Trabajo Fin de Máster

Máster en Automática e Informática Industrial

Diseño, desarrollo y evaluación de algoritmos basados en aprendizaje profundo para automatización de experimentos *Lifespan* con *C. elegans*.

Antonio García Garví

Tutor: Antonio José Sánchez Salmerón

Septiembre 2020

Agradecimientos

A mis padres y a mi hermana, por su ayuda y apoyo.

A mi tutor, Antonio José Sánchez Salmerón, por toda la confianza que me ha dado y por estar siempre disponible para resolver las dudas y los problemas que han surgido en este curso atípico.

A los compañeros del ai2, sin los cuales no se podría haber realizado el proyecto: Duncan, Pablo, Joan, Jordi.

Resumen

En los últimos años, los nematodos *C. elegans* cultivados en placas de Petri se han utilizado en muchas investigaciones relacionadas con el envejecimiento. El desarrollo de nuevas herramientas para automatizar los experimentos de *lifespan* permite realizar más ensayos en menos tiempo y evitar errores humanos, obteniendo resultados más precisos.

El objetivo de este TFM consiste en diseñar y desarrollar métodos para abordar este problema utilizando técnicas de aprendizaje profundo. Posteriormente, se evaluarán los resultados comparando los resultados con los obtenidos empleando técnicas tradicionales de visión por computador.

Inicialmente, el trabajo se centrará en la creación y edición de forma supervisada de un conjunto de imágenes bien etiquetadas. Posteriormente se diseñarán distintas arquitecturas de redes neuronales y se optimizará cada una de ellas sobre el espacio de hiperparámetros utilizando Python y Pytorch. Finalmente, se evaluarán las distintas arquitecturas propuestas, utilizando como criterios de optimización tanto las tasas de aciertos como los costes temporales de computación.

Palabras clave: lifespan; edición y creación de conjunto de datos; aprendizaje profundo; redes neuronales convolucionales

Resum

En els últims anys, els nematodes *C. elegans* conreats en plaques de Petri s'han utilitzat en moltes recerques relacionades amb l'envelliment. El desenvolupament de noves eines per a automatitzar els experiments de lifespan permet realitzar més assajos en menys temps i evitar errors humans, obtenint resultats més precisos.

L'objectiu d'aquest TFM consisteix a dissenyar i desenvolupar mètodes per a abordar aquest problema utilitzant tècniques d'aprenentatge profund. Posteriorment, s'avaluaran els resultats comparant els resultats amb els obtinguts emprant tècniques tradicionals de visió per computador.

Inicialment, el treball se centrarà en la creació i edició de forma supervisada d'un conjunt d'imatges ben etiquetades. Posteriorment es dissenyaran diferents arquitectures de xarxes neuronals i s'optimitzarà cadascuna d'elles sobre l'espai de hiperparámetros utilitzant Python i Pytorch. Finalment, s'avaluaran les diferents arquitectures proposades, utilitzant com a criteris d'optimització tant les taxes d'encerts com els costos temporals de computació.

Paraules clau: lifespan; edició i creació de conjunt de dades; aprenentatge profund, xarxes neuronals convolucionals

Abstract

In recent years, C. elegans nematodes grown in Petri dishes have been used in many investigations related to aging. The development of new tools to automate lifespan experiments allows more tests to be carried out in less time and to avoid human error, obtaining more accurate results.

The objective of this TFM is to design and develop methods to address this problem using deep learning techniques. Subsequently, the results will be evaluated by comparing the results with those obtained using traditional computer vision techniques.

Initially, work will focus on supervised creation and editing of a set of well-labeled images. Subsequently, different neural network architectures will be designed and each one will be optimized on the hyperparameter space using Python and Pytorch. Finally, the different proposed architectures will be evaluated, using both the accuracies and the temporary computing costs as optimization criteria.

Keywords: lifespan; edition and creation of a data set; Deep learning; convolutional neural networks

Contenido

Índice de figuras

Índice de tablas

Índice de ecuaciones

1 Introducción

En los últimos años, se han producido grandes avances en campos como la medicina, la biotecnología y la nutrición. Esto ha propiciado un aumento de la esperanza de vida de las personas, sin embargo, esto no se traduce en un aumento de la calidad de vida.

Con el envejecimiento, aparecen enfermedades neurodegenerativas como el Alzheimer, que constituyen un gran problema social. Por este motivo, es primordial la búsqueda de nuevos medicamentos, componentes terapéuticos y productos alimenticios que ayuden a hacer frente a estas enfermedades y a mejorar la calidad de vida.

En la investigación de enfermedades neurodegenerativas y del envejecimiento, destaca el nematodo *Caenorhabditis elegans* (*C. elegans)* como modelo de estudio debido a sus características, que lo hacen propicio para estos estudios.

El *C. elegans* es un nematodo que destaca por su tamaño (aproximadamente 1 mm de longitud) y su cuerpo transparente, lo cual permite cultivarlos y manipularlos en placas de Petri estándar de forma económica y observarlos mediante microscopio. Presenta un sistema nervioso simple y su genoma se encuentra completamente secuenciado desde el año 1998. Aproximadamente un 38% de sus genes se corresponden con su homólogo en el ser humano, lo que permite estudiar con ellos enfermedades que afectan a los humanos.

Tienen un ciclo de vida corto, comprendido entre 2 y 3 semanas para la mayoría de las cepas, pudiendo ser mayor para algunas variedades. Esto permite realizar ensayos de corta duración con resultados inmediatos. En la figura 1B se puede ver al nematodo en distintas fases: embrión, larva y adulto.

Se alimentan principalmente de la bacteria *Escherichia coli* (*E.coli*) que se muestra en la figura 1A, lo que facilita su alimentación y mantenimiento.

