Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Reconocimiento y Clasificación de Eimeria mediante la aplicación de Detección de Elipses y Redes Neuronales

Autor: Pedro Soriano Aguado
Tutor: Susana Hornillo Mellado

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Reconocimiento y Clasificación de Eimeria mediante la aplicación de Detección de Elipses y Redes Neuronales

Autor:
Pedro Soriano Aguado

Tutor:
Susana Hornillo Mellado

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Carrera: Reconocimiento y Clasificación de Eimeria mediante la aplicación de Detección de Elipses y Redes Neuronales

Autor: Pedro Soriano Aguado

Tutor: Susana Hornillo Mellado

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Sin duda esta ha sido una etapa especial. Puedo decir sin que me tiemble el pulso que ha albergado los mejores y los peores momentos de mi vida. No los hubiera podido sobrevivir ni disfrutar plenamente sin la gente que ha participado en todo este período. Y sin ellos, no sería quién soy a día de hoy. Les estoy eternamente agradecido y cualquier texto o discurso que escriba palidece al lado de lo que me han aportado. Muchísimas gracias por ayudarme a ser la mejor versión de mí mismo.

Muchísimas gracias a mis compañeros de carrera, que hicieron de la rutina algo de lo que enamorarse. Muchas gracias Luigy, muchas gracias Mati, muchas gracias Manu, muchas gracias Juan, muchas gracias Esther y muchas gracias Emilio. Gracias a vosotros cosas como levantarme por las mañanas para ir a la ETSI, estar en la planta 2 de la biblioteca estudiando hasta horas intempestivas, programar hasta que se me agarrotaran los dedos en el CDC y comer con prisas en la FCom, eran cosas que hacía con una sonrisa en los labios.

Muchísimas gracias a la gente de mi residencia y de mi piso, que hicieron de la convivencia prácticamente una chirigota diaria. Muchas gracias Julio, muchas gracias Alba, muchas gracias Jesús y muchas gracias Migue. Ir al salón a ver qué tramabais, Julio tardando en cenar más de dos horas y Jesús y su querido despacho, son algunos de los pequeños momentos que me llevo conmigo.

Muchísimas gracias a Susana, ya antes de ser mi tutora pude comprobar que era una profesora que realmente se implica con los alumnos y que exuda dedicación en lo que hace. Y tras haberla tenido como tutora, no ha hecho más que demostrar con creces lo descrito.

Por último, muchísimas gracias a mi familia. Vosotros habéis hecho esto posible, y no me refiero a la financiación. Vosotros me habéis hecho ver que por muy mal que parecieran las cosas, siempre hay algo de lo que estar agradecido. Y no puedo estar más que de acuerdo con vosotros. Muchas gracias Mamá, muchas gracias Papá y muchas gracias Marta. Por mucho que el hecho de haber estudiado fuera me haya tenido lejos de vosotros, siempre habéis estado cerca. Gracias.

Pedro Soriano Aguado

Sevilla, 2020

Resumen

El Eimeria es un parásito animal que provoca enfermedades y muerte animal, disminuyendo la productividad. Existen múltiples especies de Eimeria para cada especie animal, con diferencias mínimas de tamaño y forma entre ellas, que afectan de formas distintas al portador. Por tanto, un reconocimiento rápido y preciso de cada especie es crucial para un diagnóstico eficaz.

Lo que se propone en este Trabajo Fin de Grado es utilizar varias técnicas punteras relacionadas con la visión artificial para automatizar el procesamiento de las imágenes y su posterior clasificación. Para ello se han usado los datos vinculados al estudio de Castañón et al. [2] para el diseño y codificación del programa.

Es importante destacar que el algoritmo en el que se basa proviene del artículo de Ginoris et al. [1] pero aplicando herramientas más modernas, como el reconocimiento de elipses y el uso de una red neuronal que se basa en la extracción de características para su clasificación.

Abstract

Eimeria is an animal parasite that causes severe diseases and animal death, consequently producing a decrease in productivity. There are multiple Eimeria species for each animal species, with minimal differences in terms of size and shape, which affect the carrier in different ways. Therefore, a quick and precise recognition of each species is crucial in order to elaborate an effective diagnostic.

In this paper, it is proposed the usage of several state-of-the-art techniques related to computer vision to automatize the image processing and its subsequent classification. The dataset used, present in the Castañon et al. [2] project, was the groundwork to the design and coding of the model.

The algorithm which inspired this model comes from the article provided by Ginoris et al. [1], but several more contemporary tools have been applied, like ellipse detection and a neural network based on feature extraction in the interest of classifiying the data.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

How odd I can have all this inside me and to you it's just words.

- David Foster Wallace, The Pale King-

El Eimeria es un parásito animal que puede causar enfermedades altamente peligrosas y muerte animal, con lo que provoca una reducción de la productividad. Un diagnóstico temprano se obtiene mediante la examinación de las imágenes microscópicas de las heces. Los ooquistes de Eimeria varían en formas, tamaños y texturas, y pueden ser detectados midiendo las diferencias de los rasgos descritos. Hasta tiempos recientes, este proceso siempre se ha realizado a mano, mediante profesionales del sector realizando el contado ellos mismos. Como es de suponer, esta vía requiere de mucho tiempo y capital humano.

En este trabajo se ha planteado un enfoque al reconocimiento de estos parásitos partiendo del algoritmo desarrollado por Ginoris et al. [1] pero con ciertos añadidos y modificaciones, como la implementación del reconocimiento de elipses y una clasificación realizada mediante una red neuronal.

1.1. Estado del Arte

A continuación, se revisarán distintos estudios relacionados con el tema o algunos de los apartados del mismo, ya que en este trabajo se incluyen distintos procedimientos y ejemplos de documentación que englobe a todos ellos es realmente escasa. Por tanto, aunque traten aspectos de distinta índole, constituyen piezas complementarias al trabajo que se ha realizado. Todos estos se van a mostrar en orden cronológico.

Castañón et al. [2] para el reconocimiento de la Eimeria se basan en extraer las características morfológicas (se basa en la curvatura, la geometría y la textura de las imágenes). Todos ellos constituyen un total de 13 parámetros utilizados para clasificar las imágenes. Clasifican 7 tipos de Eimeria presentes en aves. Utilizan un clasificador Bayesiano e implementan un algoritmo para optimizar la combinación de características usadas. El porcentaje de acierto fue del 85.75%.

Ginoris et al. [1] describe un procedimiento para la identificación de seres microscópicos nacidos en el agua. Describe un procedimiento muy detallado para el preprocesamiento con énfasis en la segmentación. Hace uso de un análisis de discriminantes y posteriormente usa una red neuronal de dos capas para la clasificación. Dependiendo del tipo de organismos (diferencia entre los que tienen tallo y los que no) su porcentaje de acierto global varía entre 51.4% a 85.6%.

Suzuki et al. [3] su objetivo es la detección de 15 tipos de parásitos intestinales. Aplican una técnica de preprocesamiento llamada TF-Test cuyo fin es limpiar el ruido de la imagen. Hace uso de un OPF (Optimum Path Forest) porque les resulta más eficiente que una red neuronal artficial o un SVM (Support Vector

Machine). Es capaz de alcanzar una sensibilidad del 96.3% al parásito y una capacidad del 99.17% para clasificarlos.

Ghazali et al. [4] trata de reconocer parásitos intestinales relacionados con la malaria. Plantea tres vías distintas de preprocesamiento, luego hace uso de la extracción de rasgos y por último clasifica mediante un F-DTS (Filtration and Steady Determinations Thresholds System). Obtienen unos porcentajes de acierto del 93% y el 94% para los dos tipos principales.

Savkare et al. [5] ofrece una alternativa para la identificación de los parásitos causantes de la malaria mediante el análisis de los glóbulos rojos infectados. Emplea la segmentación, la extracción de rasgos y finalmente ejecuta la clasificación mediante un SVM (Support Vector Machine). Consigue un ratio de acierto del 80%.

El investigador Mohamed A. E. Abdalla ha sido bastante prolífico en los temas relacionados con este trabajo, con lo que es relevante repasar sus trabajos de manera continuada para ver qué modificaciones y mejoras introduce en entregas posteriores.

Abdalla et al. [6] versa sobre la identificación de los distintos tipos de Eimeria (como también ocurrirá en trabajos posteriores) cuyas imágenes fueron obtenidas de la base de datos ofrecida por Castañón et al. [1]. Destaca el uso de técnicas de análisis basadas en píxeles como una mejora con respecto el análisis basado en la morfología. Emplea para la segmentación un algoritmo basado en la vecindad de Moore. El algoritmo KNN (K-Nearest Neighbours) es el elegido para llevar a cabo la clasificación. Llegan a alcanzar un 82% de aciertos.

Abdalla et al. [7] introduce como novedad el uso del "Rectangle Frame Feature Set" para optimizar la eficacia de las muestras. Aquí llega a conseguir un porcentaje de acierto del 82.83% haciendo uso de menos características que en estudios anteriores.

Finalmente, en Abdalla et al. [8] aplica el método de Relieff para evitar la sobrecarga de parámetros, y para la clasificación emplea K-Nearest y una ANN (Artficial Neural Network) de manera paralela, consiguiendo ésta última el mejor porcentaje de acierto, siendo del 96.6% en el caso de los Eimeria en aves.

Li [9] realiza un extenso repaso a la historia del uso de técnicas de reconocimiento aplicadas a seres microscópicos de distintos campos. Concluye que los procedimientos más usados (en sus respectivas fases) son la segmentación de imágenes, las características de forma (Shape characteristics), la selección y fusión de rasgos, los valores de decisión, SVM (Support Vector Machines) y ANN (Artificial Neural Networks) y por último, un sistema de evaluación basado en la precisión.

Li (Xiang) [10] emplea imágenes hiperespectrales, que se trata de un tipo de imágenes que contienen información de todo el espectro electromagnético y que tienen un gran potencial a la hora de clasificar células. Los dos pilares fundamentales son (1) Procesamiento de imágenes hiperespectrales y (2) Redes neuronales convolucionales para la clasificación de las células. Hacen uso de técnicas de "Data Augmentation" para aumentar la base de datos de entrenamiento aplicando alteraciones en las imágenes originales [11]. Su precisión gira en torno al 90%, variando según el tipo de datos usados.

Ferdous [12] se centran en clasificar bacterias mediante una DCNN (Deep Convolutional Neural Network) que ha que ha sido entrenanda aplicando técnicas de "Transfer Learning". Aplican preprocesamiento, pero dejan que la red se encargue de la extracción de características. El nivel de acierto de la predicción ronda el 95%.

Monge [13] parte de las imágenes de las Eimeria para ave que ya aparecieron en Castañón et al. [1] y los distintos trabajos de Mohamed A.E. Abdalla. Recurren también a la técnica de "Data Augmentation" aunque aplicando sólo un número limitado de transformaciones, como giros e inversiones horizontales y verticales. Emplea una CNN (Convolutional Neural Network) para la clasificación y con ésta llega a obtener un 90.42% de precisión.

Con el transcurso de los años, es fácilmente apreciable que la tendencia general parece inclinarse por las redes neuronales cada vez más gracias al alto grado de precisión que son capaces de alcanzar.

2 EIMERIA

We're all one thing, like cells in a body. 'Cept we can't see the body. The way fish can't see the ocean. And so we envy each other. Hate each other. How silly is that? A heart cell hating a lung cell.

- Charlie Kaufman, Adaptation -

El Eimeria se trata de un género del parásito coccidio, y normalmente habita en el intestino de su huésped. Son específicos para cada huésped y se caracterizan por su ciclo de vida que se desarrolla parasitando una única especie. Debido a su propagación masiva en poblaciones de distintas especies animales y al hecho de que un huésped pueda estar infectado por múltiples especies de Eimeria, son considerados de vital importancia para la veterinaria. [14] [15]

Muchas especies no son patógenas y pueden no causar enfermedad alguna. Ciertas especies en cambio son altamente patogénicas y causan diarrea, exicosis [17], anorexia y otros síntomas causando un decrecimiento de la producción así como en algunos casos la muerte. Se diagnostica que un animal ha sido infectado por Eimeria mediante la examinación de las heces en busca de los ooquistes. En el caso de las aves, el diagnóstico se suele confirmar mediante disecciones diagnósticas. [14] [16]

La progresión de la coccidiosis es tan rápida que cualquier tipo de tratamiento resulta un tanto futil porque muy probablemente sea ya demasiado tarde para tratarlo. Muy comúnmente se añaden los medicamentos directamente en la comida de manera preventiva. Los brotes se pueden controlar con la mejora de la higiene, reduciendo las multitudes y aislando los infectados. La desinfección es bastante impráctica dado que los ooquistes infecciosos son resistentes a muchos desinfectantes convencionales. [16]

2.1. Morfología

Los parásitos coccidianos pasan por tres etapas de desarrollo: esquizontes, gamontes y ooquistes. Los esquizontes empiezan como pequeñas y redondas células basofílicas [18] localizadas dentro de las células huésped. Los esquizontes maduros se presentan como racimos confinados por la membrana de múltiples cuerpos pequeños basofílicos. Los gamontes presentan ya diferenciación sexual, siendo los microgamontes masculinos y los macrogamontes femeninos. [14] [16]

El ooquiste constituye la etapa clave para el diagnóstico del Eimeria. Esta etapa se puede encontrar en las heces de los huéspedes y puede ser usada fácilmente para un diagnóstico in vivo. Poseen una pared robusta de dos o tres capas dependiendo de la especie. Una vez finalizado su ciclo útil, éstos son capaces de expulsar hasta ocho esporozoitos [19] siendo cada uno de ellos capaz de infectar una célula del huésped. [14] [16]

2.2. Ciclo de vida

El ciclo vital del Eimeria se da tanto en el entorno como en un huésped. Mientras que están en el exterior y durante la ingestión, estos se encuentran en forma de ooquiste. Cuando el huésped expulsa al parásito para la posterior infección de otro ser, éste se encuentra en una fase temprana de ooquiste sin esporular. El ooquiste transiciona a su fase tardía cuando forma esporas mediante reproducción asexual. La formación de esporas requiere de un medio aeróbico y tarda en torno a un día. [21]

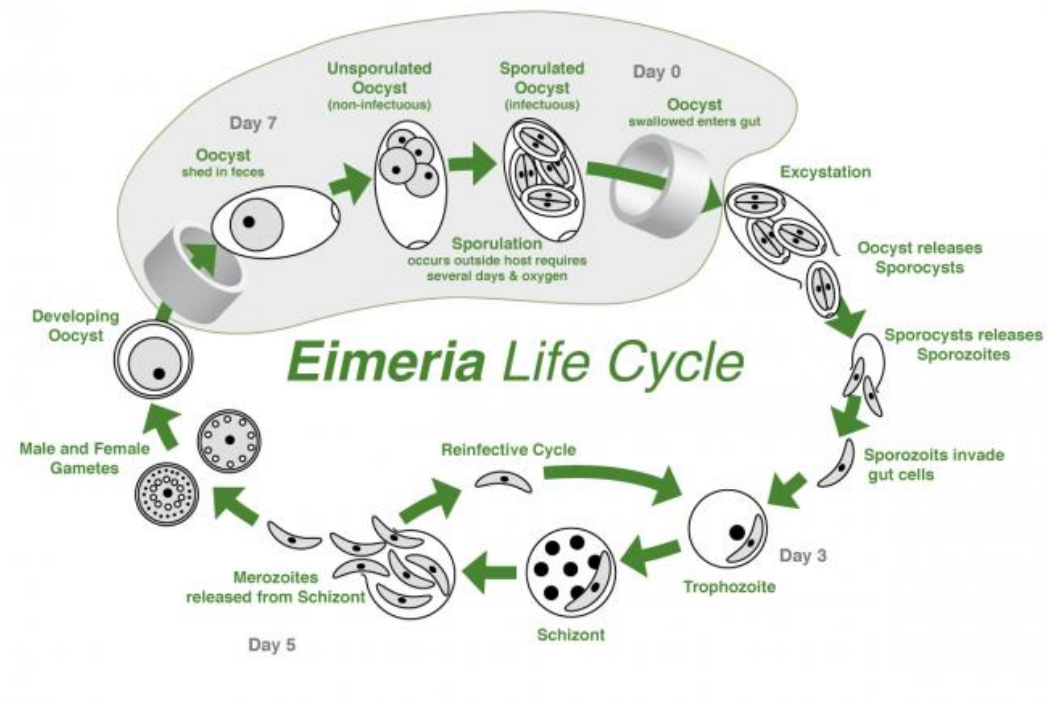

Figura 2-1. Ciclo vital del Eimeria (imagen extraida de [22])

Los ooquistes son entonces consumidos por un animal y alcanzan el intestino, donde se descomponen y liberan los esporozoitos [19]. Los recién formados esporozoitos circulan el intestino donde invaden las células epiteliales de las paredes. Luego se convierten en tropozoitos y se unen para formar un esquizonte. Los esquizontes entonces expulsan merozoitos, que abandonan la célula para infectar otras células epiteliales.[21]

Todo este ciclo continúa varias generaciones, hasta que los merozoitos se transforman en macho o hembra y llevan a cabo la reproducción sexual. El resultado es un ooquiste sin esporular, que es liberado al exterior por el huésped a través de las heces, empezando el ciclo de nuevo. [21]

2.3. Tipos de Eimeria

La parte experimental de este trabajo únicamente se centrará en las especies de Eimeria presentes en las aves, de modo que, aunque están presentes en muchos mamíferos, únicamente se describirán aquellas relacionadas con las gallinas.

En aves, hay siete especies reconocidas de Eimeria:

- *E. máxima* (a)
- *E. brunetti* (b)
- *E. tenella* (c)
- *E. necatrix* (d)
- *E. praecox* (e)
- *E. acervulina* (f)

- *E. mitis* (g)

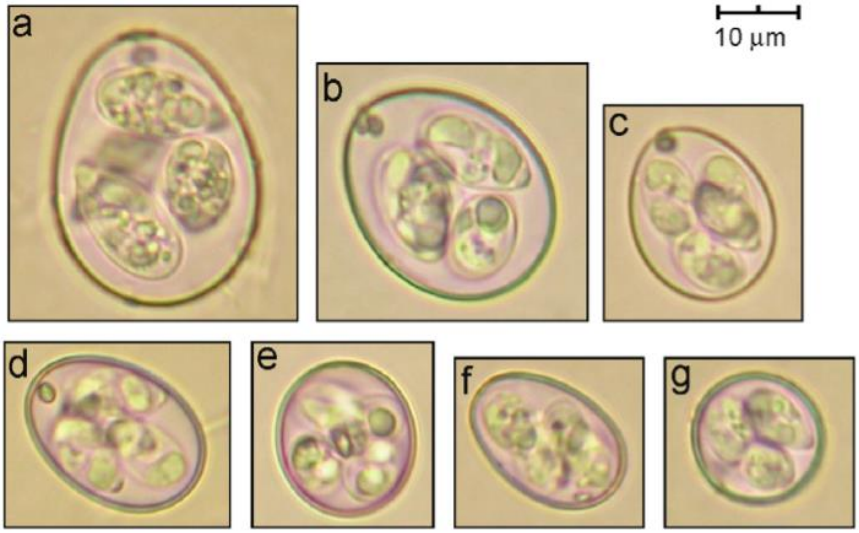

Figura 2-2. Se muestran los distintos tipos siguiendo la asignación indicada arriba (imagen extraida de Castañón et al. [2])

En general, poblaciones mixtas de Eimeria están presentes en muchas granjas comerciales. La prevalencia de las distintas especies de Eimeria depende del área geográfica en donde se encuéntrela granja. Por ejemplo, en Europa la *E.acervulina* es la más extendida, independientemente del tamaño de la granja. Si bien no se ha encontrado una relación directa entre el tamaño de la granja y la presencia de Eimeria, infecciones de especies mixtas son mucho más comunes en granjas pequeñas.

2.4. Base de Datos Usada

Para este trabajo se ha hecho uso de la base de datos proporcionada por Castañón et al. [2] accedida desde [20].

La base de datos tiene un tamaño total de 4467 imágenes de ooquistes de las distintas siete especies que parasitan a las aves, y además en algunos casos existen diversas cepas (Figura 2-3) de la misma especie. Es importante destacar que la cantidad de imágenes de cada especie no es equitativa, de modo que en el momento de la realización del clasificador se tomarán medidas para corregir esa desproporcionalidad. Todas estas imágenes contienen un ooquiste por imagen y se han obtenido mediante el recorte de las imágenes originales que contenían múltiples células de este tipo, no han sufrido ningún tipo preprocesamiento y se ha mantenido su tamaño original [2].

