizar los tiempos de los modelos basados en aprendizaje profundo.

- Si entramos en detalle y solo nos fijamos en los métodos que utilizan redes neuronales, podemos concluir que los resultados son ligeramente superiores en los dos modelos que las utilizan como clasificador. Sin embargo, para seleccionar uno u otro modelo sería necesario conocer las circunstancias concretas del entorno donde se van a utilizar. Esto es debido a que, aunque tengan mejor rendimiento, su tiempo de ejecución es más elevado y si su aplicación es o no posible dependerá de las necesidades del caso. En relación con el factor del tiempo de ejecución, los resultados avalan que la estrategia seguida en la construcción de la red neuronal ad hoc con la idea de reducir el tamaño de los datos y, con ellos, el tiempo de ejecución del proceso, funciona.

- Por otra parte, dentro del empleo del color como característica para entrenar una máquina vector soporte, destaca el comportamiento positivo de la segmentación por luminancia ante el uso del método que utiliza umbrales de color.

7.3. Líneas de trabajo futuras

Por último, para cerrar la memoria, en este apartado se presenta la lista que reúne algunos de los posibles objetivos que podrían darle continuidad a este proyecto. Sin embargo, antes de pasar a ella, me gustaría llevar a cabo una reflexión sobre cómo se ha desarrollado el proyecto. En primer lugar, nos encontramos satisfechos con el trabajo realizado porque se han cumplido con los tres principales objetivos marcados: profundizar en Deep Learning y conocer sus capacidades, ser capaces de desarrollar modelos empleando sus principios y obtener métodos basados en Deep Learning que obtengan buenos resultados y nos permitan identificar inequívocamente a los individuos del caso a partir del cual surge este proyecto. A pesar de ello, hay dos impedimentos que en momentos determinados del trabajo han complicado llevar éste a buen puerto. El primero de ellos, la falta de documentación práctica que existe respecto a Deep Learning. En el ámbito

teórico podemos encontrar infinidad de artículos y trabajos que hablan de él, pero la documentación de su puesta en práctica es mucho más escasa. Este obstáculo se pudo superar gracias a la selección de MATLAB como herramienta y a la amplia lista de manuales que aporta. El segundo fue que los tiempos de ejecución han estado muy por encima de lo estimado, por lo que en determinados momentos ha comprometido las fechas de entrega.

Ahora sí, se lista a continuación los posibles próximos pasos en relación con este trabajo:

- Experimentar sobre la red neuronal creada a propósito mediante la variación de parámetros diferentes a los que se han tratado. Por ejemplo, la tasa inicial de entrenamiento o el tamaño de las imágenes de entrada necesario para que el rendimiento no decaiga.

- Optimizar el modelo en el que se crea una red neuronal ad hoc para el caso que nos ocupa con el fin de reducir los tiempos.

- Desarrollar modelos que sean capaces de discernir entre imágenes en las que aparece un solo individuo y aquellas en las que aparecen varios peces, ya sea superpuestos o no.

- Mejorar el proceso de captación de las imágenes para evitar las imágenes difuminadas y disponer así de información de mayor calidad.

- Integrar el clasificador seleccionado en el proceso de captación y segmentación de las imágenes en tiempo real.

8. Glosario

TFM: Trabajo Final de Máster

SVM: Support Vector Machine (Máquina de Vector Soporte)

CNN: Convolutional Neural Network (Red Neuronal Convolucional)

GPU: Graphics Processing Unit (Unidad de Procesamiento Gráfico)

ILSVRC: ImageNet Large Scale Visual Recognition Challenge

ReLU: Capa de Unidad Lineal Rectificada

FCL: Fully connected layer (Capa completamente conectada)

RGB: Composición de color Red-Green-Blue

ECOC: Error-Correcting Output Codes

9. Bibliografía

[1] Biro, P.A., Sampson, P., 2015. Fishing directly selects on growth rate via behavior implications of growth-selection that is independent of size. Proceedings of the Royal Society B 282 (1802), 13–15.

[2] Ciuti, S., Muhly, T.B., Paton, D.G., McDevitt, A.D., Musiani, M., Boyce, M.S., 2012. Human selection of elk behavioural traits in a landscape of fear. Proc. Biol. Sci. 279, 4407–4416.

[3] Diaz Pauli, B., Wiech, M., Heino, M., Utne-Palm, A.C., 2015. Opposite selection on behavioural types by active and passive fishing gears in a simulated guppy Poecilia reticulata fishery. J. Fish. Biol. 86 (3), 1030–1045.