Además, la mayoría son hermafroditas, de forma que pueden autofecundarse .También existen machos, pero en una proporción muy pequeña. En la figura 1C se puede ver un *C. elegans* adulto hermafrodita.

Es posible monitorizar la actividad de determinadas neuronas introduciendo proteínas como la GFP (*Green Fluorescent Protein*). En la figura 1D se puede ver el sistema nervioso de un *C. elegans* marcado con GFP.

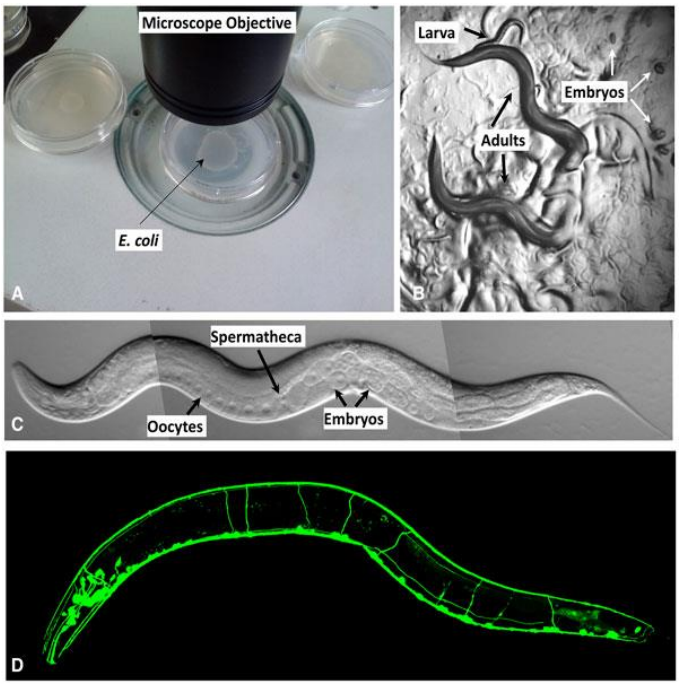

Fig. 1 Distintas imágenes de C. elegans (Corsi et al., 2015)

A pesar de todas las ventajas que ofrece este nematodo para la investigación, también presenta algunos inconvenientes, ya que su manipulación y monitorización requiere de personal técnico altamente cualificado que dedique una gran cantidad de tiempo y esfuerzo diariamente.

No obstante, en la mayoría de los laboratorios de investigación se realizan estas tareas de forma manual. Estas tareas de manipulación y monitorización son muy laboriosas. Por este motivo, es de gran importancia el desarrollo de equipos y herramientas que permitan la automatización de estas tareas, reduciendo los tiempos de experimentación.

Fig. 2 Investigación con C. elegans en laboratorios (imagen Worcester Polytechnic Institute)

La automatización, además de la reducción de tiempos y facilitar las tareas del investigador, permite obtener medidas más precisas y evita posibles errores humanos que se puedan producir en el desarrollo de estas tareas.

2 Justificación y antecedentes

El Instituto Universitario de Automática e Informática Industrial (ai2) de la Universidad Politécnica de Valencia (UPV) trabaja en el desarrollo de sistemas automatizados para la monitorización avanzada del comportamiento de *C. elegans* cultivados en placas de Petri estándar. Estos sistemas, basados en técnicas de seguimiento de posturas y visión activa, permiten analizar la función cognitiva y el envejecimiento.

La monitorización de *C. elegans* cultivados en placas de Petri estándares se trata de una tarea compleja debido a: (1) la gran variedad de formas o poses que pueden adoptar estos nematodos; (2) el problema de contaminación y condensación, que hace necesario el uso de técnicas de iluminación especiales y (3) el problema de agregación de varios *C. elegans*, que requiere técnicas de segmentación específicas.

Para resolver este problema, el grupo de investigación del instituto ai2 ha desarrollado un sistema de visión de artificial con iluminación inteligente (Puchalt et al., 2019), que permite la adquisición de imágenes reduciendo el efecto de los problemas comentados, y además manteniendo los rangos de intensidad de los *C. elegans* y del fondo prácticamente constantes, facilitando significativamente la segmentación.

Dentro de los ensayos realizados con *C. elegans* para estudiar el envejecimiento, uno de los más destacados es el de *Lifespan* (Tissenbaum, 2015). Un experimento de *Lifespan* consiste en realizar el conteo de nematodos vivos en la placa de ensayo de forma periódica (Amrit et al., 2014). El experimento comienza a partir del inicio de la edad adulta y finaliza cuando muere el último nematodo. Utilizando este conteo se elaboran curvas de supervivencia como se ve en la figura 3, donde se representa el porcentaje de supervivencia de la población en cada día.

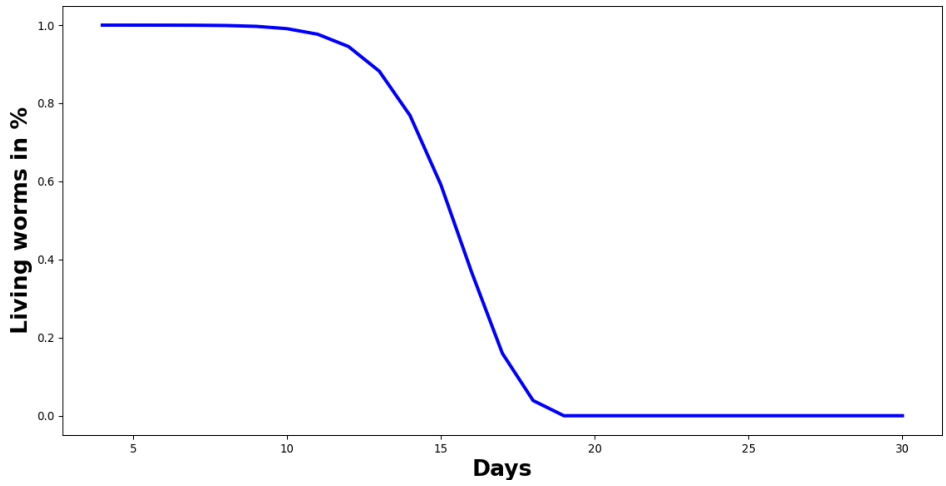

Fig. 3 Curva típica de lifespan

El grupo de investigación del ai2 ha desarrollado un algoritmo de segmentación y conteo automático de *C. elegans* (Puchalt et al., 2020) a partir de las imágenes capturadas por el sistema de visión con iluminación inteligente. Este sistema captura secuencias de 30 imágenes por placa cada día, que son almacenadas para posteriormente ser procesadas