Tabla 2-1. Base de datos disponible

Tipos de Eimeria	Imágenes totales	Imágenes cepa 1	Imágenes cepa 2	Imágenes cepa 3
E. acervulina	744	144	432	168
E. brunetti	442	442	-	-
E. maxima	360	128	104	128
E. mitis	825	208	230	387
E. necatrix	502	173	329	-
E. praecox	898	222	219	457
E. tenella	696	150	353	193

Figura 2-3. Imágenes de *Acervulina*, cepas ACE103, ACEH, y ACER7, respectivamente.

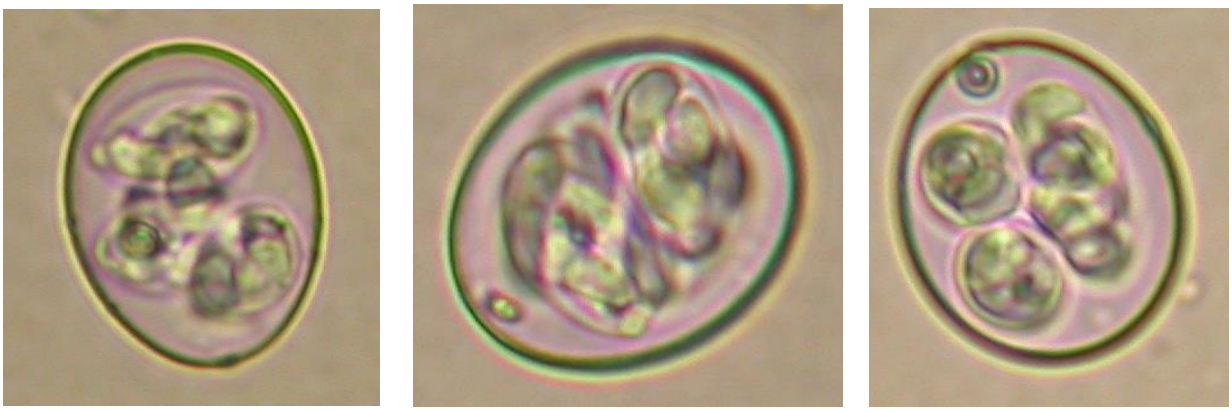

Figura 2-4. Imágenes de *Brunetti*, todas de la cepa BRUC.

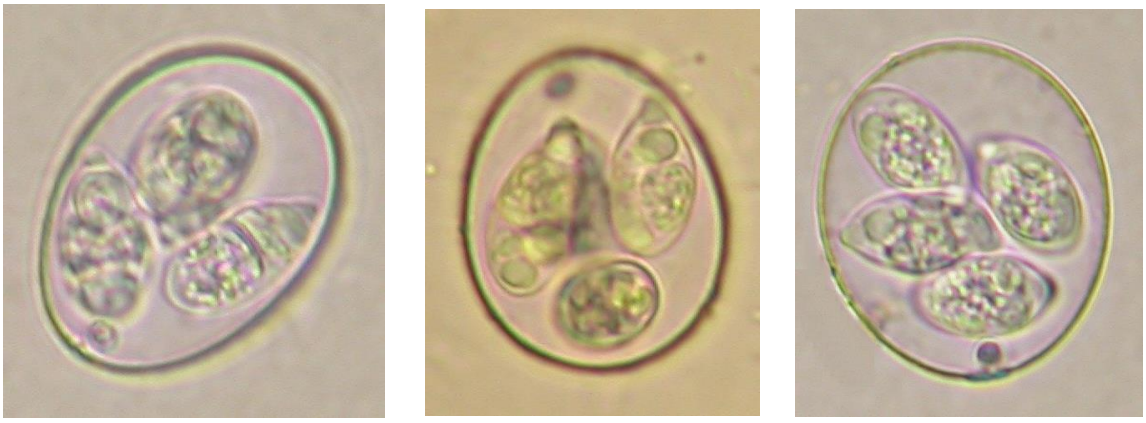

Figura 2-5. Imágenes de *Maxima*, cepas MAX50, MAX103, y MAXL, respectivamente.

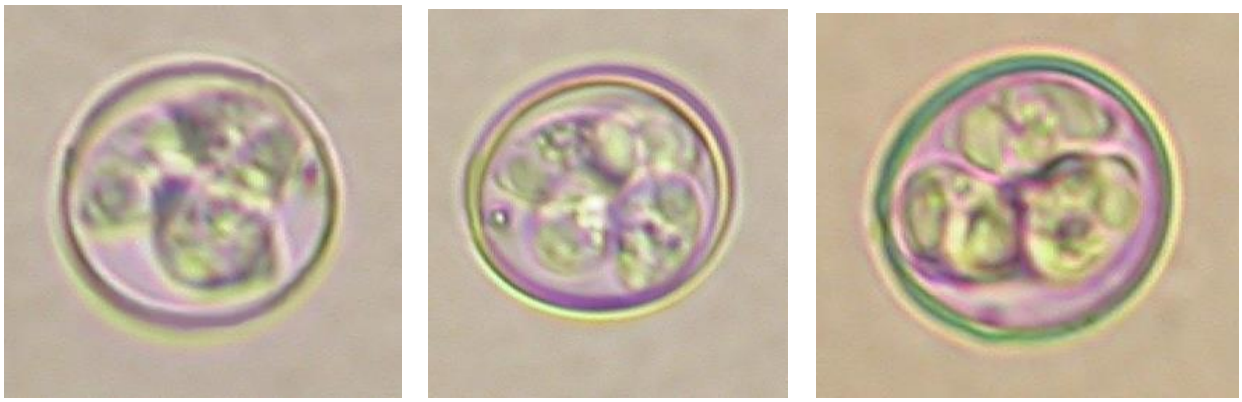

Figura 2-6. Imágenes de *Mitis*, cepas MIT30, MIT44, y MITRT, respectivamente.

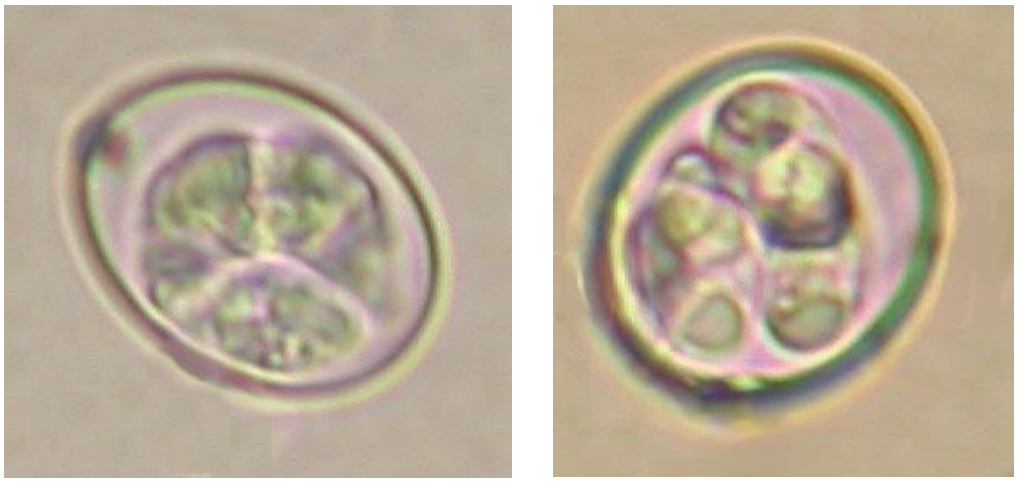

Figura 2-7. Imágenes de *Necatrix*, cepas NEC103, y NECDF, respectivamente.

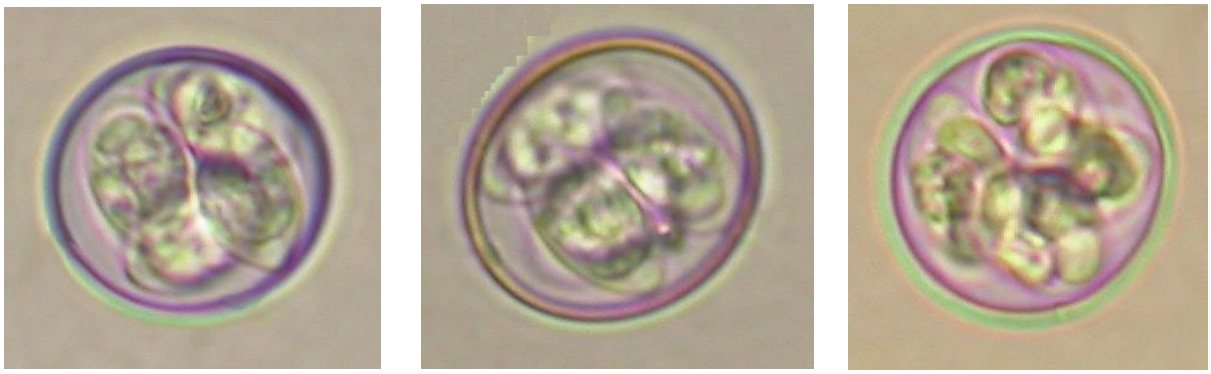

Figura 2-8. Imágenes de *Praecox*, cepas PRA1D1A, PRAD, y PRAH, respectivamente.

Figura 2-9. Imágenes de *Tenella*, cepas TENCR, TENH, y TENMC, respectivamente.

Nota: No todas las imágenes empleadas tienen el mismo tamaño, aquí han sido reescaladas para poder comparar los rasgos y formas de manera más sencilla.

3 DEEP LEARNING - SUMARIO

If AI has a goal and humanity just happens to be in the way, it will destroy humanity as a matter of course without even thinking about it…It's just like, if we're building a road and an anthill just happens to be in the way, we don't hate ants, we're just building a road.

- Elon Musk -

Las redes neuronales han sido una de las herramientas científicas que más drásticamente han evolucionado hasta copar gran parte de las actividades relegadas a inteligencias artificiales. Tal y como se puede ver en la Figura 3-1 [23], el uso de este tipo de modelos aumenta exponencialmente con el paso de los años.

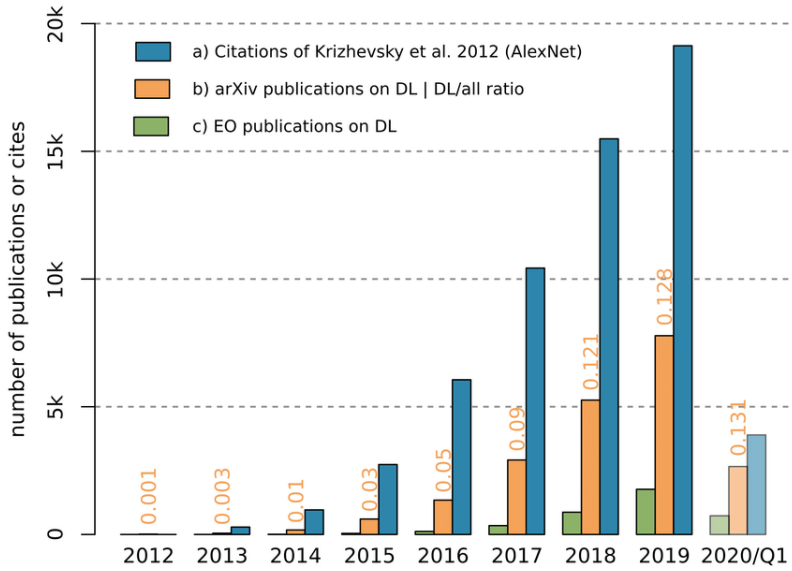

Figura 3-1. Evolución de la publicación de artículos relacionados con el *Deep Learning* (extraída de [24]).

Antes de nada, es indispensable diferenciar entre inteligencia artificial (IA), *machine learning* (ML), y *deep learning* (DL) que son conceptos estrechamente relacionados y que hoy en día se usan de manera conjunta en artículos relacionados con las redes neuronales, con lo que una comprensión clara de cada uno de ellos [26] es requerida para proseguir con este trabajo.

- **Inteligencia artificial:** Se trata de una ciencia como podría ser la física o las matemáticas. Investiga las posibles vías de diseño que permitan crear programas y máquinas capaces de solucionar problemas, cualidad que hasta el momento sólo se le atribuía a los humanos. Como indica la imagen adjuntada a continuación [27], engloba tanto al ML como al DL.
- ***Machine Learning*:** Incluido dentro de la IA, permite a los sistemas tener la capacidad de aprender y

perfeccionarse continuamente mediante la recopilación de experiencia durante el desarrollo de su tarea. Estos incluyen herramientas como algoritmos u otras técnicas de automatización para poder llevar a cabo su labor.

- ***Deep Learning*:** A su vez incluido dentro del ML, utiliza estructuras complejas que tratan de replicar el funcionamiento del sistema neuronal humano para poder llevar a cabo tareas complejas [28] como la visión artificial o el reconocimiento de voz.

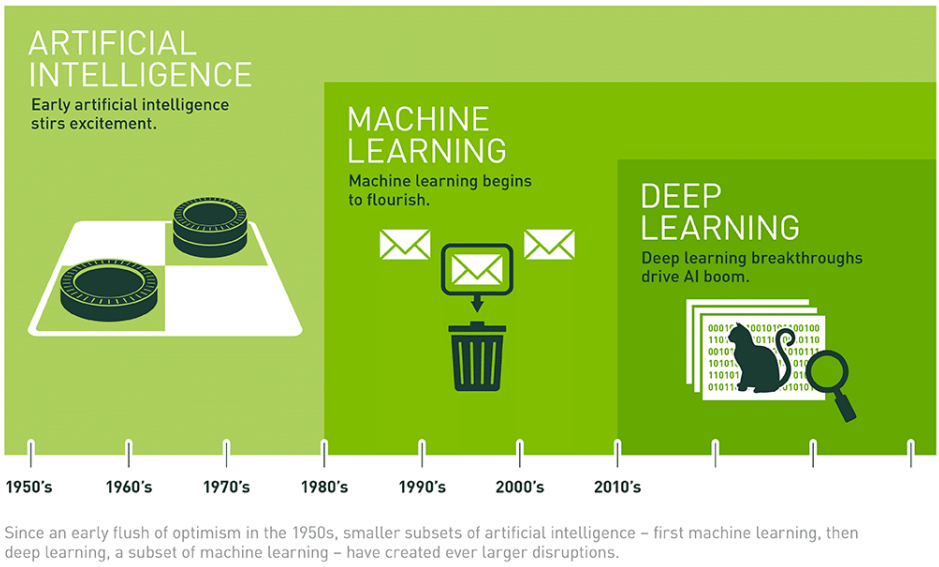

Figura 3-2. Progreso de la IA en los últimos 60 años (extraída de [27]).

3.1 Machine Learning

El *Machine Learning* (ML) es un campo de la informática que estudia algoritmos y técnicas para obtener soluciones de manera automática a problemas complejas que son difíciles de programar haciendo uso de una metodología convencional. Convencionalmente, se suele dar primero una especificación al programa, que conlleva diseñar un número de reglas necesarias para cubrir las necesidades de nuestro problema y posteriormente implementar todo esto mediante un lenguaje informático. [29]

Principalmente, aprenden gracias a los datos que se le pasan, de modo que a una mayor cantidad de datos, más precisos se vuelven. La meta que se propone un algoritmo de ML es tratar de crear un modelo, a partir de una base de datos previamente etiquetada, que sea capaz de predecir elementos de fuera de la base de datos mediante el uso de ese modelo. Este enfoque se titula *Supervised Machine Learning*. [29]

Los algoritmos de ML tienen la propiedad de ser más precisos que los de los humanos debido a que consideran todas las posibles variables sin ningún tipo de parcialidad ni prejuicio hacia los parámetros. Pese a que se sabe que funcionan, muchas veces no se tiene claro qué es lo que ocurre internamente, con lo que existe también investigadores que se concentran en evaluar la interpretabilidad de los modelos.[29]

3.1.1 Tipos de *Machine Learning*

Como ya se mencionó en el apartado anterior, la precisión de los algoritmos de ML depende de la cantidad de datos disponible. Los principales modelos de aprendizaje son:

- ***Supervised Learning* (SL)**
- ***Unsupervised Learning* (UL)**

- ***Semi-supervised Learning*** **(SSL)**
- ***Reinforcement Learning*** **(RL)**

El principal diferenciador entre ellos es la existencia de datos etiquetados (*labelled data*) o sin etiquetar. Si sabemos la respuesta correcta a la hora de responder de qué tipo son, entonces se consideran datos etiquetados. Por ejemplo, si tenemos una imagen de una flor y sabemos a qué especie pertenece, entonces es una imagen etiquetada. En cambio, cuando no sabemos reconocer a qué tipo pertenece la información, se consideran datos sin etiquetar (*unlabelled data*). [29]

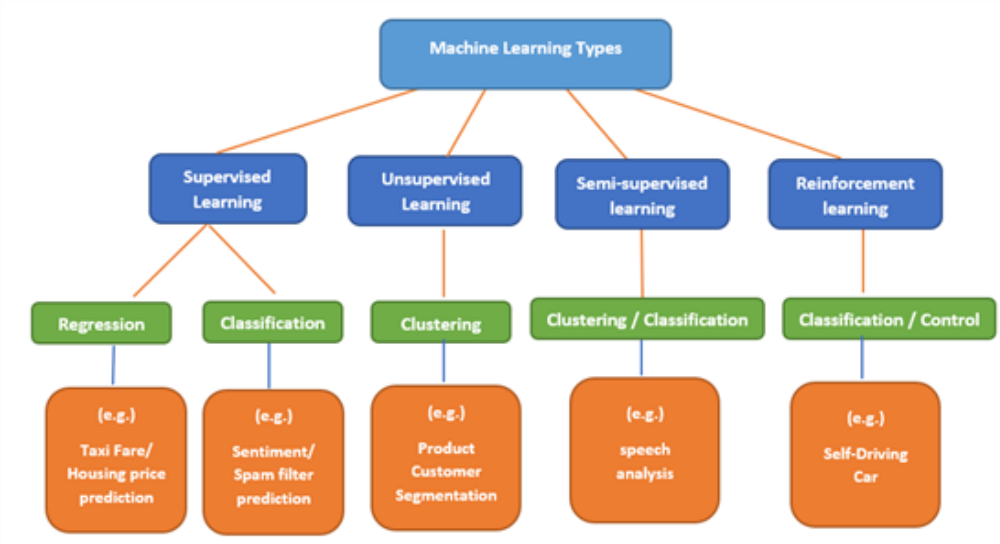

Figura 3-3. Clasificación de los Tipos de Machine Learning (extraída de [32]).

Supervised Learning

El *supervised learning* requiere de la entrada de una base de datos, y junto a ésta, las respuestas de la pregunta que se le plantea al algoritmo que sea capaz de resolver. El algoritmo entonces trata de aprender a reconocer qué papel juegan los distintos valores de los parámetros suministrados para determinar la respuesta. De esta forma, cuando se le pase a la máquina algún caso nuevo, sea capaz de predecir correctamente la respuesta. [29]

Un ejemplo muy típico consiste en pasar miles de imágenes y adjuntar a cada una de ellas la respuesta de a qué tipo pertenecen. Una vez concluida esta fase de "entrenamiento", la máquina debería ser capaz de diferenciar entre clases, como por ejemplo, diferenciar una rosa de un tulipán.[29][31]

Gran parte del uso del ML que tiene lugar en el mundo está relacionado con el SL, y estos son los dos tipos de problemas más comunes: [33]

- ***Classification problems***: Se basan en la capacidad de ser capaces de distinguir entre distintos tipos o categorías.
- ***Regression problems***: Se centran en tratar de prever la evolución de una variable continua, como pudiera ser las acciones de una compañía en bolsa.

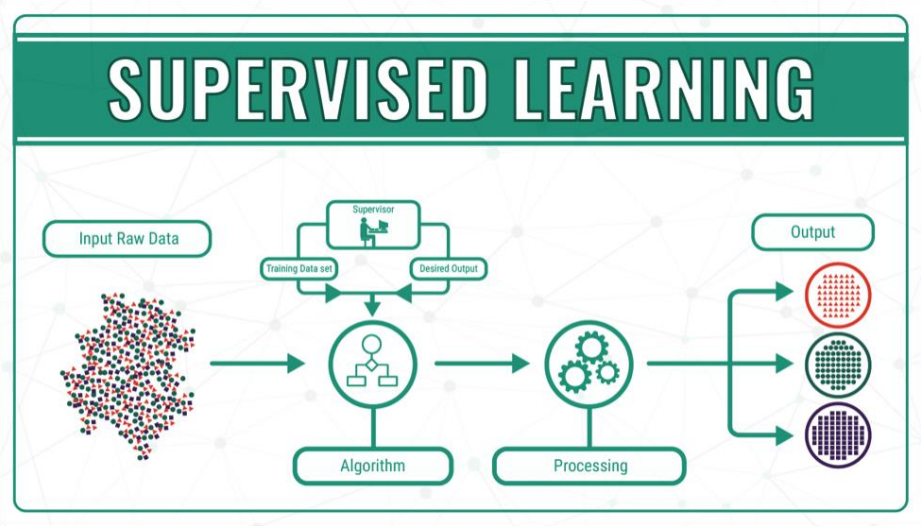

Figura 3-4. Esquema simplficado del SL (extraída de [33]).