[4] P. Marti-Puig, M. Serra-Serra, A. Campos-Candela, R. Reig-Bolano, A. Manjabacas, M. Palmer. Quantitatively scoring behavior from vídeo-recorded, long-lasting fish trajectories. Environmental Modelling and Software. Elsevier. 2018.

[5] Machine Learning, Mathworks, visitada el 26/02/2018. https://es.mathworks.com/discovery/machine-learning.html

[6] Aman Naimat. The New Artificial Intelligence Market: A Data-Driven Analysis of Industries and Companies Adopting AI. O'Reilly Media. 2016. Visitada el 25/03/2018. http://www.oreilly.com/data/free/files/the-new-artificial-intelligence-market.pdf

[7] Gartner says AI technologies will be in Almost Every New Software Product by 2020: visitada el 25/03/2018. https://www.gartner.com/newsroom/id/3763265

[8] César Abascal Gutiérrez. TFM: Deep Learning aplicado al diagnóstico de soldadura mediante espectroscopia óptica de plasmas. Universidad de Cantabria. Octubre 2017.

[9] Bryan García Navarro. TFM: Implementación de Técnicas de Deep Learning. Universidad de la Laguna. Septiembre 2015.

[10] Deep Learning Image Classification, Mathworks: visitada el 28/02/2018 https://es.mathworks.com/help/nnet/deep-learning-image-classification.html

[11] Deep Learning, Mathworks: visitada el 27/02/2018. https://es.mathworks.com/discovery/deep-learning.html

[12] Inteligencia Artificial como servicio: reconocimiento de imágenes. Visitada el 01/03/2018. https://www.paradigmadigital.com/techbiz/inteligencia-artificial-servicio-reconocimiento-imagenes/

[13] Deep Learning for Image Recognition: why it's challenging, where we've been, and what's next. Visitada el 26/03/2018. .https://towardsdatascience.com/deep-learning-for-image-classification-why-its-challenging-where-we-ve-been-and-what-s-next-93b56948fcef

[14] Machine Learning y Deep Learning with Matlab, Mathworks: visitada el 03/03/2018. https://es.mathworks.com/videos/machine-learning-and-deep-learning-with-matlab-1499677542930.html

[15] Rodrigo Colombo. TFM: Deep Learning para el reconocimiento de imágenes en Raspberry Pi 2. Universidad de la Laguna. Marzo 2016.

[16] Carlos Pérez Estruch. TFM: Emotions Recognition using Deep Learning. Universidad Politécnica de Valencia. Septiembre 2016.

[17] Convolutional Neural Network. Visitada el 10/03/2018. https://es.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

[18] E-book Machine Learning con MATLAB. https://es.mathworks.com/campaigns/products/offer/machine-learning-with-matlab.html

[19] Burges, C. A tutorial on support vector machines for pattern recognition. Bell Laboratories, Lucent Techonologies, USA. 1998. Data Mining and Knowledge Discovery, 2, 121-167.

[20] Clase ClassificationECOC. Mathworks. Visitada el 17/04/2018. https://es.mathworks.com/help/stats/classificationecoc-class.html

[21] Clase imageCategoryClassifier. Mathworks. Visitada el 20/04/2018. https://es.mathworks.com/help/vision/ref/imagecategoryclassifier-class.html

[22] AlexNet según Wikipedia. Visitada el 24/05/2018. https://en.wikipedia.org/wiki/AlexNet

[23] Clase alexnet. Mathworks. Visitada el 21/03/2018. https://es.mathworks.com/help/nnet/ref/alexnet.html

[24] Number of Parameters and Tensor Sizes in a Convolutional Neural Network. Learn OpenCV. Visitada el 24/05/2018. https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/

[25] trainingOptions. Mathworks. Visitada el 03/05/2018. https://es.mathworks.com/help/nnet/ref/trainingoptions.html

[26] Deep Learning. neuronalnetworkanddeeplearning.com. Visitada el 03/05/2018. http://neuralnetworksanddeeplearning.com/chap6.html

[27] Feature Extraction using AlexNet. Mathworks. Visitada el 24/03/2018. https://es.mathworks.com/help/nnet/examples/feature-extraction-using-alexnet.html

[28] Create Simple Deep Learning Network for Classification. Mathworks. Visitada el 26/03/2018. https://es.mathworks.com/help/nnet/examples/create-simple-deep-learning-network-for-classification.html

[29] Francisco José Núñez Sánchez-Agustino. TFM: Diseño de un sistema de reconocimiento automático de matrículas de vehículos mediante una red neuronal convolucional mediante una red neuronal convolucional. Universitat Oberta de Catalunya. Junio 2016.