por el algoritmo de conteo automático. Este algoritmo presenta un error de 4,62 ± 2,01%, que se consigue reducir a 2,24 ± 0,55% utilizando una técnica de postprocesado.

En los últimos años, los grandes avances en el campo de la inteligencia artificial y en las prestaciones de los computadores han generalizado el uso de estas técnicas en el ámbito de la visión por computador, permitiendo resolver problemas de clasificación de objetos en imágenes que hasta hace unos años parecían inabordables. Por este motivo, se ha investigado el uso de las redes neuronales artificiales para resolver problemas de clasificación de *C. elegans.* En un trabajo anterior, se analizó el uso de las redes neuronales convolucionales (García Garví, 2019) para la clasificación de *C. elegans* como vivos o muertos.

Fig. 4 Esquema clasificador C. elegans vivos/muertos

Este clasificador obtiene altas tasas de acierto (en torno a 95-97%) y permite analizar si un *C. elegans* en concreto está vivo o muerto. Sin embargo, sigue siendo necesario utilizar una etapa de segmentación para la identificación de los *C. elegans* para poder emplearse, lo cual puede hacer que se pierda precisión dada la dificultad de la segmentación (oclusiones, contaminación, variedad de posturas, etc.) y que aumente el tiempo de computación.

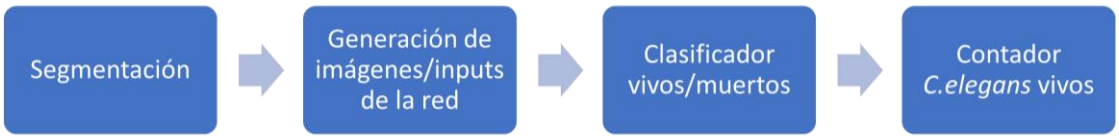

Fig. 5 Etapas sistema con clasificador vivos/muertos

Por estas razones, el siguiente objetivo es el de utilizar técnicas de aprendizaje profundo para la automatización completa de los experimentos de *Lifespan*, en las que la entrada del sistema sea el conjunto de imágenes de la placa de Petri completa en lugar de imágenes individuales de *C. elegans.* De esta manera, se evitarían los posibles errores acumulados de la distintas etapas y reduciría el tiempo de computación.

3 Objetivos

El objetivo principal de este proyecto es el de diseñar, desarrollar y evaluar técnicas de aprendizaje profundo para la automatización de experimentos *Lifespan* con nematodos *C. elegans.*

En concreto, se pretende obtener automáticamente el conteo de *C. elegans* vivos presentes en una placa de Petri en cada uno de los días del experimento, para obtener finalmente la curva de *Lifespan* (figura 6).

Para ello se deben conseguir los siguientes objetivos específicos:

- Creación, etiquetado y edición del conjunto de imágenes de entrenamiento.
- Planteamiento de arquitecturas de redes neuronales.
- Implementación en el *framework* de *Deep learning Pytorch*.
- Evaluación de resultados y propuestas de mejora.
- Aumento de datos.

Los resultados obtenidos se deben comparar con la solución utilizando técnicas tradicionales de visión por computador.