Unsupervised Learning

Para este tipo de ML, al programa se le proporciona únicamente una base de datos, pero sin ningún tipo de respuesta correcta con lo que el posible resultado es desconocido para el propio programador. La máquina por lo tanto analiza todos los datos que posee y trata de buscar parecidos con el uso de todas las operaciones lógicas que es capaz de aplicar. Finalmente será capaz de reconocer y agrupar los datos según su similitud. [29][33][34]

Un paralelismo muy claro es el uso de tallajes en ropa o zapatillas. Los fabricantes no pueden predecir las medidas exactas de los clientes que van a hacer algún tipo de compra, con lo que estudian el abanico de medidas de los posibles consumidores y los agrupan eficientemente entre un número limitado de tallas que tratan de satisfacer al mayor número de personas posible. [29]

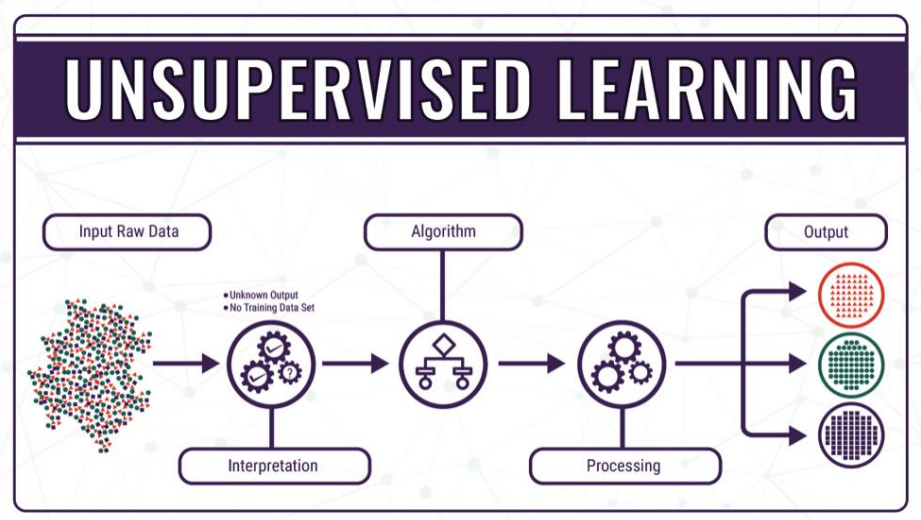

Figura 3-5. Esquema simplficado del UL (extraída de [34]).

Semi-supervised Learning

Como su propio nombre indica, se encuentra a medio camino entre el SL y el UL, puesto que recibe tanto datos etiquetados como sin etiquetar para ser usados durante su entrenamiento. El escenario típico suele ser que se le proporcionan unos cuantos datos etiquetados y otros sin etiquetar, con lo que el algoritmo irá formando grupos o *clusters* [36] e irá usando los datos ya reconocidos para proporcionar etiquetas a aquellos datos que se encuentren en dichas agrupaciones. [29] [35]

Su principal uso deviene de querer aprovechar toda la información disponible en nuestro haber, independientemente de que no esté toda etiquetada. [35]

Reinforcement Learning

El *reinforcement learning* consiste en el entrenamiento de modelos de ML que sean capaces de realizar tomas de decisiones de manera secuencial. Consta de cuatro elementos esenciales: [37]

- Agente: El programa o modelo que es entrenado, con el objetivo de cumplir con la tarea que especifique.
- Ambiente: El entorno, real o virtual, en el que el agente lleva a cabo las acciones.
- Acción: Aquellos actos realizados por el agente, que provocan un cambio de estado en el entorno.
- Recompensas: Los resultados de la evaluación de la acción, que puede desembocar en recompensa o penalización. [38]

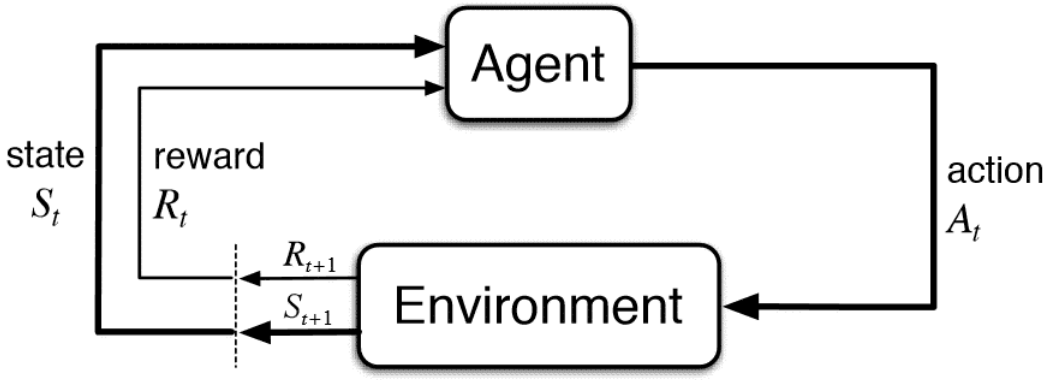

Figura 3-6. Esquema simplficado del RL (extraída de [39]).

El modelo se enfrenta a una situación muy similar a un videojuego, ya que emplea técnicas de ensayo y error para conseguir la meta propuesta. Para conseguir el objetivo propuesto por el programador, se le recompensa o penaliza a la máquina de modo que eventualmente consiga maximizar las recompensas totales. [38]

De este modo, se considera que el RL es una de las mejores formas para testear la creatividad de la máquina, ya que el programador únicamente diseña las recompensas y el objetivo, es la máquina la que tiene averiguar cómo ejecutar la tarea, empezando con tácticas más brutas y refinando durante la ejecución, que gracias a que puede hacer varios intentos paralelos permite que este método sea eficiente. [38]

3.2 *Deep Learning*

El *Deep Learning* (DL) se encarga de procesar modelos computacionales que están compuestos por múltiples capas para ser capaces de entender datos estructuralmente intrincados con varios niveles de abstracción. Han mejorado dramáticamente los resultados obtenidos en campos como el reconocimiento de voz, la detección y el reconocimiento visual de objetos, la reconstrucción de circuitos cerebrales [41], la predicción de los efectos de las mutaciones en los genes [44] y el análisis de los aceleradores de partículas [43], por mencionar unos cuantos. [42]

El DL hace uso del *Representation Learning* (RL), que se trata de un conjunto de métodos que permiten encontrar las representaciones necesarias para la clasificación por medio del aporte de datos. Si bien es cierto que los métodos de DL emplean métodos de RL con múltiples capas de representación obtenidas combinando módulos capaces de transformar la representación de un nivel en una representación con un mayor nivel de abstracción. [42]

Es capaz de realizar estos análisis por el uso de algoritmos de propagación hacia atrás de errores (*backpropagation algorithm*) para averiguar cómo debe reajustar su configuración de parámetros internos empleados para procesar la representación [45] de cada capa a partir de la representación de la capa anterior.

[42]

3.2.1 Propagación hacia atrás para entrenar arquitecturas de múltiples capas

La propagación hacia atrás es el alma del entrenamiento de redes neuronales. Consiste en afinar los pesos de una red neuronal mediante el análisis del error obtenido en iteraciones anteriores. Una afinación de los pesos adecuada nos proporciona tasas de errores menores, haciendo el modelo más consistente gracias a su generalización. [42][48]

La ecuación que se emplea en el proceso de propagación hacia atrás puede aplicarse repetidas veces para propagar gradientes a través de todos módulos, desde la salida (donde la red realiza su predicción) hasta el inicio (allí donde se le proporciona la información). Tras este procedimiento, ya se puede obtener el valor de los gradientes relacionados con los pesos de cada módulo. [42] [48]

Muchas de las aplicaciones del DL emplean arquitecturas de redes neuronales con prealimentación (*feedforward*) (Figura 3-7) en las cuales aprenden a mapear una entrada de tamaño fijo (como pudiera ser una imagen) a una salida de tamaño fijo también (como pudiera ser la probabilidad calculada para cada una de las posibles categorías).

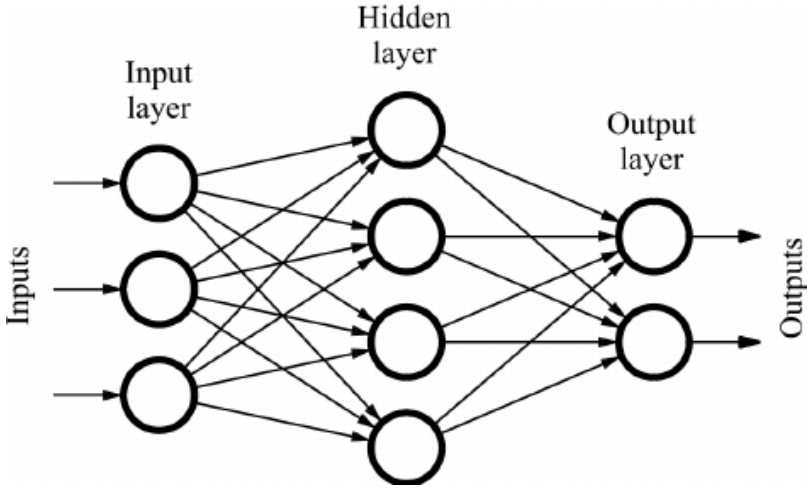

Figura 3-7. Ejemplo sencillo de Red Neuronal con prealimentación (extraída de [49]).

En el paso de una capa a la siguiente, un grupo de nodos calcula una suma ponderada de sus datos de entrada de la capa anterior y pasan el resultado a través de una función no-lineal. Actualmente la función no-lineal más extendida es la unidad lineal rectificada (*rectified linear unit*, ReLU), que consiste en un rectificador de media onda $f(z) = \max(z, 0)$ (Figura 3-8). En el pasado se usaban funciones no-lineales más suaves tales como $1/(1 + \exp(-z))$ o $\tanh(z)$, pero la ReLU suele aprender mucho más rápido en redes neuronales de muchas capas. Aquellos nodos que no son de entrada o salida, llamados nodos ocultos se encargan de distorsionar la entrada de manera no-lineal de tal forma que para cuando se llega la última capa, las categorías acaban siendo separables de manera lineal. [42] [50].

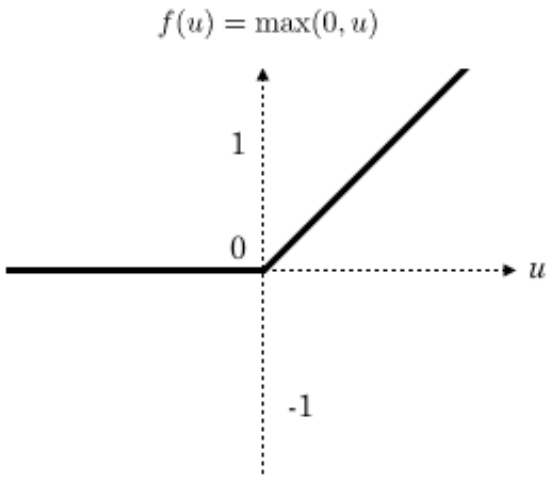

Figura 3-8. Función de activación ReLU (extraída de [87]).

3.2.2 CNN

Las redes neuronales convolucionales (también conocidas como *convolutional neural networks* o CNN) son un tipo de redes neuronales especializadas diseñadas particularmente para trabajar con información que venga en un formato de múltiples matrices, como podría ser uma imagen RGB compuesta por una matriz de intensidades de píxel por cada canal de color. [42] [53]

Las redes neuronales profundas sacan partido de la propiedad de que muchas señales naturales [54] están formadas por jerarquías compositivas, en las cuales los rasgos de alto nivel se obtienen componiendo aquellos rasgos de bajo nivel. Estas jerarquías están presentes en imágenes, en el habla humana, sílabas, palabras y frases. La capa de reducción (*pooling*) se coloca generalmente después de la capa convolucional. Su utilidad principal radica en la reducción de las dimensiones espaciales del volumen de entrada [88].

La organización de las capas convolucionales y de agrupamientos en las CNN está directamente inspirada por las nociones de células simples y complejas pertenecientes a la neurociencia visual [55]. [42]

La arquitectura de una CNN se estructura en una serie de etapas. Las primeras etapas están formadas por dos clases de capas: capas convolucionales y capas de reducción.

Capas Convolucionales

Las capas convolucionales generan nuevas matrices llamadas mapas de rasgos. El mapa de rasgos acentúa aquellos rasgos singulares de la imagen original. La capa de convolución opera de una manera muy distinta, sobre todo comparada a capas de otras redes neuronales. Esta capa no emplea ni pesos conectados ni ningún tipo de suma ponderada, en cambio, sí que aplica filtros que convierten las imágenes. Estos filtros se denominan filtros convolucionales y gracias al proceso de pasar las imágenes a través de éstos se generan los mapas de rasgos.

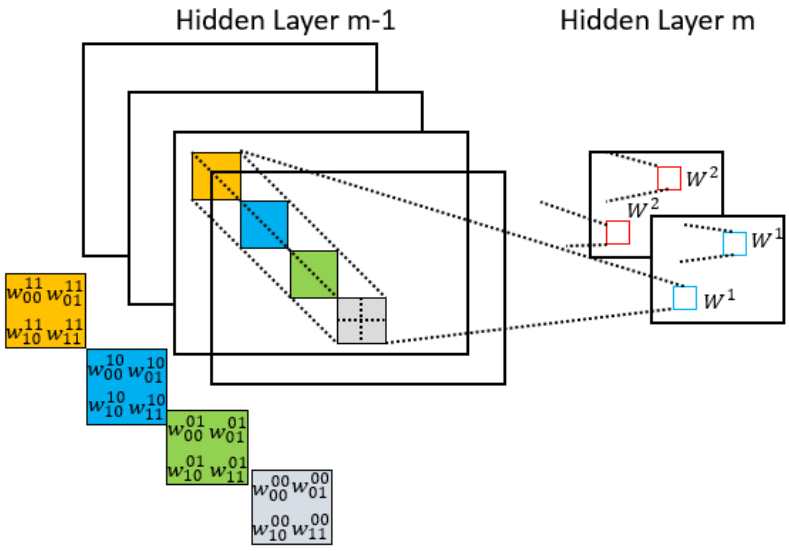

Figura 3-9. Demostración del funcionamiento de una capa convolucional (extraída de [56]).

Todos los elementos en un mapa de rasgos comparten el mismo banco de filtros. Por el contrario, distintos mapas de rasgos de una capa hacen uso de distintos bancos de filtros (Figura 3-8). Existen dos razones para utilizar esta arquitectura. Primeramente, en datos matriciales tales como imágenes, grupos locales de elementos frecuentemente están altamente correlacionados, conformando patrones locales reconocibles que son fácilmente detectables. En segundo lugar, al ser invariante a la ubicación espacial, si un patrón aparece en una región de la imagen, podría aparecer en cualquier otra parte, de ahí la idea de elementos con los mismos pesos en distintas ubicaciones y detectándose el mismo patrón en diferentes regiones.[42]

Capas de Reducción

La capa de reducción reduce el tamaño de la imagen, ya que recombina píxeles vecinos de la imagen en un único valor que los condensa. La reducción de la cantidad datos a procesar es una técnica habitual que muchos otros métodos de procesamiento ya han estado utilizando.

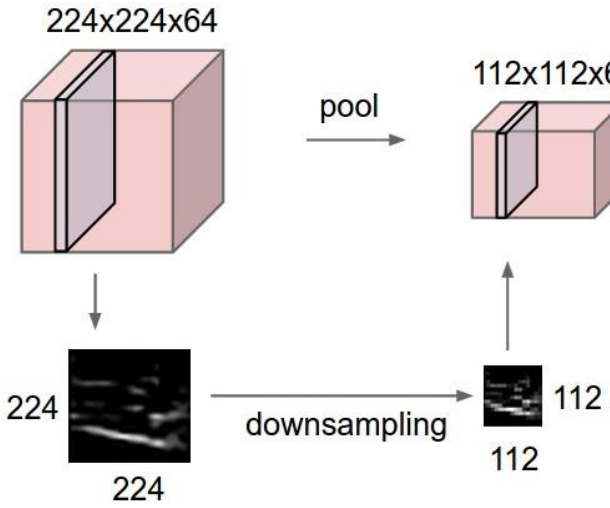

Figura 3-10. El *pooling* submuestrea la entrada, respetando el valor de la profundidad (extraída de [57]).

La operación realizada por esta capa también se conoce como "reducción de muestreo", ya que la reducción de tamaño conduce también a una pérdida de información. Sin embargo, esta pérdida resulta algo beneficioso para la red porque así se tiene una menor sobrecarga de cálculo para las próximas capas [88].

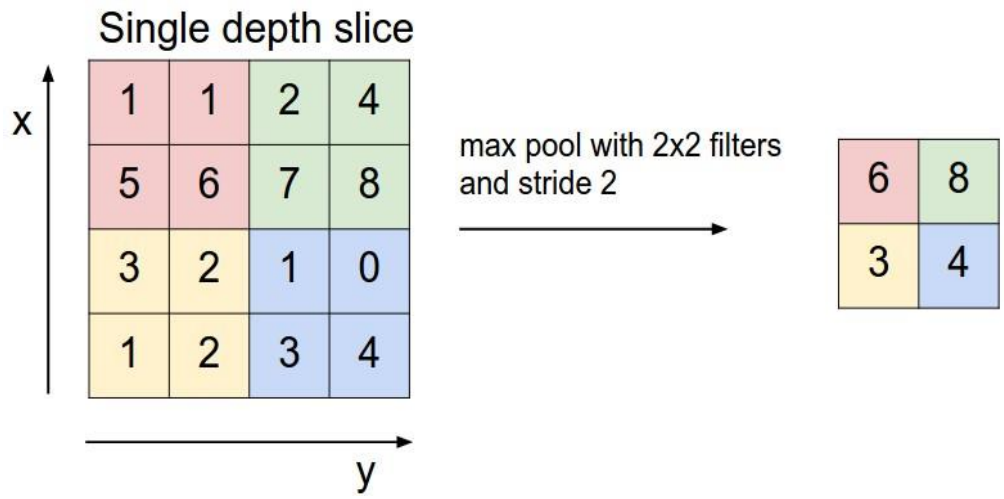

Figura 3-11. Demostración de una operación de submuestreo (extraída de [57]).

La capa de reducción normalmente computa el máximo de una región de un mapa de rasgos (o en ciertas instancias, incluso de varios) (Figura 3-10). Se pierde gran parte de la imagen original, pero se conserva la información importante ya que todavía retenemos los valores altos. Esto reduce la dimensión de la representación y crea una invariancia a distorsiones y pequeños cambios.

La disposición habitual suele ser dos o tres capas convolucionales, junto a operaciones no-lineales y capas de reducción, seguidas por capas de conexión completa (*fully-connected layers*). La transmisión de gradientes por propagación hacia atrás en CNN es tan sencilla como en una red neuronal profunda común, permitiendo a los pesos en los bancos de filtros ser entrenados. [42]

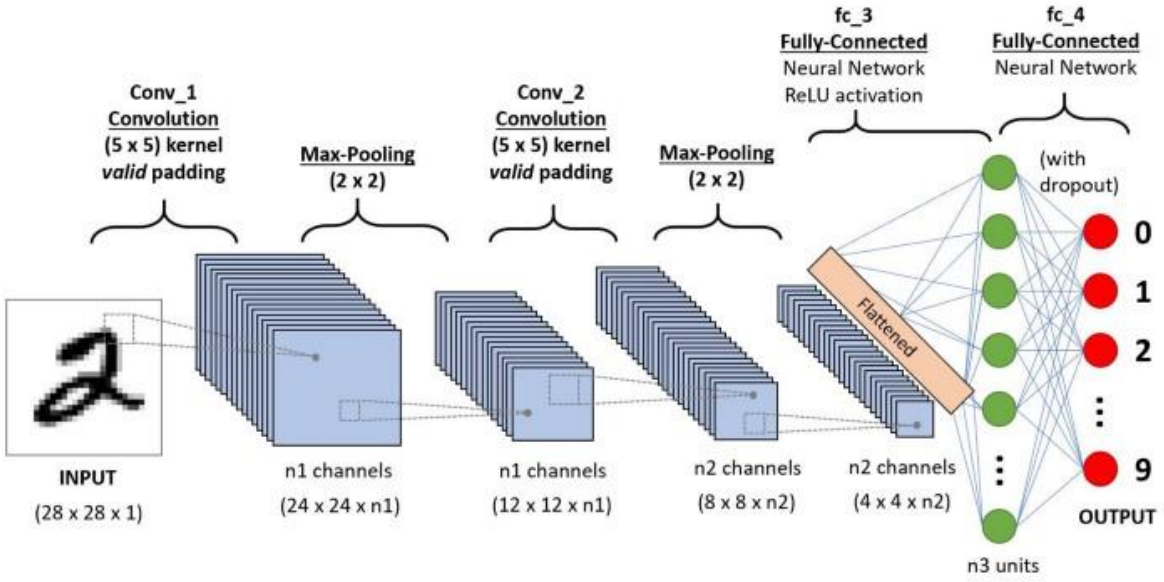

Figura 3-12. Esquema general de una CNN (extraída de [58]).