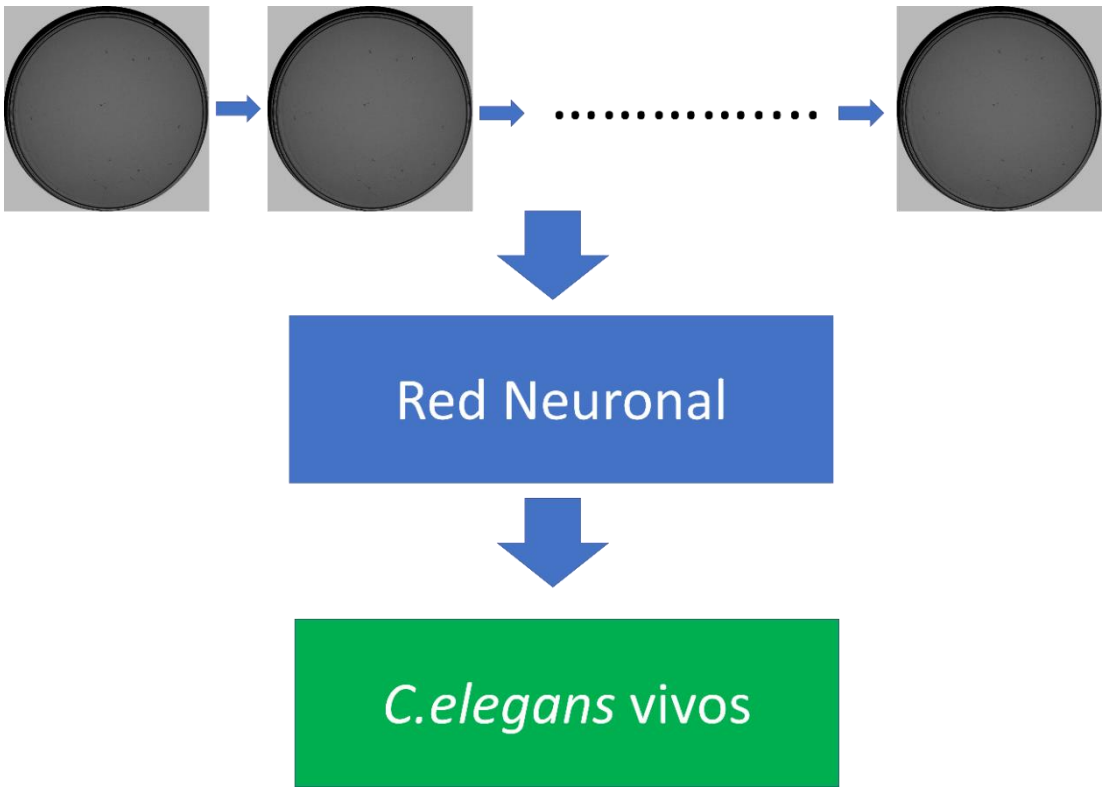

Fig. 6 Esquema objetivo del proyecto

4 Estado del arte

4.1 Antecedentes de visión artificial

El aumento de la competitividad es un requerimiento primordial en la situación económica actual. Un factor clave para mejorar la competitividad es aumentar la productividad incorporando nuevos sistemas de inspección que permitan la automatización de diferentes procesos. Los sistemas de inspección visual automática, basados en visión por computador, han demostrado ser una herramienta fundamental para mejorar los procesos. Estos sistemas de visión permiten la inspección continua, evitando fatigas y distracciones, y facilitando la cuantificación de las variables de calidad en prácticamente el 100% de la producción. Esto se traduce, no sólo en una mejora de la calidad final de los productos, sino también en un ahorro en términos económicos y medioambientales.

Durante las últimas décadas se han podido resolver multitud de aplicaciones mediante la implantación de sistemas de inspección 2D en la industria utilizando técnicas de visión por computador tradicionales. El principal problema a resolver en estos sistemas ha sido seleccionar un conjunto mínimo de características discriminante para clasificar los objetos de interés, además de la gran variabilidad de las imágenes que puede dificultar la segmentación de estos objetos de interés. Para resolver este problema se han propuesto algunas técnicas robustas de segmentación que intentan absorber dicha variabilidad (Benlloch et al., 1995; A. J. Sánchez et al., 2008). Además, también se han propuesto infinidad de soluciones de seguimiento de objetos, como por ejemplo (A. Sánchez & Ramos, 2000).

Por otro lado, los enormes avances tecnológicos producidos en las cámaras lineales, los escáneres 3D (Sanchez, A. J., & Marchant, J. A., 2000; Ricolfe-Viala & Sanchez-Salmeron, 2011; Eugenio Ivorra et al., 2014; E. Ivorra et al., 2015; Verdú et al., 2015a) y los sensores hiperespectrales (Grau et al., 2011; Verdú et al., 2015b; Eugenio Ivorra et al., 2016) están permitiendo resolver algunas aplicaciones complejas en el sector industrial que hace unos años eran impensables. Estas nuevas tecnologías aplicadas sobre todo a los sectores de salud, biotecnología, agroalimentario y cosmético, se están empezando a introducir en la industria.

Además, los últimos avances producidos en la aplicación de técnicas de detección y seguimiento de objetos deformables (E. Berti et al., 2012; Martínez Bertí et al., 2012; Martinez-Berti et al., 2016; E. M. Berti et al., 2017; E. Berti et al., 2017; Martinez-Berti et al., 2017) basadas en redes neuronales convolucionales; están permitiendo empezar a automatizar algunas aplicaciones de detección complejas.

La automatización de un experimento de lifespan es un problema complejo debido a la gran variabilidad de formas que pueden adoptar los *C. elegans*, y a la posibilidad de que estos se agreguen entre ellos o queden ocultos. En este trabajo se plantea el análisis de una nueva técnica de imagen basadas en redes neuronales recurrentes de cara a intentar automatizar esta aplicación.

4.2 Inteligencia artificial, Machine Learning, Deep Learning

Los conceptos de inteligencia artificial, *machine learning* y *Deep learning* están relacionados, pero son diferentes. A pesar de esto, se suelen emplear de forma incorrecta en los medios de comunicación, por ello es necesario comprender que son cada uno. La relación entre estos conceptos se puede visualizar como círculos concéntricos (figura 7) de forma que el *machine learning* es un campo de la inteligencia artificial y el *deep learning* es un campo del *machine learning*.

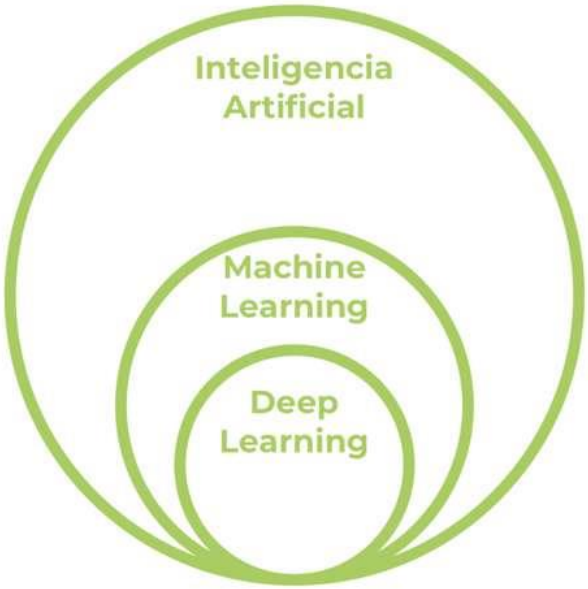

Fig. 7 Conceptos IA, Machine Learning y Deep Learning («¿Qué Es La Inteligencia Artificial?», 2019)

En la literatura se encuentran diferentes formas de definir inteligencia artificial, la mayoría coinciden en que se trata de una disciplina del área de las ciencias de la computación que trata de conseguir que las máquinas sean capaces de realizar o emular tareas "inteligentes", propias de los seres humanos.

Dentro de la inteligencia artificial se pueden encontrar distintos subcampos como pueden ser la robótica, la visión por computador o el análisis de texto y voz. En estas áreas se pueden resolver problemas utilizando algoritmos clásicos en los que el programador diseña e implementa la secuencia de tareas a ejecutar o algoritmos en los que la máquina "aprende" por sí sola a realizar las tareas. De aquí surge el concepto de aprendizaje automático o *machine learning*.

Machine learning o aprendizaje automático es la rama del campo de la inteligencia artificial que busca dotar a las máquinas de la capacidad de aprendizaje, generalización de un conocimiento a partir de un conjunto de experiencias.

En *machine learning* existen tres grandes grupos de paradigmas de aprendizaje: supervisado, no supervisado y reforzado.

En el aprendizaje supervisado se proporciona al algoritmo un conjunto de datos junto con su clasificación o etiqueta. A partir de esto, el algoritmo debe ser capaz de encontrar las relaciones que le permitan generalizar y obtener la solución al problema con nuevos datos de entrada. Este paradigma es el que más se ha estudiado y aplicado históricamente. Algunos ejemplos son las redes neuronales, los árboles de decisión o las SVM.

A diferencia del anterior, el aprendizaje no supervisado no recibe como entrada la clasificación o etiqueta esperada, sino simplemente los datos de entrada. La ventaja de este paradigma respecto al supervisado es que es menos costoso conseguir el *dataset*, ya que se evita la etapa de etiquetado. Un ejemplo de este paradigma es el *clustering*.

En el aprendizaje reforzado se define una tarea a resolver por un agente inteligente dentro de un *framework* o espacio de simulación. Si el algoritmo ejecuta correctamente la tarea será recompensado, mientras que en caso contrario será penalizado. De esta forma debe ser capaz de encontrar la mejor estrategia para resolver la tarea obteniendo la mayor recompensa.

Dentro del *machine learning* existen distintas técnicas como:

- Árboles de decisión
- Modelos de regresión
- Modelos de clasificación
- Redes neuronales

De estas técnicas, en los últimos años han destacado por encima del resto las redes neuronales artificiales. Estas son capaces de aprender de forma jerarquizada, es decir de conceptos más sencillos a más abstractos.

Las redes neuronales artificiales son modelos computacionales inspirados en las redes neuronales biológicas (figura 8), de ahí su nombre. Su unidad básica de procesamiento son las neuronas, que al igual que las neuronas biológicas presentan unas conexiones de entrada a través de las cuales reciben unos valores. Cada conexión de entrada tiene asociado un peso o valor que pondera su efecto en la salida.

Con estos valores, la neurona realiza una suma ponderada para obtener un valor de salida.

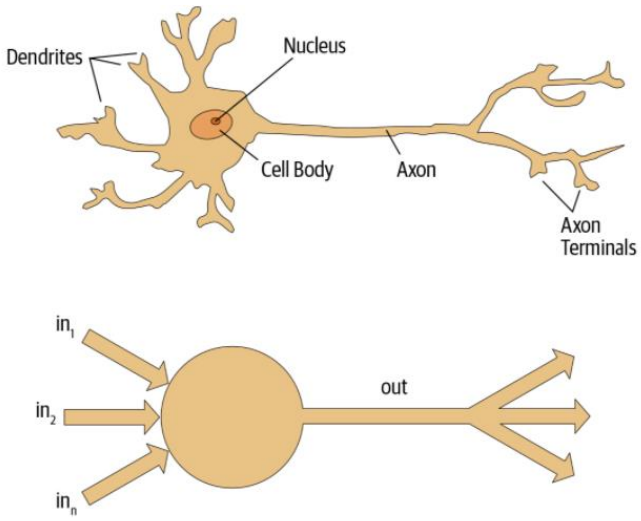

Fig. 8 Analogía entre redes neuronales biológicas y artificiales (Jeremy Howard & Sylvain Gugger, 2020)

Estas neuronas están conectadas unas con otras de forma estructurada en capas. Según la posición ocupada en la red reciben distintos nombres. En primer lugar, se encuentra la capa de entrada, que es la encargada de recibir los datos. En medio se encuentran las capas ocultas, que son las encargadas de realizar una serie de transformaciones de los datos que permitan obtener la salida deseada. Por último, las capas de salida son las que devuelven las predicciones realizadas. En la siguiente figura se puede observar dicha estructura:

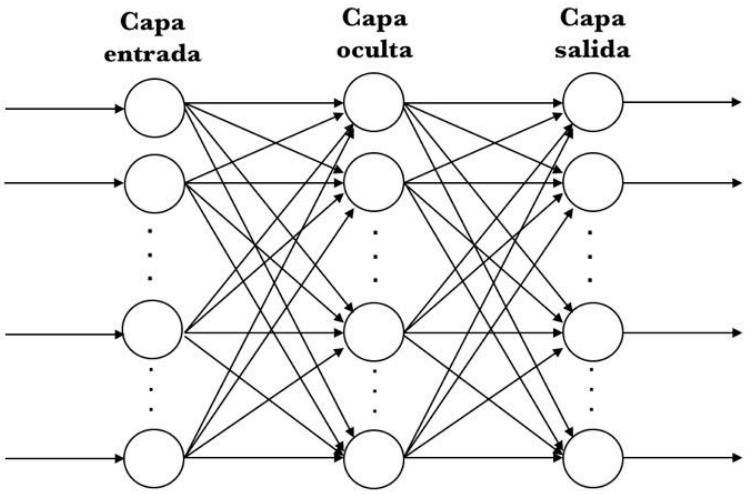

Fig. 9 Estructura en capas de las redes neuronales («¿Qué Es La Inteligencia Artificial?», 2019)

El número de capas empleado es cada vez mayor, de ahí surge el concepto de *Deep learning* o aprendizaje profundo. Este aumento de capas permite obtener patrones más abstractos.