En resumen, inspiradas por las arquitecturas cerebrales existentes en la naturaleza [59], los algoritmos de CNN aprenden a representar rasgos de manera jerárquica gracias al uso de estrategias como campos locales receptivos (*local receptive fields*) [60], pesos compartidos (mismos pesos para la creación de los mapas de

rasgos), y el submuestreado. Cada banco de filtros puede ser entrenado con métodos con o sin supervisión. Una CNN es capaz de aprender jerarquías de rasgos sólidas automáticamente y mantener cierta invariancia a los desplazamientos translacionales y distorcionales. [46]

3.2.3 Feature Extraction

La extracción de características (*feature extraction*) tiene como propósito reducir el número de atributos de una base de datos por medio de la creación de nuevos atributos obtenidos a partir de los ya existentes. De esta manera, estos nuevos atributos deberían de ser capaces de resumir gran parte de la información contenida en los originales.

La extracción de carácterísticas puede ser implementada de forma automática o manual:

- Extracción de características manual: Requiere que el programador identifique las características que son relevantes para la resolución del problema propuesto y desarrolle una forma para extraerlas. Tener un entendimiento del tema en cuestión puede facilitar a la hora de reconocer que parámetros pueden ser más útiles.
- Extracción de características automática: Emplea algoritmos específicos o redes profundas para extraer características de manera autónoma de imágenes o señales sin necesidad de intervención humana. Esta técnica puede ser muy útil cuando quieres pasar rápidamente de la obtención de datos a desarrollar los algoritmos de deep learning.

Las primeras capas de las redes neuronales profundas en tiempos recientes se usan para la extracción de características en imágenes. Para señales temporales, la extracción de características requiere de conocimiento sobre la materia antes de que se puedan montar modelos de predicción efectivos.

4 METODOLOGÍA

Nothing of me is original. I am the combined effort of everyone I've ever known.

- Chuck Palahniuk -

Una vez llegados a este apartado, todo lo que se verá a partir de ahora será mucho más práctico ya que contamos con los conocimientos básicos expuestos en capítulos anteriores para un fácil entendimiento de toda la metodología. En este capítulo se verán los distintos pasos realizados para llegar hasta el resultado final, el cual será analizado en el capítulo posterior.

4.1 Selección de las Imágenes y Almacenamiento

El primer paso antes de ni siquiera empezar a programar es organizar las imágenes de la base de datos. A finales del Capítulo 2 ya se había comentado la desigualdad presente en el número de imágenes según el tipo. Para evitar posibles sesgos, se ha decidido que se usará la misma cantidad de imágenes para cada clase. Luego, viendo la existencia de subtipos, que presentan ciertas diferencias entre ellos, se ha optado por tener dos tipos de bases de datos: una con una única cepa de cada tipo (a) y otra con las múltiples cepas mezcladas (b).

De modo que siguiendo estos criterios, se ha calculado que para la base de datos (a) la cantidad de imágenes por tipo es de 128 (896 imágenes en total), ya que es la cantidad máxima de imágenes de una cepa que todos los tipos pueden alcanzar. Y para la base de datos (b) la cantidad establecida es de 320 (2240 imágenes en total), que es la cantidad máxima que todos los tipos pueden alcanzar. (ver Tabla 2-1 como referencia).

La composición de estas dos bases de datos es la siguiente:

- Base de datos de 128: "*training_images_128_onevariant*"
    - Acervulina: 128 imágenes de la cepa ACE103
    - Brunetti: 128 imágenes de la cepa BRUC
    - Maxima: 128 imágenes de la cepa MAX50
    - Mitis: 128 imágenes de la cepa MITRT
    - Necatrix: 128 imágenes de la cepa NECDF
    - Praeccox: 128 imágenes de la cepa PRAD
    - Tenella: 128 imágenes de la cepa TENCR

- Base de datos de 320: "*training_images_320_mixed*"
    - Acervulina: 100 imágenes de la cepa ACE103, 120 de la ACEH, y 100 de la ACER7.
    - Brunetti: 320 imágenes de la cepa BRUC.
    - Maxima: 105 imágenes de la cepa MAX50, 100 de la MAXH, y 115 de la MAXL.
    - Mitis: 100 imágenes de la cepa MIT30, 100 de la MIT44, y 120 de la MITRT.
    - Necatrix: 100 imágenes de la cepa NEC103, y 220 de la NECDF.
    - Praecox: 100 imágenes de la cepa PRA1D1A, 100 de la PRAD, y 120 de la PRAH.

- Tenella: 120 imágenes de la cepa TENCR, 100 de la TENH, y 100 de la TENMC.

Para acceder de manera cómoda a todas estas imágenes se ha empleado la biblioteca de métodos relativa a *imageDatastore*() [70], que en pocas líneas permite almacenar como una variable del workspace al conjunto de imágenes con las que se desea trabajar e incluso les asigna una etiqueta (que en esta ocasión contiene información sobre la especie a la que pertenece el elemento) en función de qué carpeta procedían.

```
rootFolder = 'training_images_128_onevariant';  %modificable
categories =
{'Acervulina','Brunetti','Maxima','Mitis','Necatrix','Praecox','Tene
lla'};
imds = imageDatastore(fullfile(rootFolder, categories),
'LabelSource', 'foldernames');
```

Y para realizar la lectura de las imágenes de la base de datos, la librería está bien provista para poder extraer todos los datos que nos interesen, que en este caso son el nombre, la etiqueta y la imagen.

```
thisFullFileName = imds.Files{k};
file_class = imds.Labels(k);
fprintf('Reading in %s.\n', thisFullFileName);  %opcional
f = readimage(imds, k);
```

4.2 Preprocesamiento

El primer módulo diseñado de este proyecto fue el referente al preprocesamiento de las imágenes. Es muy común en trabajos relacionados con el machine learning llevar acabo algún tipo de preprocesamiento de datos. El objetivo de paso es dejar los datos preparados para ser pasados directamente a tu modelo de ML para que el análisis y procesamiento le resulte más fácil. En este proyecto en concreto antes de pasárselo a la red neuronal, se realiza el cálculo de varios parámetros morfológicos.

El preprocesamiento de imágenes puede consistir en operaciones tan simples como en reajustar el tamaño de la misma para que todas tengan un mismo tamaño estandarizado, corregir los colores para detectar algún elemento anormal o simplemente pasarla a blanco y negro para calcular su histograma.

En este trabajo se han realizado una combinación de multiples operaciones debido al alto grado de complejidad de la tarea. Como inspiración se partió del esquema base visto en el Ginoris [1] (Figura 4-1) pero le fueron aplicadas modificaciones para hacerlo todavía más efectivo. [62]

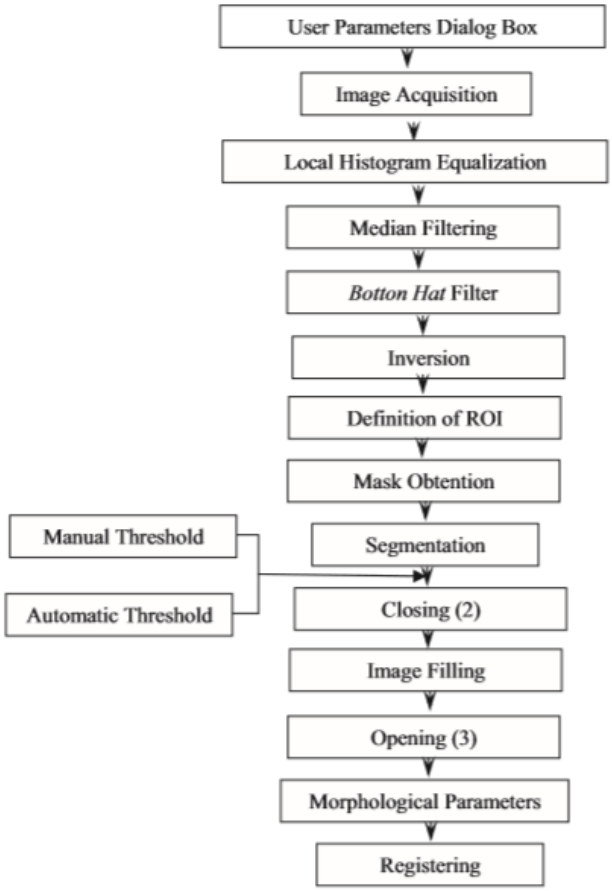

Figura 4-1. Esquema del tratamiento aplicado por el artículo de Ginoris (extraído de [1]).

Durante el desarrollo del algoritmo, se usó la imagen "ACE103 (100).jpg", siendo ésta una imagen ya recortada de un ooquiste de la clase Acervulina y el subtipo "103". Si bien se programó alrededor de esta imagen, se han hecho diversas pruebas para ver que el algoritmo fuera los más generalizado posible y así funcionara con el mayor número de muestras que tenemos a nuestra disposición así como a otras posibles bases de datos que quieran ser usadas en un futuro.

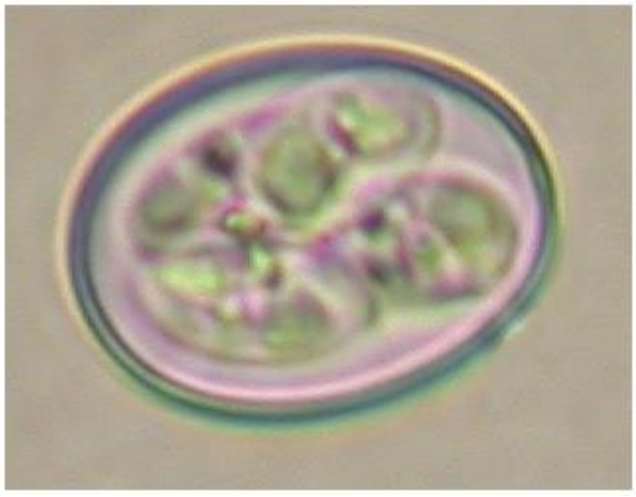

Figura 4-2. Imagen "ACE103 (100).jpg" en su estado original, sin modificaciones.

4.2.1 Obtención de la imagen filtrada

Como es típico en aplicaciones de tratamiento de imagen, lo primero que se hace es el paso a blanco y negro, lo cual se ha optado por la vía más sencilla con el comando de MATLAB $rgb2gray()$, que simplemente convierte la imagen a color original a una imagen en escala de grises.

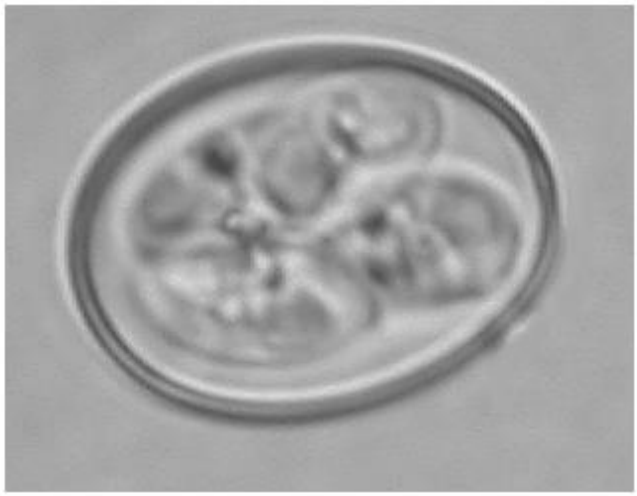

Figura 4-3. La imagen convertida a escala de grises.

Una vez obtenida la imagen en escala de grises, se realiza el filtrado de la imagen. El filtrado es una técnica de mejora de imagen que permite corregir posibles desperfectos de las imágenes y así tratar de homogeneizarlas lo más posible.

El esquema proporcionado por Ginoris indica la realización de una ecualización del histograma de las imágenes, pero en con estas imágenes no se obtienen los resultados esperados lo cual probablemente se deba por las intensidades de gris de los orgánulos que hay dentro del ooquiste.

El primero de los filtros que se aplican se trata de un filtro de mediana. Los filtros de media provocan que los bordes de la imagen se vean más borrosos, pero los filtros de mediana permiten eliminar los valores anormales de la imagen manteniendo la nitidez de la imagen original. Esto se debe a que no les afecta para el cómputo del valor final la presencia de valores anormales.

El siguiente procedimiento consiste pasarle a la foto un filtrado de sombrero inferior (*Bottom Hat Filter)* cuya utilidad consiste en potenciar los objetos oscuros sobre un fondo claro, con lo que el resultado final es que se mejora el contraste de la imagen. En Matlab para aplicarlo es necesaria la creación de un elemento de estructuración de forma de disco. En esta instancia se siguen las recomendaciones de Matlab de restar las imágenes para alcanzar un resultado final deseable [63].

```
se = strel('disk',3);
fbh = imsubtract(imadd(fmedian,imtophat(fmedian,se)),
imbothat(fmedian,se));
```

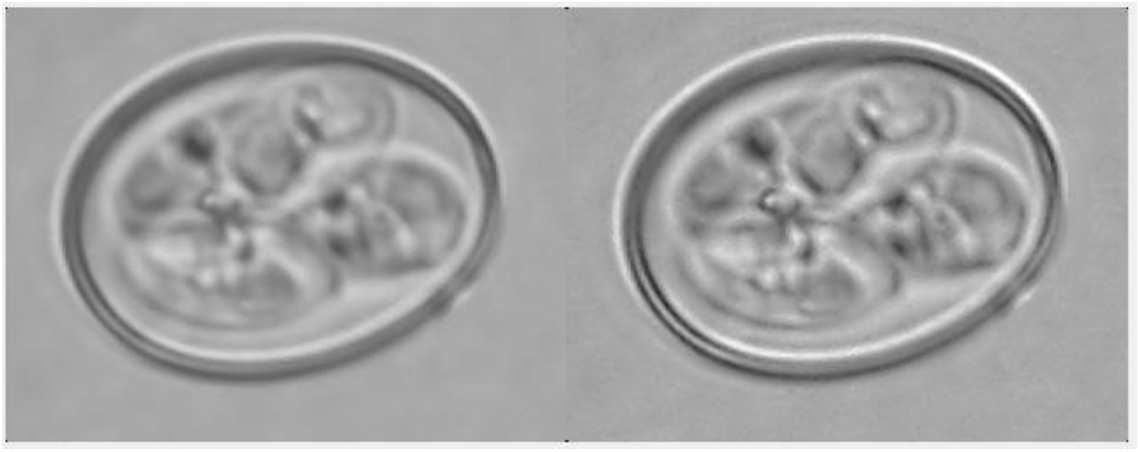

Figura 4-4. Tras la aplicación del filtro de mediana, y posteriormente, del filtrado de sombrero inferior.

Tras todos estos filtros, se invierten los valores de la imagen para configurar la imagen de modo que futuras operaciones se consiga que el ooquiste se quede de color blanco y el fondo sea oscuro.

4.2.2 Segmentación y Post-tratamiento

Primeramente, para poder proseguir con el método planteado por Ginoris, sería necesario definir la región de interés (*ROI*). Este paso se realiza habitualmente de forma manual por el usuario, pero afortunadamente las imágenes con las que se está trabajando ya vienen recortadas, con lo que no se requiere de ninguna modificación extra para este paso.

La aplicación de máscaras en el campo del procesamiento de imagen consiste en multiplicar a la imagen por una matriz de unos y ceros, debido a lo cual, las partes multiplicadas por ceros se pondrán a cero en la imagen de salida (que en este caso significa que se pondrá de color negro) y las partes multiplicadas por unos no sufrirán alteración alguna.

```
fmasc=uint8(255*(froi>105));
```

Figura 4-5. Resultado de aplicación de la máscara a la imagen.

Para quitar gran parte del ruido y dejar la imagen con el menor número posible de huecos, se ha aplicado un doble cierre. La operación de cierre en la matemática morfológica consiste en una dilatación seguida de una erosión [64]. En Matlab esta operación requiere de la creación de un elemento estructura con forma de disco, como ya ocurría con el filtro de sombrero inferior.

```
se = strel('disk',1);
fcie1 = imclose(fmasc,se);
se = strel('disk',4);
fcie2 = imclose(fcie1,se);
```

Figura 4-6. Resultado tras aplicar por primera (izq) y segunda (der) vez el cierre.

Para acabar de rellenar el ooquiste y dejarlo como una silueta blanca que permita fácilmente su reconocimiento, dos procesos más deben ser aplicados a la imagen: el relleno y la apertura.

El relleno como se puede intuir rellena los agujeros (de mayor o menor tamaño) que pueda tener la imagen. En Matlab esto se consigue mediante el comando $imfill()$, al cual se le pasa la imagen de entrada y nos devuelve la imagen rellenada.

```
ffill=imfill(fcie2);
```

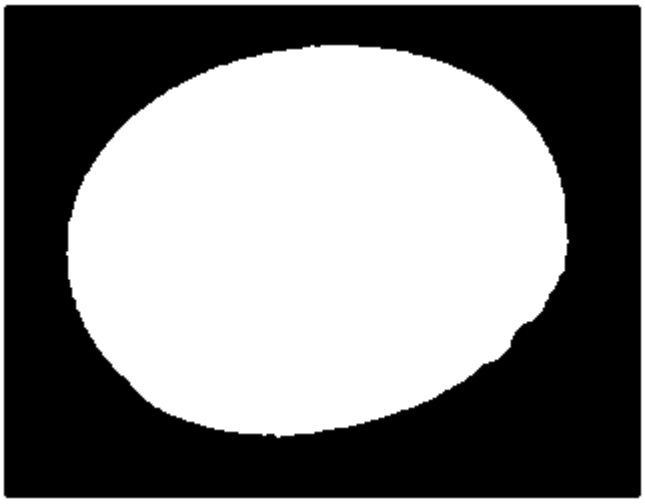

Figura 4-7. Visualización del ooquiste rellenado.

Como ya se ha mencionado antes, ahora toca llevar a cabo el doble proceso de apertura, que consiste en una erosión seguida de una dilatación, lo cual no deja de ser el opuesto al cierre. Aquí se aplica el proceso dos veces, mientras que el algoritmo de Ginoris [1] lo itera tres veces.

```
se = strel('disk',1);
fop1 = imopen(ffill,se);
se = strel('disk',8);
fop2 = imopen(fop1,se);
```

Para el propósito de realizar la apertura de la imagen se dispone del comando *imopen*(), que lleva a cabo la apertura morfológica y una vez más requiere de un elemento estructural de tipo disco, si bien puede verse que el tamaño designado para este caso es mucho mayor.

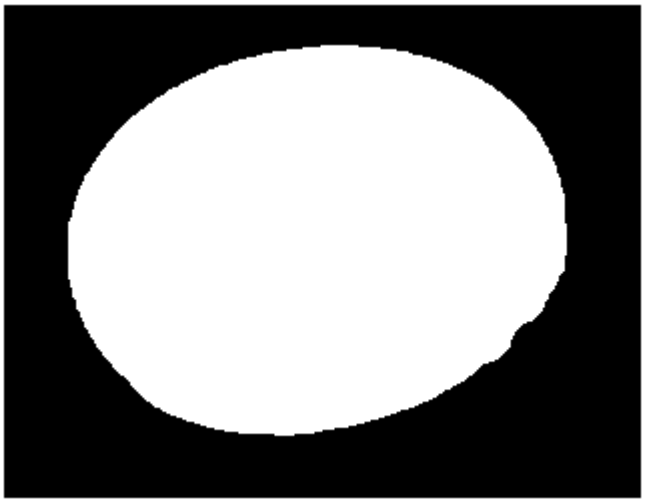

Figura 4-8. Ooquiste tras haber pasado la operación de apertura.