Las redes neuronales existen desde el año 1958, cuando Frank Rosenblatt creó el perceptrón. Sin embargo, no ha sido hasta la última década cuando se ha producido la revolución del *Deep Learning*. Esto se debe a múltiples factores. Por un lado, se encuentra la mejora del hardware de adquisición (cámaras con mayor precisión y menor

coste) y almacenamiento de datos que permiten disponer del gran volumen de datos necesario para entrenar los modelos.

Por otro lado, destaca el avance de la supercomputación («Supercomputación, Corazón de Deep Learning», 2018). Hasta el año 2012, el incremento del poder de computación estaba ligado a las mejoras en la CPU. Ese año fue cuando el equipo de Alex Krizhevsky comenzó a emplear las GPUs para el entrenamiento de su red AlexNet en la competición ImageNet, reduciendo significativamente el tiempo de cómputo necesario.

Esto se debe a que las redes neuronales necesitan realizar cálculos con matrices y las GPU, desarrolladas originalmente para los gráficos 3D de los videojuegos, están especializadas en realizar este tipo de cálculos. Las CPU ejecutan instrucciones de forma secuencial a gran velocidad, están formadas por pocos núcleos y poseen un conjunto de instrucciones de carácter general. Por el contrario, las GPUs ejecutan instrucciones en paralelo y poseen un mayor número de núcleos más específicos.

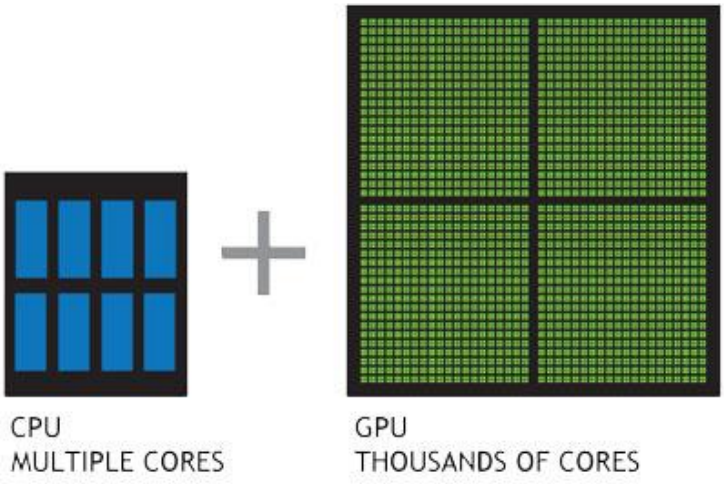

Fig. 10 Comparativa CPU vs GPU (¿GPU vs. CPU? ¿Qué es la computación por GPU? | NVIDIA*, s. f.)*

Desde entonces se ha continuado mejorando la capacidad de cómputo utilizando interconexión de servidores con múltiples GPU, y en los últimos años se ha comenzado a desarrollar chips especializados en el procesamiento de algoritmos de aprendizaje profundo como las TPU (Tensor Processing Unit).

Por último, dentro de los factores que han influido en el avance en los últimos años en la inteligencia artificial destaca la cultura del código abierto, el desarrollo de *frameworks* de *Deep Learning* (como Pytorch, Tensorflow o Keras) y los sistemas *cloud* que han hecho mucho más accesible estas tecnologías, aumentando de esta forma las investigaciones y reduciendo los tiempos.

4.3 Deep Learning para experimentos con *C. elegans*

En los últimos años ha aumentado el uso de técnicas de aprendizaje profundo frente al uso de técnicas tradicionales de visión por computador en el campo del análisis de imágenes biológicas. Esto es debido a los grandes avances en el campo de las redes neuronales y la mejora de los dispositivos de adquisición y almacenamiento de datos, que permite obtener grandes conjuntos de datos para el entrenamiento.

En este proyecto se ha realizado una búsqueda de proyectos e investigaciones que utilizan redes neuronales para automatizar experimentos de *C. elegans.* Algunos de ellos son los siguientes:

- En (Silva et al., 2003) se utilizan redes neuronales para la segmentación (*competitive neural network*) e identificación (RBF) de nematodos.
- En (Feng Ning et al., 2005) se realiza el fenotipado automático de embriones de *C. elegans* en desarrollo a partir de videos utilizando redes neuronales convolucionales para la detección, identificación y medición de núcleos celulares, citoplasmas y las paredes celulares.
- En (Kong et al., s. f.) se desarrolla un sistema de detección y clasificación de especies de *C. elegans*, larvas y huevos utilizando una arquitectura de red neuronal ImageNet.
- En (Li et al., 2017) se utilizan redes neuronales recurrentes (LSTM) para generar trayectorias de *C. elegans*. Partiendo de la pose actual son capaces de predecir las siguientes con este modelo.
- En (Wang et al., 2018) se utilizan técnicas de aprendizaje profundo reforzado para el modelado del movimiento celular en la etapa temprana de la embriogénesis de los *C. elegans.*
- *WorMachine* (Hakim et al., 2018) es un software desarrollado en Matlab que permite analizar imágenes capturadas en experimentos con *C. elegans*. En este proyecto se emplean técnicas de *machine learning* para clasificación de características binarias y obtención de parámetros relacionados con distintos fenotipos.
- En (Javer et al., 2019) se plantea el uso de un clasificador que mediante redes neuronales convolucionales permite distinguir entre cepas de *C. elegans* basándose en posturas.
- En (Lin et al., 2020) se emplean distintos modelos de redes neuronales convolucionales para medir la edad fisiológica de los *C. elegans*, que permiten realizar estudios de cribado genético y la experimentación con fármacos antienvejecimiento.