Con esta imagen se da el caso de que prácticamente no se perciben alteraciones entre el rellenado y la apertura, pero eso no es ningún tipo de error, ya que en muchos casos ayuda a proporcionarles una silueta más suave y uniforme, que como se verá a continuación, es una pieza fundamental para que podamos calcular nuestros parámetros morfológicos.

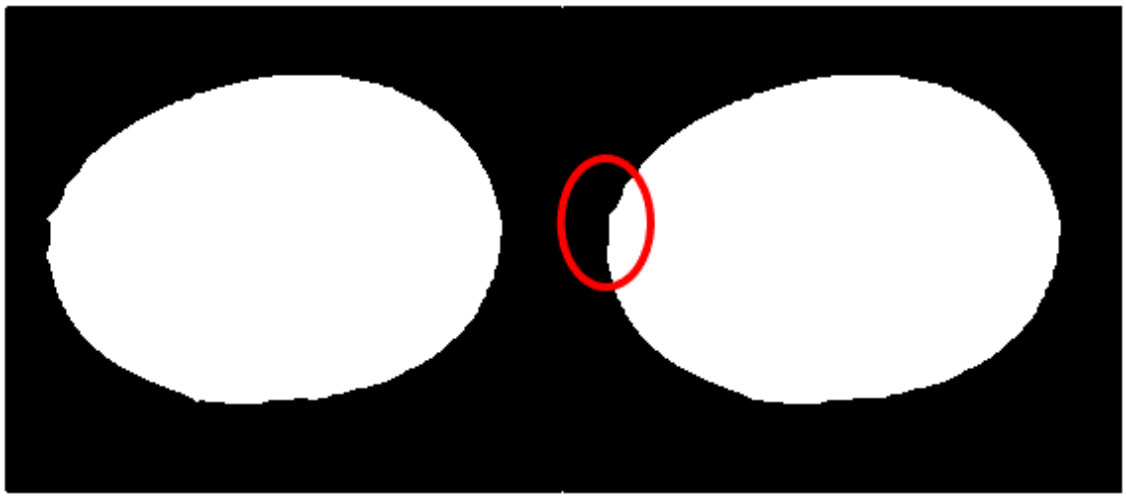

Figura 4-9. Imagen "ACE103 (36).jpg" tras el rellenado y la apertura, en el que se percibe el suavizado de su perfil.

Finalmente, pese a que Ginoris [1] no lo menciona, también se ha calculado una versión de la imagen en que únicamente se tiene el trazado de la silueta del ooquiste, que resultará práctica para el reconocimiento de elipses.

La detección de bordes identifica aquellos cambios bruscos en las intensidades de la imagen. Matlab brinda múltiples opciones para el cálculo de los mismos, pero en este caso se ha llevado a cabo con la función

edge() y con la opción "*canny*". El método de Canny se basa en el cálculo de gradientes y en fijar dos tipos de umbrales para definir un ciclo de histéresis con el que decidir si el punto pertenece a un borde o no [65].

```
fe = edge(fop2,'canny');
```

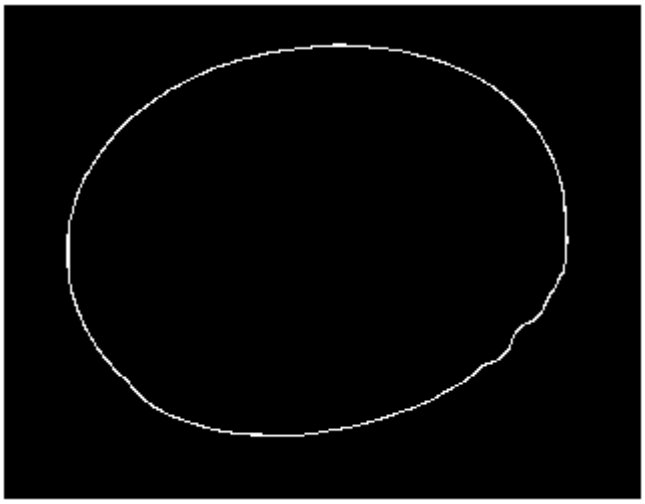

Figura 4-10. Imagen original ya completamente tratada.

4.3 Cálculo de los parámetros

Uno de los alicientes de este trabajo, expuesto al inicio del mismo, era la intención de actualizar el algoritmo planteado por Ginoris [1] y un punto clave para ello era la implementación de métodos de detección de elipses para mejorar los resultados.

La detección de formas es uno de los temas más importantes en el campo de la vision artificial. El algoritmo aquí implementado se basa en la Transformada de Hough, una técnica de extracción de características que se caracteriza por un mecanismo de "votación" con el que se puede identificar la posición de formas como rectas, círculos y elipses [67].

La función (cortesía de Martin Simonovsky [66]) calcula las tres mejores opciones posibles en las que se pueden encontrar elipses. Esta información incluye los datos de las coordenadas del centro de la elipse, la longitud de los ejes mayor y menor, el ángulo de la elipse y la puntuación recibida.

De los parámetros que se emplean en este trabajo para la clasificación, cuatro de ellos se obtienen gracias al reconocimiento de elipses. Éstos son los ejes mayor y menor, el ángulo de la elipse y la excentricidad. La excentricidad no la obtenemos de manera directa, sino que la extraemos a partir de los otros parámetros.

```
% Longitud / Eje Mayor
 eje_M=bestFits(1,3);
% Ancho / Eje Menor
 eje_m=bestFits(1,4);
% Angulo elipse
 angle_elip = bestFits(1,5);
% Excentricidad
 exc = sqrt(1-(eje_m/eje_M)^2);
```

La excentricidad es un parámetro característico de las elipses que determina su forma, pudiendo ser más redondeada o más alargada según su valor.

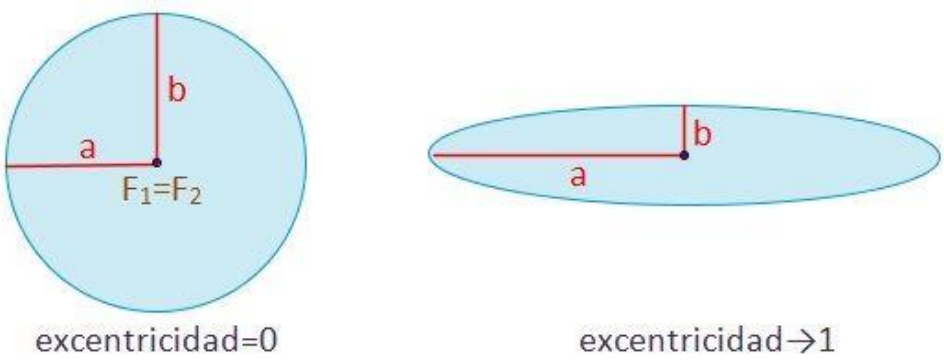

Figura 4-11. Visualización de los dos posibles extremos de la excentricidad en elipses (extraída de [68]).

Los parámetros utilizados restantes son la superficie y la entropía. Para conseguir el valor de la superficie basta con utilizar el comando *bwarea*() aplicada a la imagen tras las aperturas. La idea de la obtener la entropía proviene del artículo de Castañón [2]. La entropía es un parámetro que mide la cantidad de detalle que hay en una imagen. Un valor alto de entropía implica que se trata de una imagen rica en complejidad.

```
% Superficie
 surfc = bwarea(fop2);
% Entropia
 entro = entropy(fbh);
```

Para la obtención del valor de entropía, se ha hecho uso de la imagen resultante de haber aplicado los dos filtros, ya que es fundamental que la imagen conserve sus detalles para poder realizar el cálculo de manera correcta.

Prosiguiendo con los demás parámetros, el siguiente es el perímetro. Para el cálculo del mismo se ha diseñado una función de Matlab independiente al resto del código, ya que comprometería la legibilidad del mismo. Gracias a la obtención del valor del perímetro, también se tiene acceso al factor de forma.

```
% Perimetro
 infoperi = bwperim(fop2);
 peri = calculoPerim_v2(infoperi);
% Factor de Forma
 FF = (peri^2)/(4*pi*surfc);
```

Por último, siguiendo las recomendaciones de Ginoris [1], se computan las propiedades de Feret, como el diámetro de Feret que se define como la distancia entre dos líneas paralelas que son tangenciales al contorno de la proyección del objeto en cuestión [69]. Mediante Matlab se puede acceder a estas propiedades que indican el diámetro y el ángulo de Feret, así como las coordenadas. Debido a que no siempre es capaz de calcular estos valores y que no se quiere que se guarden valores falseados (si no encuentra nada, Matlab guarda ceros en las matrices), utilizando de manera inteligente el if-else se puede solventar este problema.

```
% Propiedades de Feret
 fer = bwferet(fe);
 if isempty(fer)
   maxdiam_fer = NaN;
   maxangle_fer = NaN;
 else
   maxdiam_fer = table2array(fer(1,1));
   maxangle_fer = table2array(fer(1,2));
 end
```

Llegados a este punto, ya se han conseguido calcular todos los parámetros. Todos ellos se almacenan en una matriz llamada *input_table*() que está diseñada para tener tantas filas como parámetros se usen en dicha iteración y tantas columnas como imágenes totales se utilicen. También existe otra tabla titulada *output_table*() en que se asigna a cada elemento un valor u otro según de que especie sea. Estas dos tablas

están estrechamente relacionadas, ya que la columna "n" de ambas, representa el mismo elemento de la base de datos.

4.4 Entrenamiento y Pruebas

La herramienta designada para la creación de la red neuronal es la *Neural Net Pattern Recognition tool,* una herramienta de Matlab [89] que se abre en una ventana una amplia gama de opciones para configurar progresivamente nuestra red neuronal. Ésta se accede mediante la introducción del comando `nprtool`.

La herramienta se basa en *patternnet,* una red neuronal profunda que en su diseño aprovecha las capacidades de las capas convolucionales de las CNN para el reconocimiento de patrones visuales de alto nivel [71].

La navegación por la ventana de *nprtool* es muy sencilla, tras pasar la primera ventana se presenta la opción de introducir la base de datos deseada para que la red neuronal pueda adaptarse y aprender (Figura 4-12).

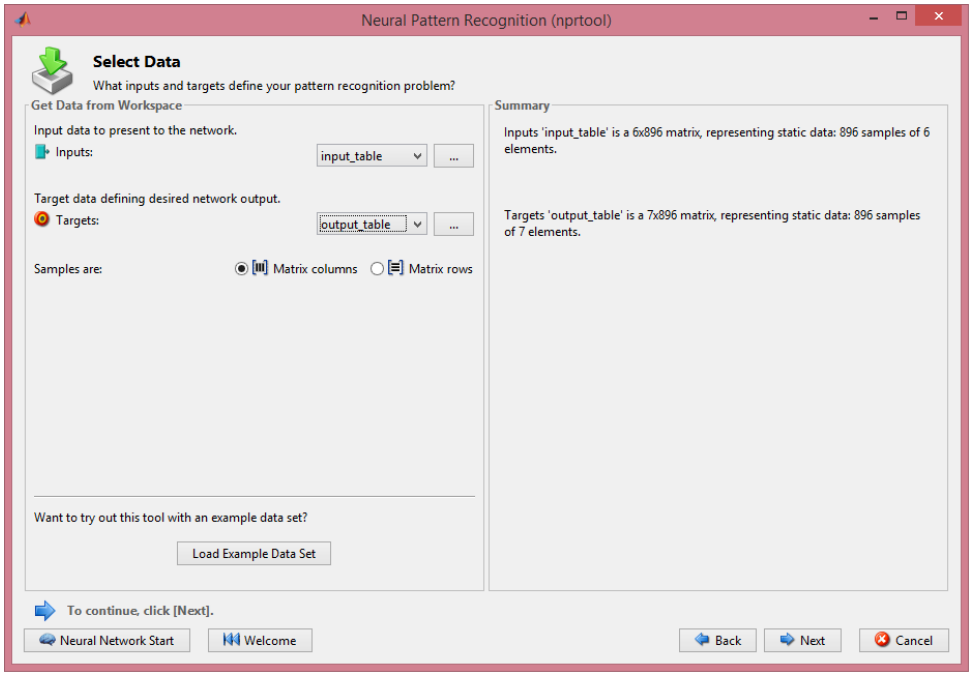

Figura 4-12. Segunda pantalla de *nprtool*, en la que se puede proceder a la introducción de datos

En la siguiente pantalla (Figura 4-13), se comunica en qué porcentaje se quieren repartir los datos etiquetados. Proporcionan tres distintas opciones a las que destinar los porcentajes: *Training, Validation* y *Testing* [72].

- *Training*: Van destinados a que el modelo aprenda (y así adapte los valores de los pesos internos) .
- *Validation*: Esta categoría de los datos ayuda a evaluar la efectividad del modelo desarrollado, pero la máquina no aprende de ellos.
- *Testing*: Se emplean para discernir finalmente la efectividad y precisión del modelo.

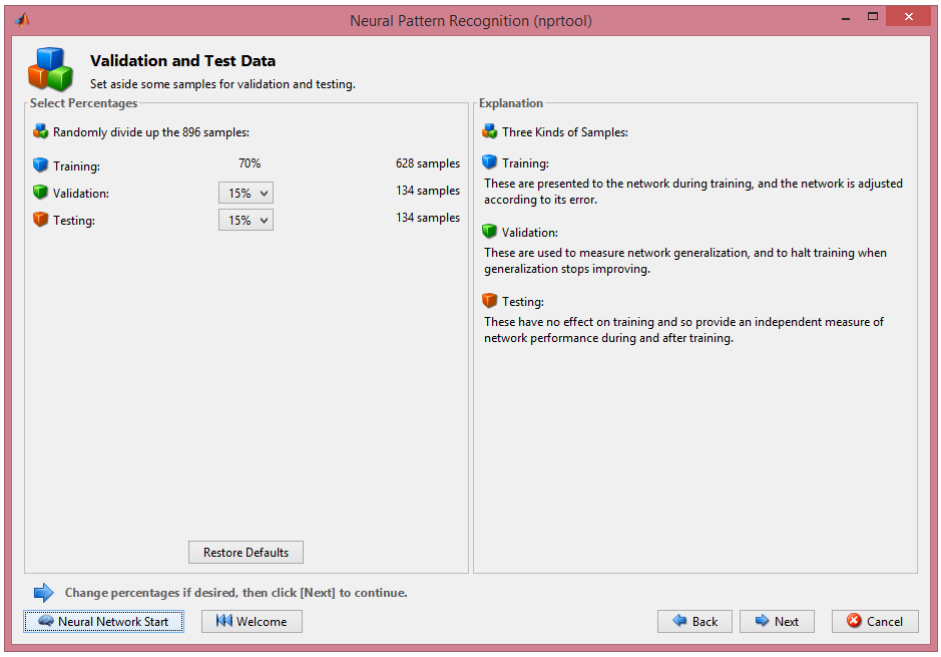

Figura 4-13. Tercera pantalla de *nprtool*, en la que se organiza la repartición de las muestras.

Esta herramienta no distingue en un inicio los diferentes tipos de datos, sino que elige aleatoriamente la repartición de datos entre las tres categorías de uso basándose en los porcentajes. Por regla general, si se tienen pocos hiperparámetros (parámetros que rigen el proceso de entreno al completo [73]) se puede reducir el número de elementos destinados a la validación. De todos modos, esta repartición de la base de datos suele ser muy específica al proyecto en el que se esté trabajando.

La cuarta ventana muestra un sencillo esquema de la estructura de la red neuronal, con la opción de poder modificar el número de neuronas de las capas ocultas. Las capas ocultas se encuentran entre las capas de entrada y de salida (Figura 4-14), y es donde los pesos de las neuronas se balancean de acorde con la información recibida [74].

Para la configuración del número de neuronas, existen varios puntos de partida posibles:

1) Ajustar el número de neuronas a la media entre las de la capa de entrada y la de salida [75].
2) Hacer que el número de neuronas sea 2/3 del tamaño de la capa de entrada más el tamaño de la capa de salida [77].
3) Asegurarse de que el número de neuronas sean menor que el doble del tamaño de la capa de entrada [77].

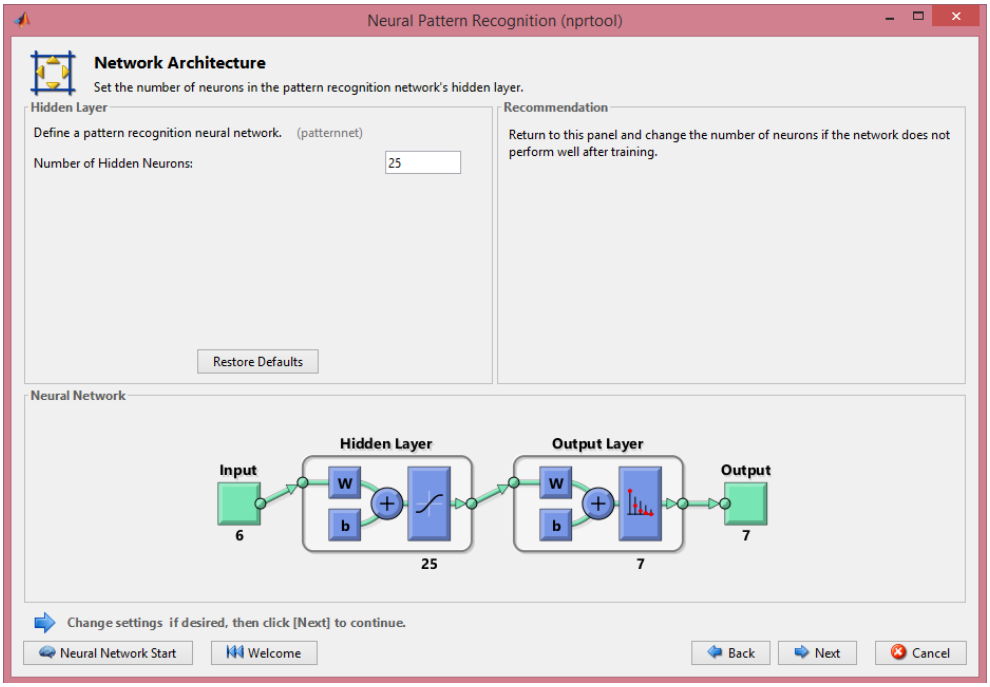

Figura 4-14. Cuarta pantalla de *nprtool*, donde se elige la cantidad de neuronas ocultas para el modelo.

La última etapa del proceso de configuración consiste en las siguientes dos pantallas complementarias la una a la otra para cumplir su función de optimizar los resultados finales mediante el análisis de los mismos.

La primera de ellas (Figura 4-15) ofrece un botón para reentrenar la red neuronal y obtener una configuración distinta de los pesos de las neuronas (lo que conlleva unos resultados distintos) así como otros dos botones para visualizar las **matrices de confusión** y las **curvas ROC** (*Receiver Operating Characteristic*), que también aparecen en la pantalla de entreno. En el campo de la inteligencia artificial, las matrices de confusión son una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado [90]. Las curvas ROC sirven para evaluar la calidad de salida del clasificador. Suelen presentarse con un eje positivo real en el eje Y y una tasa de falsos positivos en el eje X, lo que significa que la esquina superior izquierda de la gráfica es el punto ideal, ya que es donde se da una tasa de falsos positivos de cero y una tasa positiva verdadera de uno [91].

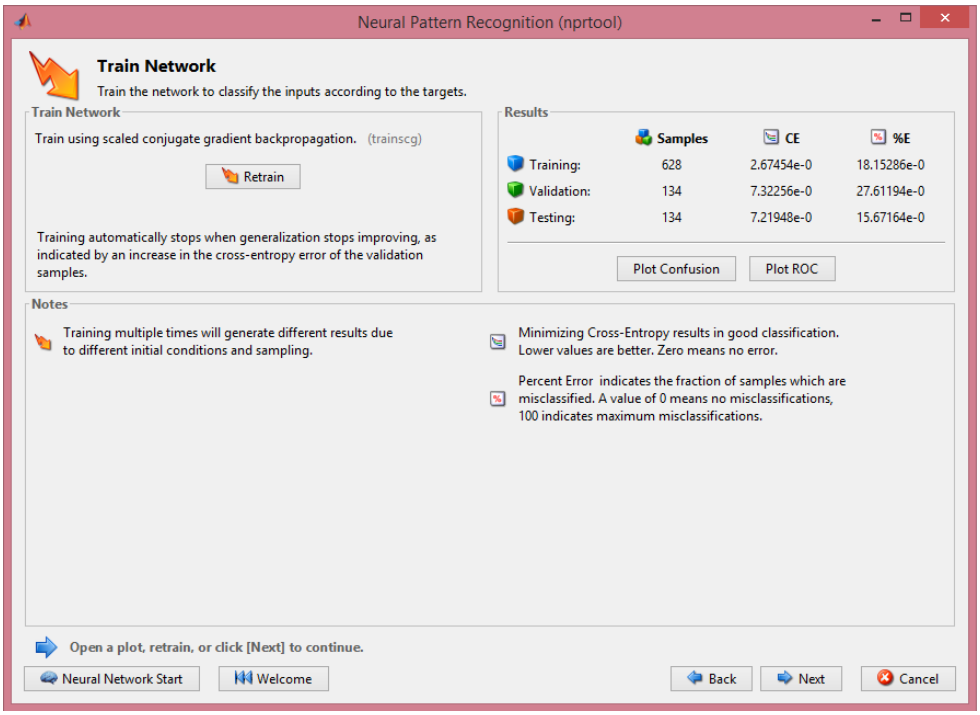

Figura 4-15. Quinta pantalla de *nprtool*, que brinda la opción de reentrenar el modelo y ver los resultados.

Las dos variables de evaluación de la efectividad provistas son la **entropía cruzada** (5.1) y el **porcentaje de error** (5.2). La entropía cruzada cuantifica la diferencia entre dos distribuciones de probabilidad pertenecientes a una misma serie de eventos [78]. En aplicaciones como esta, la entropía cruzada permite saber cómo de lejos se está de un modelo de clasificación ideal [79]. Es habitual su uso como función de pérdidas en modelos de clasificación. En el caso del porcentaje de error, este sencillamente representa la porción de los elementos que fueron clasificados de manera equivocada.

$$H(p,q) = -\sum \forall x p(x) log(q(x)) \tag{5.1}$$

$$E(\%) = \frac{misclassified\ samples}{total\ samples} \tag{5.2}$$

Por último, la pantalla de entreno (Figura 4-16) se abre de manera automática una vez se pulsa el botón *Retrain* y nos proporciona un modificador para el número de *Epochs* que se van a ver en las gráficas. Se llama *Epoch* al recorrido de entrenamiento completo por todo el conjunto de datos [92]. La pantalla ofrece un amplio rango de gráficas para poder realizar una valoración exhaustiva del modelo, que se verán puestas en práctica en el Capítulo 5 ya con las configuraciones finales.

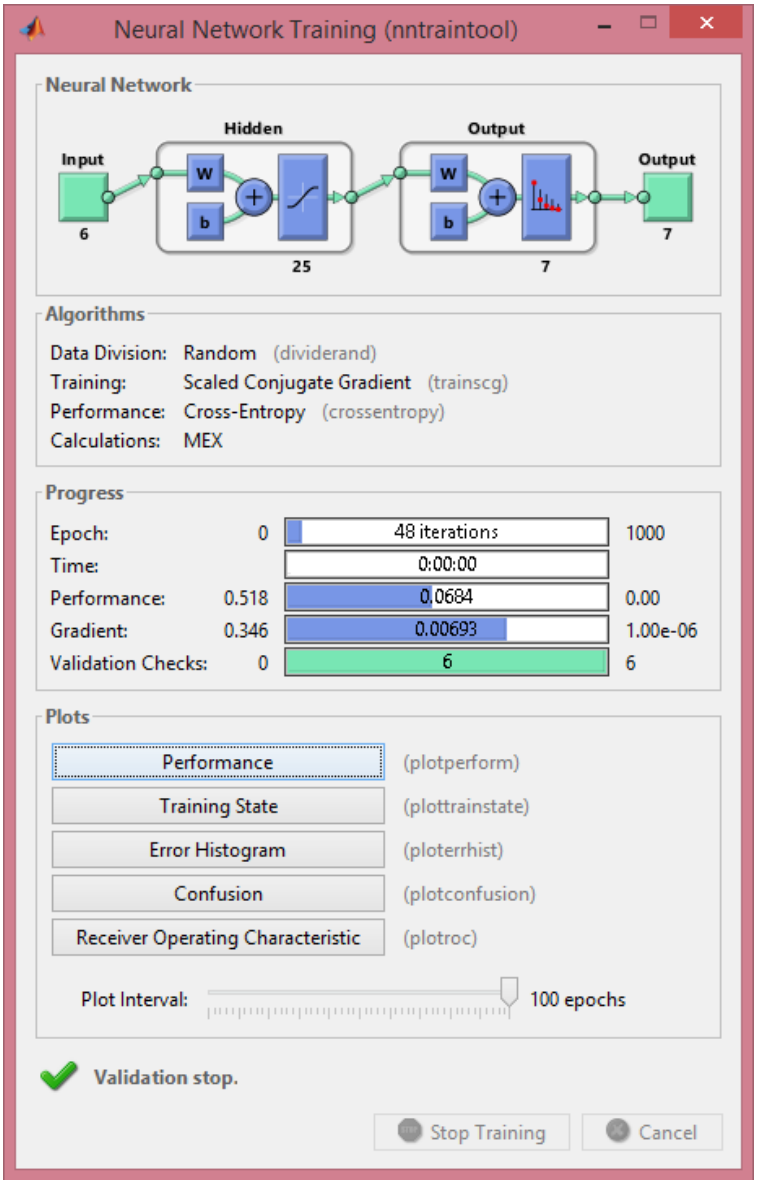

Figura 4-16. Pantalla de entreno, con cinco gráficas disponibles.

5 RESULTADOS

The danger is not to set your goal too high and fail to reach it. It's to set your goal too low and reach it.

- Georges St-Pierre -

Este capítulo va estar dividido en dos partes diferenciadas. Una primera parte en que se muestran los resultados con un espectro de configuración amplio, para ver en qué circunstancias funciona mejor el modelo, y una segunda parte en qué una vez localizadas las mejores condiciones, analizará el comportamiento de las especies dentro del modelo.

La efectividad del procedimiento se determinará según el porcentaje de reconocimiento, definido como el ratio de elementos correctamente clasificados dividido por el número total de elementos. Todas las ejecuciones tendrán asignadas la misma repartición de los porcentajes de datos, siendo esta: *Training* 70%, *Validation* 15%, y *Testing* 15%. La decisión de tomar estos valores viene dada tras una extensa fase de pruebas en la cual esta combinación demostró ser la más equilibrada, mostrando un buen rendimiento y además teniendo una cantidad de datos para el testeo lo suficientemente grande como para no proporcionar resultados no representativos [81].

Los porcentajes de reconocimiento que se presenten en este documento serán obtenidos de la media de los resultados de 25 iteraciones distintas.

5.1 Resultados de las distintas configuraciones

Para un examen exhaustivo de las capacidades de este modelo y ver cómo responde a según que parámetros, en este apartado se van a mostrar los resultados de distintas combinaciones de los mismos.

Van a aplicarse variaciones en tres aspectos: en el número de neuronas de la capa oculta, en el número de parámetros empleados y en la base de datos usada.

El número de parámetros va a variar entre los valores "6","8" y "10", con una asignación fija de los parámetros que aparezcan en las distintas cantidades. Cuando se aplique la cantidad de 6 parámetros, estos serán el eje mayor, el eje menor, el ángulo de la elipse, la excentricidad, la superficie y la entropía. Para cuando sean 8, se añadirán a los ya existentes el perímetro y el factor de forma. Por último, cuando se utilicen 10, se agregarán el máximo diámetro de Feret y el máximo ángulo de Feret (resumido visualmente en la Tabla 5-1).

Tabla 5–1. Presencia de los parámetros según las configuraciones.

	Eje Mayor	Eje Menor	Ángulo Elipse	Excentri-cidad	Super-ficie	Entropía	Perímetro	Factor Forma	Máx. Diam F	Máx Áng F
6 Parámetros	X	X	X	X	X	X				
8 Parámetros	X	X	X	X	X	X	X	X		
10 Parámetros	X	X	X	X	X	X	X	X	X	X

Para el número de neuronas de la capa oculta, se van a utilizar dos valores adaptados al número de parámetros y uno independiente. Los dos primeros surgen de las recomendaciones (1-3) mencionadas con anterioridad (ver Capítulo 4, Apartado 4.4) y el valor independiente, "25", ha sido escogido para poder enseñar un rango decente de funcionamiento y por haber demostrado un buen desempeño en una fase de pruebas previa. Por consecuente los números de neuronas posibles serán los siguientes:

- Para 6 parámetros: 7, 11 y 25.
- Para 8 parámetros: 8, 12 y 25.
- Para 10 parámetros: 9, 14 y 25.

Como ya se ha adelantado, se van a usar dos bases de datos distintas, la base de datos de 128 y la de 320. Ambas vienen ampliamente diseccionadas en el Capítulo 4, Apartado 4.1, incluyendo su composición exacta.

Tabla 5–2. Resultados de la Configuración "6 Parámetros"

Porcentaje de Reconocimiento (%)	Base de Datos de 128	Base de Datos de 320
7 Neuronas	77.90	69.06
9 Neuronas	77.98	69.75
25 Neuronas	78.94	70.17

Tabla 5–3. Resultados de la Configuración "8 Parámetros"

Porcentaje de Reconocimiento (%)	Base de Datos de 128	Base de Datos de 320
8 Neuronas	81.17	70.83
12 Neuronas	79.98	70.09
25 Neuronas	80.46	70.53

Tabla 5–4. Resultados de la Configuración "10 Parámetros"

Porcentaje de Reconocimiento (%)	Base de Datos de 128	Base de Datos de 320
9 Neuronas	80.14	70.51
14 Neuronas	80.45	71.87
25 Neuronas	80.42	70.13

Se percibe a simple vista que el modelo funciona mejor con la base de datos de 128, en torno a un 10% mejor que su contrapartida, lo cual muy probablemente se deba a que en este caso tenga más peso una mayor homogeneidad de las muestras que una mayor cantidad de datos (x2.5).

En relación al número de parámetros, el salto de "6" a un número mayor (ya sea a "8" o a "10") ha supuesto una mejora de entorno al 2.1% para la base de datos de 128. De todos modos las diferencias no han sido realmente destacables, como también ocurre con los número de neuronas.

El mejor resultado ocurrido en una iteración se ha dado con la configuración de 10 parámetros, base de datos de 128 imágenes, y 14 neuronas, en donde se ha alcanzado un porcentaje de reconocimiento global del 88.8% (Figura 5-1).

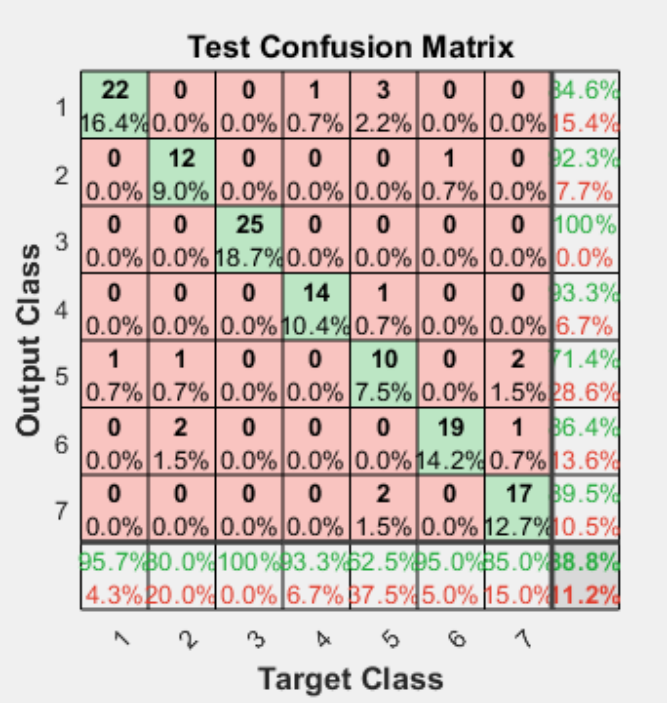

Figura 5-1. Mejor resultado (singular) obtenido por el modelo.

En cambio, el mejor resultado medio lo ha obtenido la configuración de 8 parámetros, base de datos de 128 imágenes, y 8 neuronas, llegando a un resultado medio nada desestimable del 81.17%.

5.2 Resultados por especies

Una vez finalizada la fase anterior, ya se ha podido identificar cuál es la configuración con una mayor precisión media. Ésta servirá de base para esta etapa, en la que se procederá a analizar la habilidad de reconocimiento del modelo según la especie.

Tabla 5–5. Resultados según la especie

Tipos de Eimeria	Porcentaje de Reconocimiento (%)
Acervulina	86.67
Brunetti	79.86
Maxima	97.95
Mitis	90.96
Necatrix	63.09
Praecox	75.48
Tenella	82.51

Prácticamente todas las especies muestran un porcentaje de predicción entorno al 80% o superior, con la única excepción del tipo *Necatrix,* que se queda más de un 15% por debajo de ese umbral. Destaca particularmente la gran precision que exhibe el modelo con la especie *Maxima*, con un 97.95% de aciertos.

Pese a que ya se comentó que el principal evaluador sería el porcentaje de reconocimiento obtenido de las matrices de confusion, se adjuntan todas las gráficas disponibles de la pantalla de entrenamiento para una vision más global de los resultados. Todas ellas fueron obtenidas con la configuración de mayor precisión media.

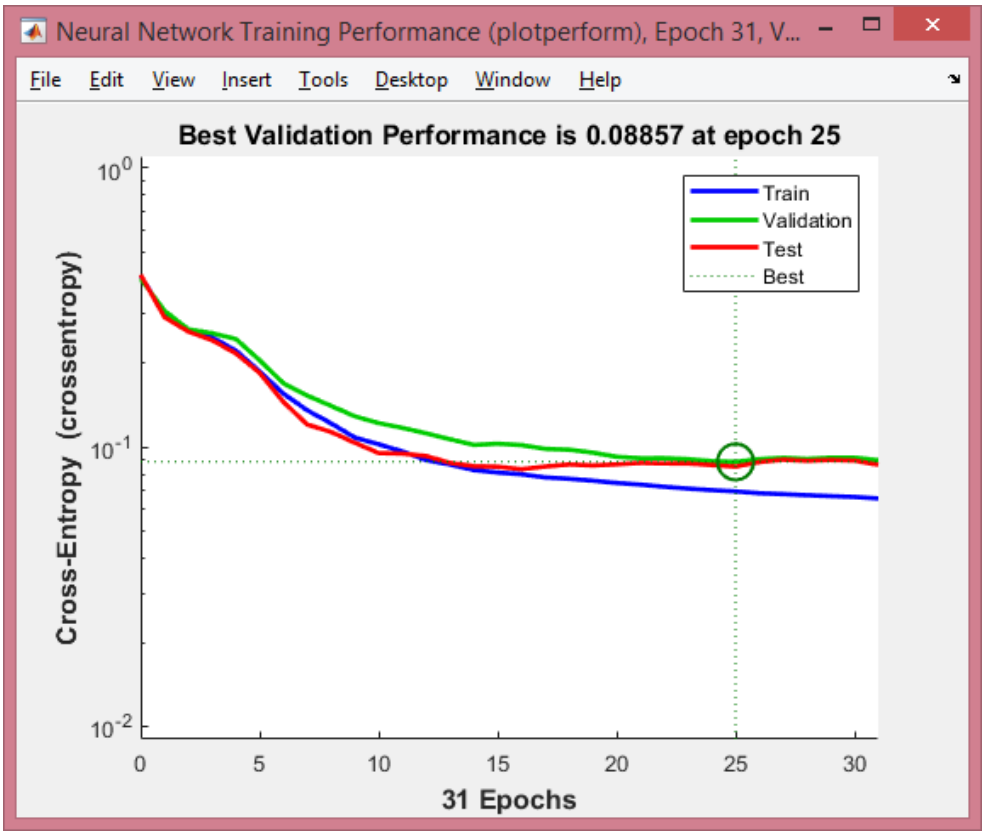

Figura 5-2. Gráfica referente al desempeño, donde se especifica cuándo ha rendido mejor.

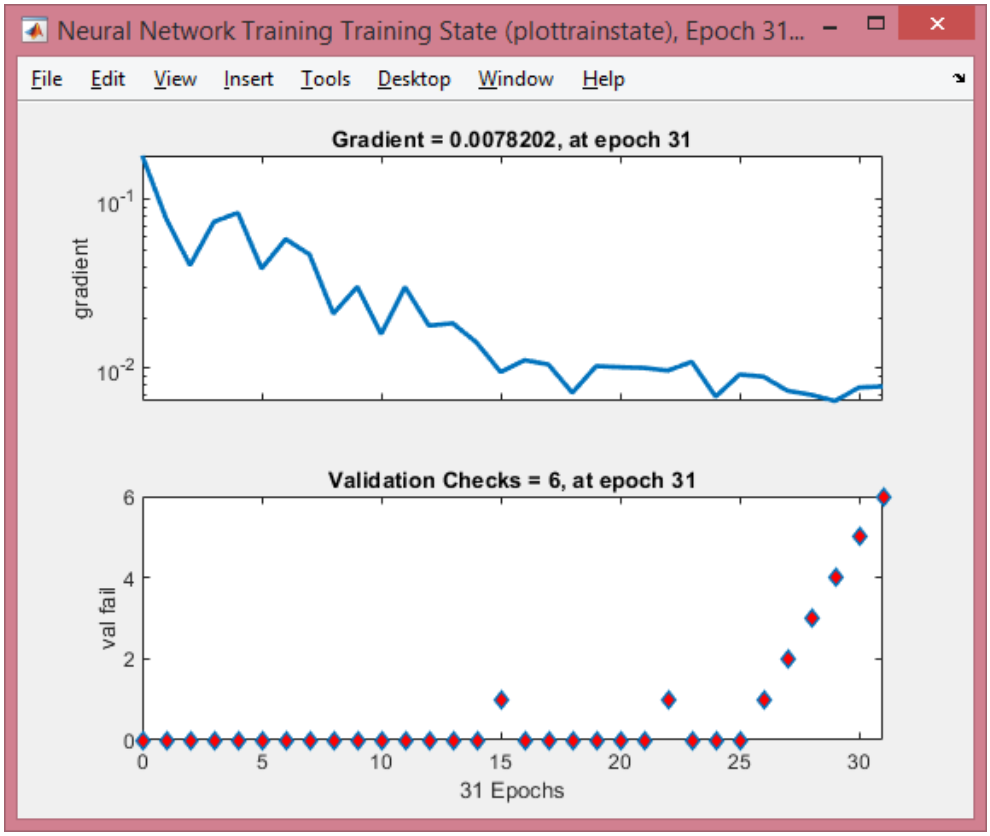

Figura 5-3. Representación del progreso, a través del gradiente y las validaciones, durante el entrenamiento.

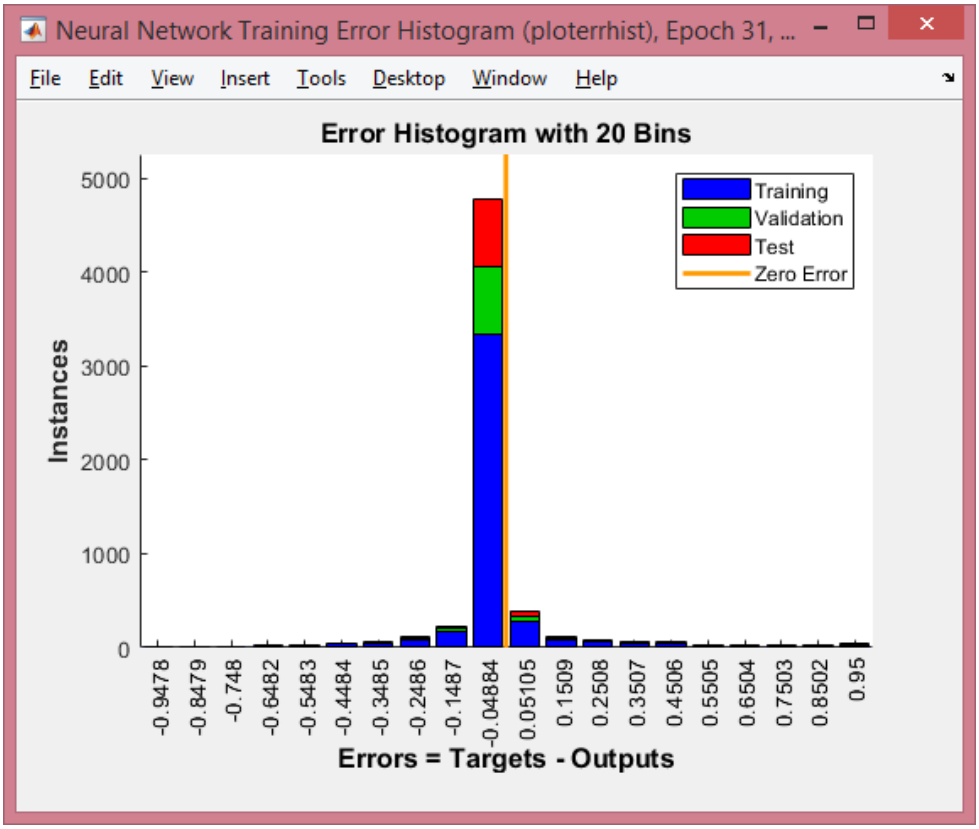

Figura 5-4. Histograma de errores, representando la dispersión de la diferencia entre los valores objetivo y los predichos.