En esta búsqueda no se ha encontrado ningún proyecto que haya resuelto el problema que se trata en este TFM.

5 Desarrollo

5.1 Hardware y software empleado

5.1.1 Hardware

El proyecto se ha desarrollado utilizando dos ordenadores. Por un lado, se ha utilizado el ordenador de sobremesa del laboratorio del instituto ai2 para el entrenamiento de las redes neuronales. Dicho equipo cuenta con las siguientes características:

- Procesador @Intel® Core™ i7-7700K CPU @4,20GHz x 8.
- Tiene 15,6 GiB de memoria RAM
- Unidad de Procesamiento Gráfico (GPU) GeForce GTX 1070 Ti/PCIe/SSE2 9

Fig. 11 Características principales del equipo disponible en el laboratorio

Por otro lado, se ha utilizado un ordenador portátil del alumno para la parte de generación de imágenes artificiales simuladas, ya que se necesitaba emplear software que solo está disponible en Windows, además de la conexión en remoto con el ordenador del laboratorio, la investigación y la elaboración de la memoria.

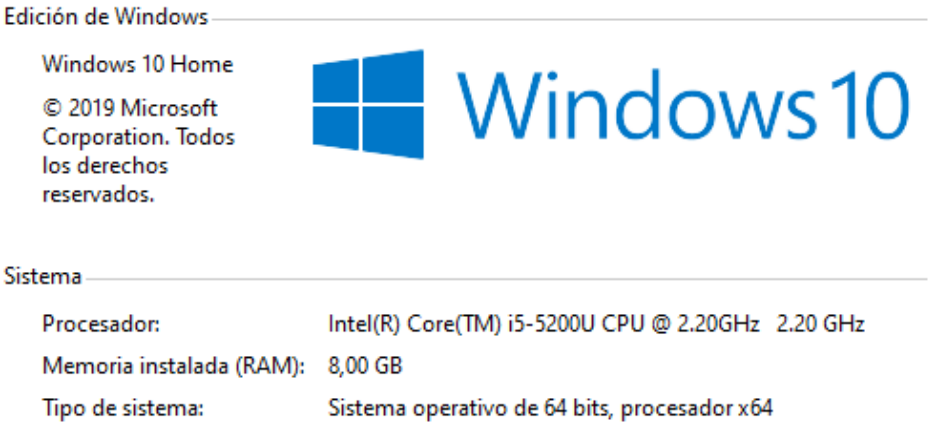

Fig. 12 Características ordenador portátil

5.1.2 Software

El equipo del laboratorio del ai2 tiene instalado el sistema operativo libre Ubuntu 16.04 LTS. El lenguaje de programación elegido es Python 3.7, que es de código abierto. Destaca por ser un lenguaje interpretado multiplataforma. Es multiparadigma, soportando orientación a objetos, programación imperativa y programación funcional.

Además, se han empleado una serie de librerías que han permitido realizar distintas tareas:

- Pytorch. Librería que permite el desarrollo del código para la implementación de las redes neuronales y su entrenamiento.
- OpenCV. Procesamiento de imágenes.
- Matplotlib, Numpy. Cálculos con vectores, realización de gráficas de errores.
- Os. Para poder listar rutas y acceder a directorios.

Para el desarrollo del código y los scripts se ha utilizado el IDE de programación PyCharm, que es uno de los más completos para Python. Facilita la depuración del código, la inclusión de *plugins* y librerías, además de la integración con Conda.

En el ordenador portátil personal se tiene instalado el sistema operativo Windows 10. Esto ha sido necesario ya que se ha empleado el programa Matlab R2019b para realizar tareas de procesamiento de imágenes y para la generación de nuevas imágenes de entrenamiento entre otras tareas. Esto no era posible realizarlo en Ubuntu, ya que Matlab no está disponible para Linux.

5.2 Etiquetado y generación del *dataset*

Como punto de partida, se dispone de una base de datos con distintos ensayos de *Lifespan* con *C. elegans* realizados en placas de Petri estándar, utilizando para la adquisición de imágenes el sistema de visión artificial del instituto ai2 comentado en el apartado de antecedentes. En estos ensayos se captura cada día una secuencia de 30 imágenes a 1fps para ser posteriormente procesadas. Este conjunto de imágenes tiene una resolución de 1944x1944 píxeles y está almacenado en formato binario comprimido siguiendo la estructura de carpetas mostrada en la figura.

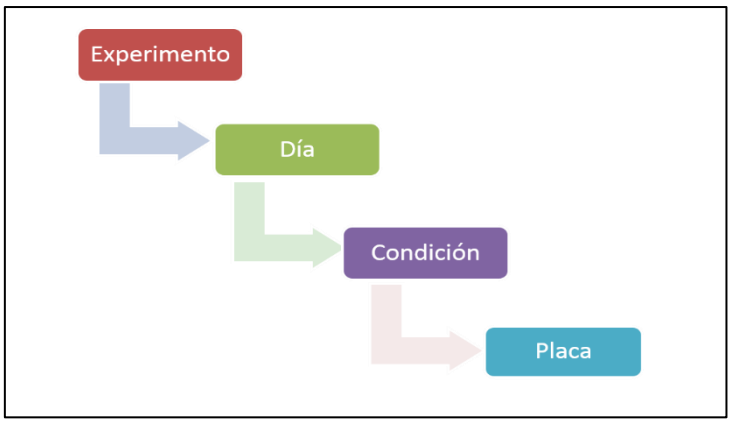

Fig. 13 Estructura base de datos original

Las imágenes se obtienen a partir de una técnica de luz trasera (*backlight)* tal y como se observa en la figura 14*,* de forma que los *C. elegans* aparecen de color negro y la placa de Petri de color gris. Las imágenes quedan oscuras porque se capturan con una intensidad de iluminación baja para evitar estresar a los nematodos.