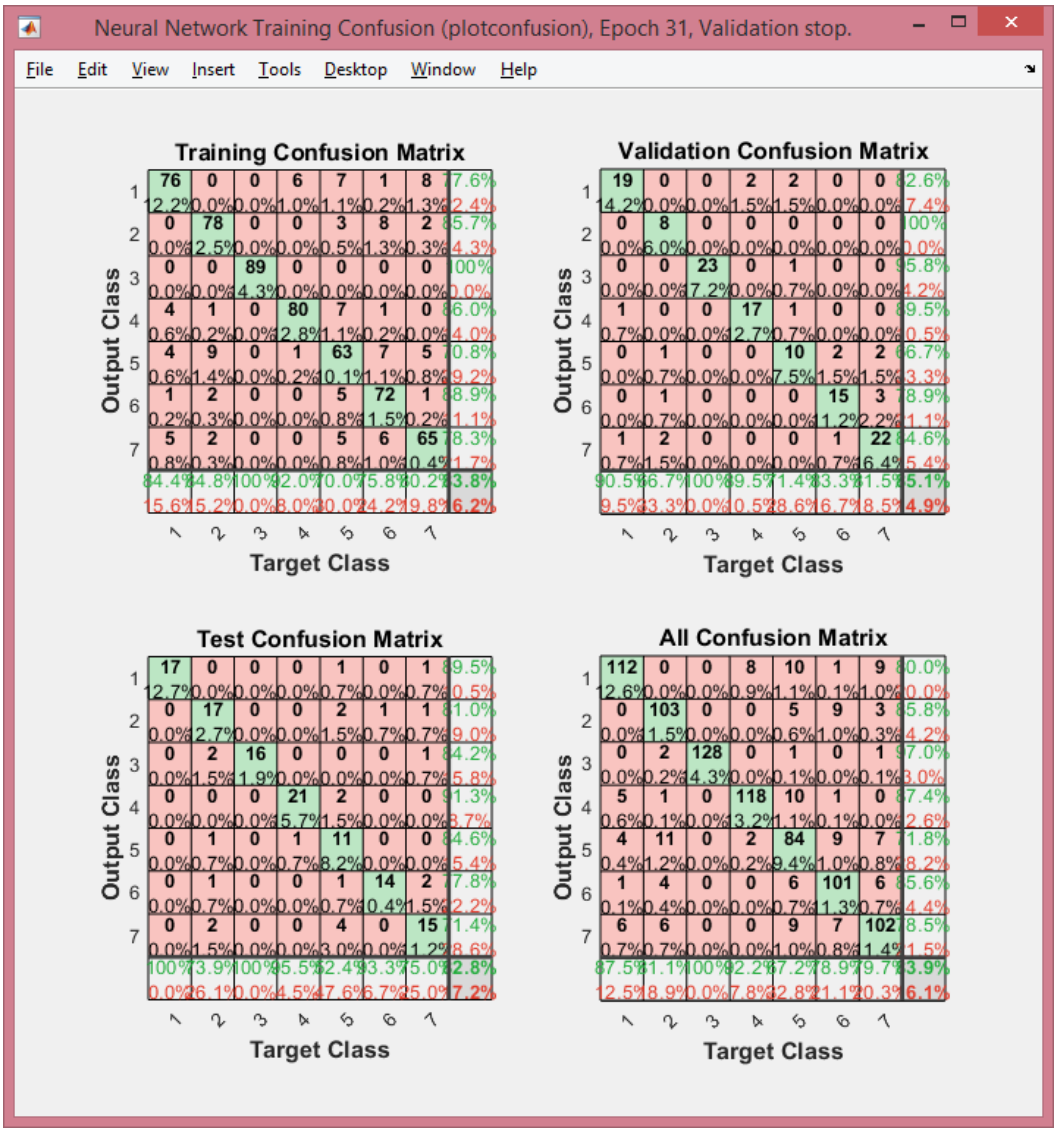

Figura 5-6. Matrices de confusión, presentan una cómoda evaluación de en dónde se suceden los aciertos y errores.

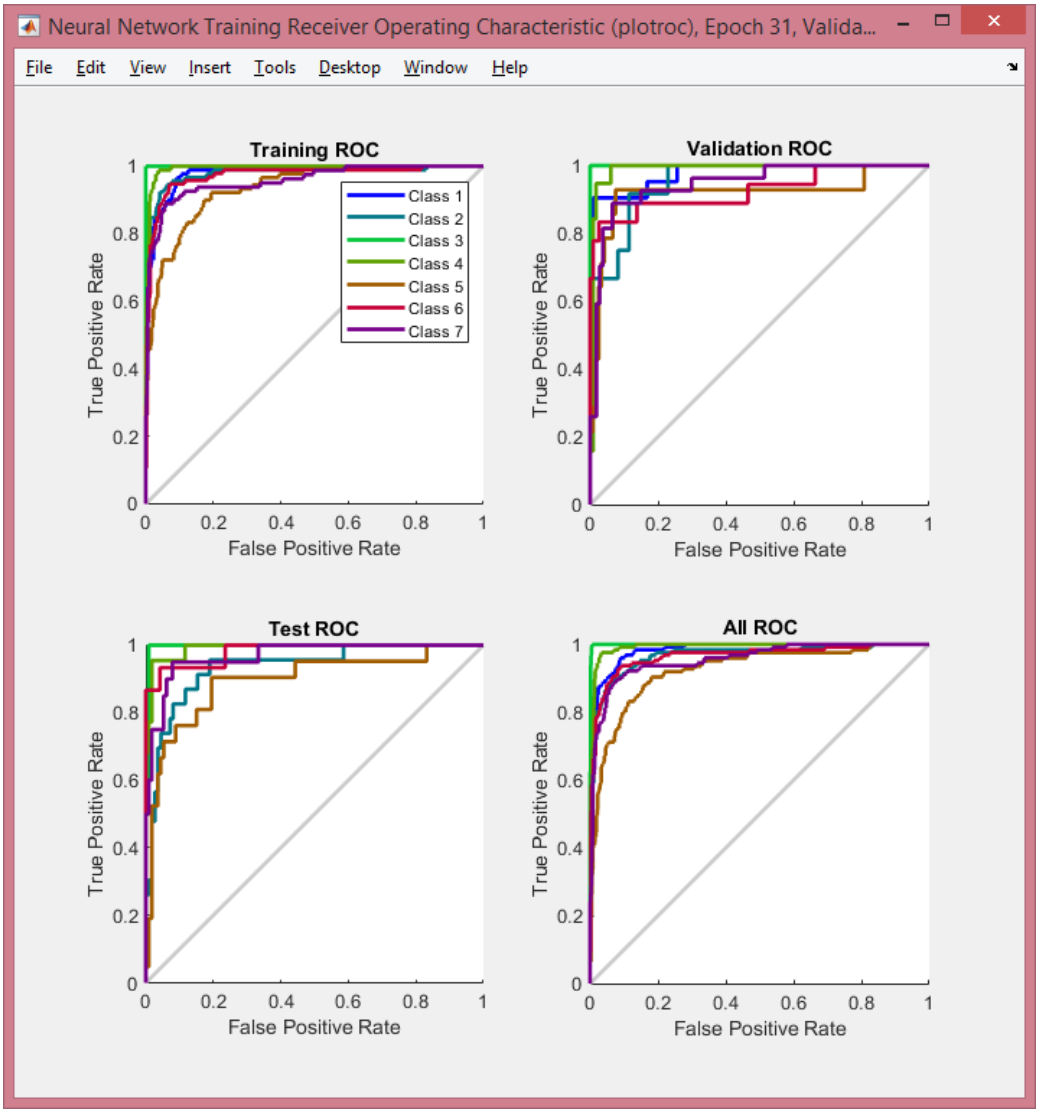

Figura 5-7. Curvas ROC que permiten ver claramente la eficiencia de las clases a lo largo de las distintas fases.

Nota: El hecho de que se hayan podido realizar tantas iteraciones (450 para el primer apartado y 25 para el segundo) de manera rápida y cómoda se debe a la eficiencia de la extracción de características [82] y de cómo está estrcuturado *nprtool*.

6 CONCLUSIONES Y LÍNEAS FUTURAS

To create is to live twice.

- Albert Camus -

El propósito de este trabajo era proponer un método alternativo para el reconocimiento de parásitos basándose en una actualización del algoritmo de Ginoris [1] cumplida mediante la aplicación del reconocimiento de elipses y el uso de redes neuronales para su clasificación.

En la fase de preprocesamiento y reconocimiento de elipses, los resultados han sido más que satisfactorios, ya que la metodología empleada ha sabido representar de manera muy precisa las siluetas de los ooquistes (ver Figura 4-10) en virtud de lo cual el reconocimiento de elipses ha funcionado de la manera deseada a lo largo de toda la base de datos.

Los porcentajes obtenidos para las clasificaciones globales rondan el 80% (considerando el caso de una sola cepa), lo cual demuestra que la red neuronal realmente es capaz de reconocer los ooquistes y sus tipos, tarea que incluso para un humano resultaría dificultosa (en la Figura 2-2 se puede ver a simple vista que algunas de las clases son prácticamente indiferenciables).

Tomando como referencia que para esta misma base de datos el documento original [2] obtuvo un 85.76%, y que aquí se alcanza (con la configuración correcta) un 81.17% (Tabla 5-3), los resultados se pueden considerar más que satisfactorios. También hay que tener en cuenta el hecho de que el artículo estaba realizado por cinco investigadores y éste es un proyecto realizado por una única persona sin tanta experiencia en el sector, con lo que una diferencia de un 4.59% en los resultados finales, puede considerarse un rotundo éxito.

6.1 Líneas Futuras

Este proyecto ha abarcado un amplio abanico de campos, por lo que las opciones de modificación y posible mejora son extensas:

- Modificación de la base de datos: Podrían usarse combinaciones de imágenes distintas a las vistas aquí o incluso hacer caso omiso de limitar a un mismo número de imágenes por clase y utilizar todas las disponibles.
- *Data Augmentation*: Muy popular en el mundo del *machine learning*, ya que permite expandir la base de datos disponible, y como es bien sabido, cuantos más datos tenga el ML, más efectivo será [83].
- Ampliación de los parámetros: Aquí se presenta una selección de parámetros concreta, pero perfectamente se podría ampliar el conjunto ([1] por ejemplo usó del orden de 40 parámetros).
- Otras redes neuronales: Aunque en este documento se haya usado *PatternNet,* pueden usarse otros tipos de redes neuronales, ya sea para mejorar los resultados o la rapidez computacional. Dentro del

propio entorno de Matlab, *alexnet* es una CNN que ha dado muy buenos resultados en el campo de la clasificación de imágenes [84][85].

- Programarlo en Python: Python ofrece actualmente opciones muy potentes con sus librerías en campos como la visión artificial, con lo que realizar un proyecto de corte similar a éste en dicho entorno seguramente proporcione resultados muy satisfactorios [86].

ANEXO A: CÓDIGOS

A.1 Código Base

```
%Codigo de extraccion de caracterisiticas de distintos grupos de imagenes v3
%Por Pedro Soriano Aguado

% Procesado de grupo de imagenes
clear all; close all; clc

%% Introduccion de datos iniciales
num_im = 320; %numero de imagenes por tipo (modificable)
num_cat = 7; %numero de categorias usadas (abierto a modificacion)
num_param = 10; %numero de parametros (modificable)
num_typ = 7; %numero de tipos (absolutos)
i=1; %contador para saber el numero de imagenes trabajadas
input_table = zeros(num_param,num_cat*num_im);
output_table = zeros(num_typ,num_cat*num_im);
%en este caso, el numero de filas será =al numero de parametros, y el
%numero de columnas sera igual al numero de clases por el numero de
%imagenes por clase
rootFolder = 'training_images_320_mixed'; %modificar para usar la otra fase
categories =
{'Acervulina','Brunetti','Maxima','Mitis','Necatrix','Praecox','Tenella'};
%version reducida
imds = imageDatastore(fullfile(rootFolder, categories), 'LabelSource',
'foldernames');

%% Bucle en que se realiza el preprocesado y la obtencion de carasteristicas
for k = 1 : num_im*num_cat
    thisFullFileName = imds.Files{k};
    file_class = imds.Labels(k);
    fprintf('Reading in %s.\n', thisFullFileName);
    f = readimage(imds, k);

    %PREPROCESADO

    %Paso 1: Obtener la imagen en byn
    fgray=rgb2gray(f);
    [g,T]=histeq(fgray);
    % figure;imshow(g);title('Bacteria Ecualizada'); axis tight;

    %Paso 2: Aplicar filtro de mediana
    fmedian = medfilt2(fgray);
    % figure;imshow(fmedian);title('Bacteria Filtro Mediana'); axis tight;

    %Paso 3: "Bottom Hat Filter"
    se = strel('disk',3);
    fbh =
imsubtract(imadd(fmedian,imtophat(fmedian,se)),imbothat(fmedian,se));
    % figure;imshow(fbh);title('Bacteria Gorro Bajo'); axis tight;
```

```
%Paso 4: Inversion
finv= imcomplement(fbh);
% figure;imshow(finv);title('Bacteria Invertida'); axis tight;

%Paso 5: Definicion de la ROI
froi = finv; %vienen ya recortadas las imágenes
% figure;imshow(froi);title('Bacteria Recortada'); axis tight;

%Paso 6: Obtencion de la mascara y Segmentacion (Umbral Manual)
fmasc=uint8(255*(froi>105));
% figure;imshow(fmasc);title('Bacteria Enmascarada'); axis tight;

%Paso 7: Doble cierre
se = strel('disk',1);
fcie1 = imclose(fmasc,se);
se = strel('disk',4);
fcie2 = imclose(fcie1,se);
% figure;imshow([fcie1,fcie2]);title('Bacteria Cerrada (2)'); axis tight;

%Paso 8: Relleno
ffill=imfill(fcie2);
% figure;imshow(ffill);title('Bacteria Rellenada'); axis tight;

%Paso 9: Triple apertura de la imagen
se = strel('disk',1);
fop1 = imopen(ffill,se);
se = strel('disk',8);
fop2 = imopen(fop1,se);
% figure;imshow([fop1,fop2]);title('Bacteria Abierta (3)'); axis tight;

%Version de solo bordes
fe = edge(fop2,'canny');


% PARAMETROS MORFOLOGICOS
 params.minMajorAxis = 50;  %utiliza esta sentencia en la demostracion
 params.maxMajorAxis = 800;  %con valores [200,500]
bestFits = ellipseDetection_mod(fe, params);

% Longitud / Eje Mayor
eje_M=bestFits(1,3);

% Ancho / Eje Menor
eje_m=bestFits(1,4);

% Angulo elipse
angle_elip = bestFits(1,5);

% Excentricidad
exc = sqrt(1-(eje_m/eje_M)^2);

% Superficie
surfc = bwarea(fop2);

%Entropia
entro = entropy(fbh); %usa la que está en byn y ya filtrada
```

```
    % Perimetro
    infoperi = bwperim(fop2);
    peri = calculoPerim_v2(infoperi);

    % Factor de Forma
    FF = (peri^2)/(4*pi*surfc);

    % Propiedades de Feret
    fer = bwferet(fe); %contiene varios parametros
     if isempty(fer)
         maxdiam_fer = NaN;
         maxangle_fer = NaN;
     else
         maxdiam_fer = table2array(fer(1,1));
         maxangle_fer = table2array(fer(1,2));

     end


    % Almacenamiento
    input_table(1,k) = eje_M; %Longitud
    input_table(2,k) = eje_m; %Ancho
    input_table(3,k) = angle_elip; %Angulo elipse
    input_table(4,k) = exc; %Excentricidad
    input_table(5,k) = surfc; %Superficie
    input_table(6,k) = entro; %Entropia

    switch file_class
        case "Acervulina"
            output_table(1,k) = 1;
        case "Brunetti"
            output_table(2,k) = 1;
        case "Maxima"
            output_table(3,k) = 1;
        case "Mitis"
            output_table(4,k) = 1;
        case "Necatrix"
            output_table(5,k) = 1;
        case "Praecox"
            output_table(6,k) = 1;
        case "Tenella"
            output_table(7,k) = 1;
    end
end
```

A.2 Cálculo del Perímetro

```
function perim = calculoPerim_2(imgbw)

%Paso 1: Buscar punto inicial
s = imgbw;
[M,N]=size(s);
ini=1;
for i=1:M
    for j=1:N
```

```
        if s(i,j) && ini
            p0_i=i;
            p0_j=j;
            ini=0;
        end
    end
end

%Paso 2: Recorrer figura e ir sumando
perim = 0;
% Variable que se pone a cero cuando haya dado la vuelta completa al
% contorno asi parar
seguir = 1;
visitado = zeros(M,N); % matriz que te recuerda si has estado ya
ic = p0_i; jc = p0_j;
while seguir
    for di=-1:1
        for dj=-1:1
             % Vecino considerado = actual + diferencial
             ivec = ic+di;
             jvec = jc+dj;
             % Si el vecino no ha sido visitado y es del contorno
             % No tiene en cuenta contornos dobles, tentáculos, cruces...
             if ivec == 0 || jvec ==0 || ivec > M || jvec > N
                 seguir = 0;
                 peri = NaN;
             else
                 if visitado(ivec,jvec) < 1 && s(ivec,jvec) > 0
                     % Se marca como el siguiente
                     isig = ivec; jsig = jvec;      % no se declaran antes
                 end
             end
        end
    end
     % Se suma al perímetro el segmento actual
     perim = perim + sqrt( (isig-ic)^2 + (jsig-jc)^2 );
     % Se comprueba si se ha dado la vuelta
     if isig == p0_i && jsig == p0_j
         % Se desactiva la marca que permite seguir
         seguir = 0;
     else
         % Se actualiza la posicion
          ic = isig; jc = jsig;
          % Se actualiza la matriz de visitados (actual marca como visitado)
          visitado(isig,jsig) = 1;
     end

% Resultado
% fprintf('\nEl perimetro calculado es: %f\n', perim);


end
```

A.3 Detección de Elipses *(cortesía de [66])*

```
function bestFits = ellipseDetection_mod(img, params)
% ellipseDetection: Ellipse detection
%
% Overview:
% --------
% Fits an ellipse by examining all possible major axes (all pairs of points)
and
% getting the minor axis using Hough transform. The algorithm complexity
depends on
% the number of valid non-zero points, therefore it is beneficial to provide
as many
% restrictions in the "params" input arguments as possible if there is any
prior
% knowledge about the problem.
%
% The code is reasonably fast due to (optional) randomization and full code
vectorization.
% However, as the algorithm needs to compute pairwise point distances, it can
be quite memory
% intensive. If you get out of memory errors, either downsample the input
image or somehow
% decrease the number of non-zero points in it.
% It can deal with big amount of noise but can have severe problem with
occlusions (major axis
% end points need to be visible)
%
% Input arguments:
% --------
% img
%   - One-channel input image (greyscale or binary).
% params
%   - Parameters of the algorithm:
%       minMajorAxis: Minimal length of major axis accepted.
%       maxMajorAxis: Maximal length of major axis accepted.
%        rotation, rotationSpan: Specification of restriction on the angle of
the major axis in degrees.
%                                If rotationSpan is in (0,90), only angles
within [rotation-rotationSpan,
%                               rotation+rotationSpan] are accepted.
%       minAspectRatio: Minimal aspect ratio of an ellipse (in (0,1))
%       randomize: Subsampling of all possible point pairs. Instead of
examining all N*N pairs, runs
%                  only on N*randomize pairs. If 0, randomization is turned
off.
%       numBest: Top numBest to return
%       uniformWeights: Used to prefer some points over others. If false,
accumulator points are weighted
%                       by their grey intensity in the image. If true, the
input image is regarded as binary.
%       smoothStddev: In order to provide more stability of the solution, the
accumulator is convolved with
%                     a gaussian kernel. This parameter specifies its
standard deviation in pixels.
%
% Return value:
% --------
% Returns a matrix of best fits. Each row (there are params.numBest of them)
contains six elements:
% [x0 y0 a b alpha score] being the center of the ellipse, its major and
minor axis, its angle in degrees and score.
```

```
%
% Based on:
% --------
% - "A New Efficient Ellipse Detection Method" (Yonghong Xie Qiang , Qiang Ji
/ 2002)
% - random subsampling inspired by "Randomized Hough Transform for Ellipse
Detection with Result Clustering"
%   (CA Basca, M Talos, R Brad / 2005)
%
% Update log:
% --------
% 1.1: More memory efficient code, better documentation, more parameters,
more solutions possible, example code.
% 1.0: Initial version
%
%
% Author: Martin Simonovsky
% e-mail: <mys007@seznam.cz>
% Release: 1.1
% Release date: 25.7.2013
%
%
% --------
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are
% met:
%
%     * Redistributions of source code must retain the above copyright
%       notice, this list of conditions and the following disclaimer.
%     * Redistributions in binary form must reproduce the above copyright
%       notice, this list of conditions and the following disclaimer in
%       the documentation and/or other materials provided with the
distribution
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.
    % default values
    if nargin==1; params=[]; end
    % - parameters to contrain the search
    if ~isfield(params,'minMajorAxis');     params.minMajorAxis = 10; end
    if ~isfield(params,'maxMajorAxis');     params.maxMajorAxis = 200; end
    if ~isfield(params,'rotation');            params.rotation = 0; end
    if ~isfield(params,'rotationSpan');        params.rotationSpan = 0; end
    if ~isfield(params,'minAspectRatio');    params.minAspectRatio = 0.1; end
    if ~isfield(params,'randomize');        params.randomize = 2; end
    % - others
    if ~isfield(params,'numBest');            params.numBest = 3; end
    if ~isfield(params,'uniformWeights');   params.uniformWeights = true; end
    if ~isfield(params,'smoothStddev');        params.smoothStddev = 1; end
    eps = 0.0001;
    bestFits = zeros(params.numBest,6);
    params.rotationSpan = min(params.rotationSpan, 90);
```

```
    H = fspecial('gaussian', [params.smoothStddev*6 1], params.smoothStddev);
    [Y,X]=find(img);
    Y = single(Y); X = single(X);
    N = length(Y);

   % fprintf('Possible major axes: %d * %d = %d\n', N, N, N*N);
    % compute pairwise distances between points (memory intensive!) and
filter
    % TODO: do this block-wise, just appending the filtered results (I,J)
    distsSq = bsxfun(@minus,X,X').^2 + bsxfun(@minus,Y,Y').^2;
    [I,J] = find(distsSq>=params.minMajorAxis^2 &
distsSq<=params.maxMajorAxis^2);
    idx = I<J;
    I = uint32(I(idx)); J = uint32(J(idx));

   % fprintf('..after distance constraint: %d\n', length(I));

    % compute pairwise angles and filter
    if params.rotationSpan>0
        tangents = (Y(I)-Y(J)) ./ (X(I)-X(J));
        tanLo = tand(params.rotation-params.rotationSpan);
        tanHi = tand(params.rotation+params.rotationSpan);
        if tanLo<tanHi
            idx = tangents > tanLo & tangents < tanHi;
        else
            idx = tangents > tanLo | tangents < tanHi;
        end
        I = I(idx); J = J(idx);
      %  fprintf('..after angular constraint: %d\n', length(I));
    else
      %  fprintf('..angular constraint not used\n');
    end

    npairs = length(I);
    % compute random choice and filter
    if params.randomize>0
        perm = randperm(npairs);
        pairSubset = perm(1:min(npairs,N*params.randomize));
        clear perm;
       % fprintf('..after randomization: %d\n', length(pairSubset));
    else
        pairSubset = 1:npairs;
    end

    % check out all hypotheses
    for p=pairSubset
        x1=X(I(p)); y1=Y(I(p));
        x2=X(J(p)); y2=Y(J(p));

        %compute center & major axis
        x0=(x1+x2)/2; y0=(y1+y2)/2;
        aSq = distsSq(I(p),J(p))/4;
        thirdPtDistsSq = (X-x0).^2 + (Y-y0).^2;
        K = thirdPtDistsSq <= aSq; % (otherwise the formulae in paper do not
work)
        %get minor ax propositions for all other points
        fSq = (X(K)-x2).^2 + (Y(K)-y2).^2;
        cosTau = (aSq + thirdPtDistsSq(K) - fSq) ./
(2*sqrt(aSq*thirdPtDistsSq(K)));
        cosTau = min(1,max(-1,cosTau)); %inexact float arithmetic?!
        sinTauSq = 1 - cosTau.^2;
```

```
        b = sqrt( (aSq * thirdPtDistsSq(K) .* sinTauSq) ./ (aSq -
thirdPtDistsSq(K) .* cosTau.^2 + eps) );
        %proper bins for b
        idxs = ceil(b+eps);

        if params.uniformWeights
            weights = 1;
        else
            weights = img(sub2ind(size(img),Y(K),X(K)));
        end
        accumulator = accumarray(idxs, weights, [params.maxMajorAxis 1]);
        %a bit of smoothing and finding the most busy bin
        accumulator = conv(accumulator,H,'same');
        accumulator(1:ceil(sqrt(aSq)*params.minAspectRatio)) = 0;
        [score, idx] = max(accumulator);
        %keeping only the params.numBest best hypothesis (no non-maxima
suppresion)
        if (bestFits(end,end) < score)
            bestFits(end,:) = [x0 y0 sqrt(aSq) idx atand((y1-y2)/(x1-x2))
score];
            if params.numBest>1
                [~,si]=sort(bestFits(:,end),'descend');
                bestFits = bestFits(si,:);
            end
        end
    end
end
```

REFERENCIAS

[1] Ginoris, Y. P., Amaral, A. L., Nicolau, A., Coelho, M. A., & Ferreira, E. C. (2007). Development of an image analysis procedure for identifying protozoa and metazoa typical of activated sludge system. Water research, 41(12), 2581–2589. https://doi.org/10.1016/j.watres.2007.02.006

[2] Castañón, C.A., Fraga, J.S., Fernandez, S., Gruber, A., & Costa, L.D. (2007). Biological shape characterization for automatic image recognition and diagnosis of protozoan parasites of the genus Eimeria. Pattern Recognit., 40, 1899-1910.

[3] C. T. N. Suzuki, J. F. Gomes, A. X. Falcão, S. H. Shimizu and J. P. Papa, "Automated diagnosis of human intestinal parasites using optical microscopy images," 2013 IEEE 10th International Symposium on Biomedical Imaging, San Francisco, CA, 2013, pp. 460-463, doi: 10.1109/ISBI.2013.6556511.

[4] K. H. Ghazali, R. S. Hadi and M. Zeehaida, "Microscopy image processing analaysis for automatic detection of human intestinal parasites ALO and TTO," 2013 International Conference on Electronics, Computer and Computation (ICECCO), Ankara, 2013, pp. 40-43, doi: 10.1109/ICECCO.2013.6718223.

[5] S. S. Savkare and S. P. Narote, "Automated system for malaria parasite identification," 2015 International Conference on Communication, Information & Computing Technology (ICCICT), Mumbai, 2015, pp. 1-4, doi: 10.1109/ICCICT.2015.7045660.

[6] M. A. E. Abdalla, H. Seker and R. Jiang, "Automated identification of chicken eimeria species from microscopic images," 2015 IEEE 15th International Conference on Bioinformatics and Bioengineering (BIBE), Belgrade, 2015, pp. 1-6, doi: 10.1109/BIBE.2015.7367686.

[7] M. A. E. Abdalla, H. Seker and R. Jiang, "Identification of rabbit coccidia by using microscopic images," 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, 2016, pp. 1-4, doi: 10.1109/ICEMIS.2016.7745328.

[8] M. A. E. Abdalla and H. Seker, "Recognition of protozoan parasites from microscopic images: Eimeria species in chickens and rabbits as a case study," 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Seogwipo, 2017, pp. 1517-1520, doi: 10.1109/EMBC.2017.8037124.

[9] Li, C., Wang, K. & Xu, N. A survey for the applications of content-based microscopic image analysis in microorganism classification domains. Artif Intell Rev 51, 577–646 (2019). https://doi.org/10.1007/s10462-017-9572-4

[10] Xiang Li, W. Li, Xiaodong Xu and Wei Hu, "Cell classification using convolutional neural networks in medical hyperspectral imagery," 2017 2nd International Conference on Image, Vision and Computing (ICIVC), Chengdu, 2017, pp. 501-504, doi: 10.1109/ICIVC.2017.7984606.

[11] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

[12] M. F. Wahid, T. Ahmed and M. A. Habib, "Classification of Microscopic Images of Bacteria Using Deep Convolutional Neural Network," 2018 10th International Conference on Electrical and Computer Engineering (ICECE), Dhaka, Bangladesh, 2018, pp. 217-220, doi: 10.1109/ICECE.2018.8636750.

[13] D. F. Monge and C. A. Beltrán, "Classification of Eimeria species from digital micrographies using CNNs," 10th International Conference on Pattern Recognition Systems (ICPRS-2019), Tours, France, 2019, pp. 88-91, doi: 10.1049/cp.2019.0254.

[14] Florin-Christensen, Monica & Schnittger, Leonhard. (2018). Parasitic Protozoa of Farm Animals and Pets. 10.1007/978-3-319-70132-5. https://link--springer--com.us.debiblio.com/book/10.1007/978-3-319-70132-5

[15] Etiología y Patogenia de la coccidiosis aviar https://avicultura.info/etiologia-y-patogenia-de-la-coccidiosis-aviar/

[16] Eimeria https://parasite.org.au/para-site/text/eimeria-text.html

[17] Significado de exicosis https://www.significadode.org/medico/exicosis.htm

[18] Qué son los basófilos y qué significa tenerlos altos o bajos https://blog.saludonnet.com/que-son-los-basofilos-y-que-significa-tenerlos-altos-o-bajos/

[19] Esporozoito https://esacademic.com/dic.nsf/eswiki/447625

[20] Página base de datos http://www.coccidia.icb.usp.br/imagedb/Download.html

[21] Página del ciclo vital microbios https://microbewiki.kenyon.edu/index.php/The_Monoxenous_Life_Cycle_Of_Eimeria#:~:text=The%20life%20cycle%20of%20Eimeria,that%20Eimeria%20use%20as%20metabolism.

[22] Imagen ciclo eimeria https://www.veterinariadigital.com/en/articulos/avian-coccidiosis-southeast-asia-and-natural-control-methods/

[23] Evolution of Deep learning related publications image https://www.researchgate.net/figure/Evolution-of-deep-learning-related-publications-a-citations-listed-in-Google-scholar_fig1_341576780

[24] Hoeser, Thorsten & Kuenzer, Claudia. (2020). Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends. Remote Sensing. 12. 10.3390/rs12101667.

[25] Wang, Haohan & Raj, Bhiksha. (2017). On the Origin of Deep Learning.

[26] Serokell. Artificial Intelligence vs. Machine Learning vs. Deep Learning: What's the Difference https://medium.com/ai-in-plain-english/artificial-intelligence-vs-machine-learning-vs-deep-learning-whats-the-difference-dccce18efe7f

[27] Michael Copeland. What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning? https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-

learning-deep-learning-ai/

[28] Connor Shorten. Popular Deep Learning Applications https://towardsdatascience.com/what-can-deep-learning-bring-to-your-app-fb1a6be63801

[29] Rebala, Gopinath & Ravi, Ajay & Churiwala, Sanjay. (2019). An Introduction to Machine Learning. 10.1007/978-3-030-15729-6.

[30] Kim, Phil. (2017). MATLAB Deep Learning. 10.1007/978-1-4842-2845-6.

[31] Taiwo Oladipupo Ayodele (February 1st 2010). Machine Learning Overview, New Advances in Machine Learning, Yagang Zhang, IntechOpen, DOI: 10.5772/9374. Available from: https://www.intechopen.com/books/new-advances-in-machine-learning/machine-learning-overview

[32] Introduction to Machine Learning and ML.NET Part 1 https://social.technet.microsoft.com/wiki/contents/articles/53298.introduction-to-machine-learning-and-ml-net-part-1.aspx

[33] Understanding supervised, unsupervised, and reinforcement learning https://mc.ai/understanding-supervised-unsupervised-and-reinforcement-learning/

[34] Ana Mezic. Why Unsupervised Machine Learning is the Future of Cybersecurity https://www.technative.io/why-unsupervised-machine-learning-is-the-future-of-cybersecurity/

[35] Semi-Supervised http://primo.ai/index.php?title=Semi-Supervised

[36] Dr. Michael J. Garbade. Understanding K-means Clustering in Machine Learning https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[37] What is reinforcement learning? The Complete Guide https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

[38] Reinforcement Learning — Aprendizaje por refuerzo https://medium.com/@bootcampai/reinforcement-learning-aprendizaje-por-refuerzo-c34bb085bb5

[39] Applications of Reinforcement Learning in Real World https://towardsdatascience.com/applications-of-reinforcement-learning-in-real-world-1a94955bcd12

[40] The Limitations of Machine Learning https://towardsdatascience.com/the-limitations-of-machine-learning-a00e0c3040c6

[41] Kisuk Lee, Nicholas L. Turner, Thomas Macrina, Jingpeng Wu, Ran Lu, H. Sebastian Seung. Convolutional nets for reconstructing neural circuits from brain images acquired by serial section electron microscopy. CoRR abs/1904.12966 (2019)

[42] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539

[43] Guest, Dan & Cranmer, Kyle & Whiteson, Daniel. (2018). Deep Learning and Its Application to LHC Physics. Annual Review of Nuclear and Particle Science. 68. 161-181. 10.1146/annurev-nucl-101917-021019.

[44] Zhou, Jian & Troyanskaya, Olga. (2015). Predicting effects of noncoding variants with deep learning-based sequence model. Nature methods. 12. 10.1038/nmeth.3547.

[45] Y. Bengio, A. Courville and P. Vincent, "Representation Learning: A Review and New Perspectives," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, Aug. 2013, doi: 10.1109/TPAMI.2013.50.

[46] X. Chen and X. Lin, "Big Data Deep Learning: Challenges and Perspectives," in IEEE Access, vol. 2, pp. 514-525, 2014, doi: 10.1109/ACCESS.2014.2325029.

[47] Michael Nielsen. Neural Networks and Deep Learning http://neuralnetworksanddeeplearning.com/chap2.html

[48] Anas Al-Masri. How Does Back-Propagation in Artificial Neural Networks Work? https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7

[49] Kadir Aydin. Feed Forward Neural Network https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[50] Edvin Beqari. A Very Basic Introduction to Feed-Forward Neural Networks https://dzone.com/articles/the-very-basic-introduction-to-feed-forward-neural

[51] SauceCat. Backward Propagation for Feed Forward Networks https://towardsdatascience.com/backward-propagation-for-feed-forward-networks-afdf9d038d21

[52] Hinton, G. E., Osindero, S. & Teh, Y.-W. A fast learning algorithm for deep belief nets. Neural Comp. 18, 1527–1554 (2006).

[53] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning . MIT Press.

[54] Schwartz, O., Simoncelli, E. Natural signal statistics and sensory gain control. Nat Neurosci 4, 819–825 (2001). https://doi.org/10.1038/90526

[55] Pettigrew, J. D., Sanderson, K. J., & Levick, W. R. (1986). Visual neuroscience. Cambridge [Cambridgeshire: Cambridge University Press

[56] Syed Anwar. Convolutional Layer. https://www.researchgate.net/figure/Convolutional-Layer-76-31-Convolutional-Neural-Network-Convolutional-neural-networks_fig2_319535615

[57] Convolutional Neural Networks (CNNs / ConvNets) https://cs231n.github.io/convolutional-networks/

[58] Sumit Saha A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[59] D. Hubel and T. Wiesel, ‘‘Receptive fields and functional architecture of monkey striate cortex,’’ J. Physiol., vol. 195, pp. 215–243, Mar. 1968.

[60] Richmond Alake. Understand Local Receptive Fields In Convolutional Neural Networks. https://towardsdatascience.com/understand-local-receptive-fields-in-convolutional-neural-networks-

f26d700be16c#:~:text=The%20local%20receptive%20field%20is,neural%20network%20for%20character%20recognition.

[61] Feature extraction for machine learning and deep learning https://www.mathworks.com/discovery/feature-extraction.html

[62] Mohamed Elgendy. Deep Learning for Vision Systems https://freecontent.manning.com/the-computer-vision-pipeline-part-3-image-preprocessing/

[63] imbothat https://es.mathworks.com/help/images/ref/imbothat.html

[64] imclose https://es.mathworks.com/help/images/ref/imclose.html

[65] Detección de bordes https://es.mathworks.com/help/images/edge-detection.html

[66] Martin Simonovsky. Ellipse Detection Using 1D Hough Transform https://es.mathworks.com/matlabcentral/fileexchange/33970-ellipse-detection-using-1d-hough-transform?s_tid=srchtitle

[67] Gangyi Wang, Guanghui Ren, Zhilu Wu, Yaqin Zhao, Lihui Jiang, "A Fast and Robust Ellipse-Detection Method Based on Sorted Merging", The Scientific World Journal, vol. 2014, Article ID 481312, 15 pages, 2014. https://doi.org/10.1155/2014/481312

[68] EXCENTRICIDAD DE UNA ELIPSE. https://www.universoformulas.com/matematicas/geometria/excentricidad-elipse/

[69] Juan Martín Peralta. Capítulo 2. PROPIEDADES DE UNA PARTÍCULA. https://docplayer.es/9208135-Capitulo-2-propiedades-de-una-particula.html

[70] imageDatastore https://www.mathworks.com/help/matlab/ref/matlab.io.datastore.imagedatastore.html

[71] Li, Hongzhi & Ellis, Joseph & Zhang, Lei & Chang, S.. (2018). PatternNet: Visual Pattern Mining with Deep Neural Network. 291-299. 10.1145/3206025.3206039.

[72] Tarang Shah. About Train, Validation and Test Sets in Machine Learning. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[73] Prabhu .Understanding Hyperparameters and its Optimisation techniques https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568#:~:text=What%20are%20Hyperparameters%3F,Model%20parameters%20vs%20Hyperparameters

[74] Hidden Layer. https://deepai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning#:~:text=In%20neural%20networks%2C%20a%20hidden,inputs%20entered%20into%20the%20network.

[75] How to choose the number of hidden layers and nodes in a feedforward neural network? https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

[76] Heaton, Jeffrey. (2012). Introduction to the math of neural networks. Heaton Research, New York.

[77] Heaton, Jeffrey. (2008). Introduction to Neural Networks for Java, 2nd Edition (2nd. ed.). Heaton Research, Inc.

[78] Viraj Kulkarni. Cross-Entropy for Dummies. https://towardsdatascience.com/cross-entropy-for-dummies-5189303c7735

[79] Hieu Pham. When should you use cross entropy loss and why? https://www.quora.com/When-should-you-use-cross-entropy-loss-and-why

[80] Mark Hudson Beale, Martin T. Hagan, Howard B. Demuth. (2010). Neural Network Toolbox™ 7 User's Guide.

[81] EduPristine. Problems of Small Data and How to Handle Them. https://www.edupristine.com/blog/managing-small-data

[82] Gabriel Ha. Deep Learning with MATLAB: Using Feature Extraction with Neural Networks in MATLAB. https://es.mathworks.com/videos/using-feature-extraction-with-neural-networks-in-matlab-1492009542601.html

[83] Perez, Luis & Wang, Jason. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning.

[84] Sunita Nayak. Understanding AlexNet. https://www.learnopencv.com/understanding-alexnet/

[85] Nawaz W., Ahmed S., Tahir A., Khan H.A. (2018) Classification Of Breast Cancer Histology Images Using ALEXNET. In: Campilho A., Karray F., ter Haar Romeny B. (eds) Image Analysis and Recognition. ICIAR 2018. Lecture Notes in Computer Science, vol 10882. Springer, Cham. https://doi.org/10.1007/978-3-319-93000-8_99

[86] Jason Brownlee. (2019). Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python

[87] Leo Pauly. ReLU activation function. https://www.researchgate.net/figure/ReLU-activation-function_fig3_319235847

[88] Raul E. Lopez Briega. Redes neuronales convolucionales con TensorFlow https://relopezbriega.github.io/blog/2016/08/02/redes-neuronales-convolucionales-con-tensorflow/

[89] nprtool https://www.mathworks.com/help/deeplearning/ref/nprtool.html;jsessionid=e2ddd47d9eaea27db36e46485974

[90] Juan Ignacio Barrios Arce. La matriz de confusión y sus métricas. https://www.juanbarrios.com/matriz-de-confusion-y-sus-metricas/

[91] Curva ROC. Freddy Abad L. https://medium.com/@freddy.abadl/curva-roc-d8c638894f49

[92] Glosario sobre aprendizaje automático. https://developers.google.com/machine-learning/glossary?hl=es-419