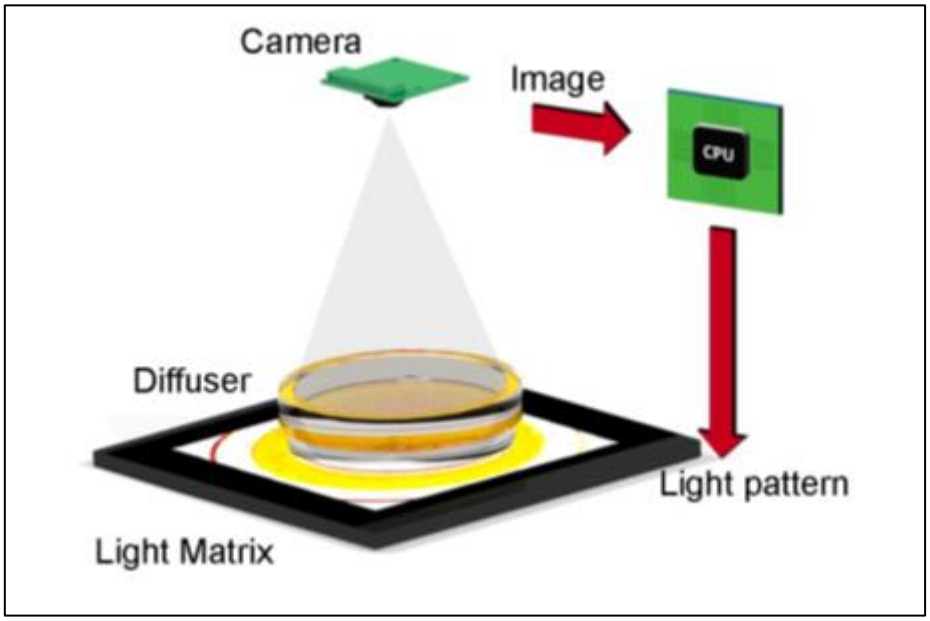

Fig. 14 Sistema de visión con control inteligente de iluminación del instituto ai2 (Puchalt et al., 2019)

Posteriormente se elimina todo lo que queda fuera de la circunferencia de la placa de Petri para facilitar la segmentación, obteniéndose una imagen como la de la figura 15.

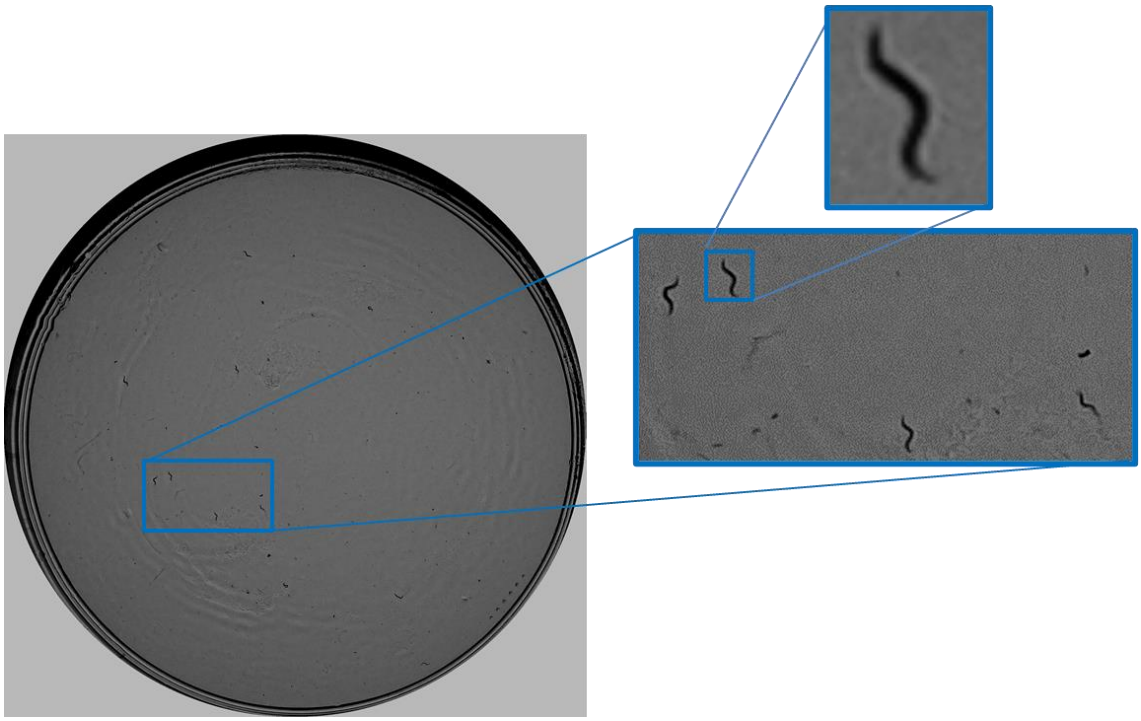

Fig. 15 Ejemplo imagen dataset con zoom para visualizar los C. elegans

Como se ha comentado en el apartado de antecedentes, el alumno ya ha trabajado con este *dataset* para un proyecto anterior. En ese proyecto se etiquetó la base de datos utilizando el procedimiento mostrado en la figura 16:

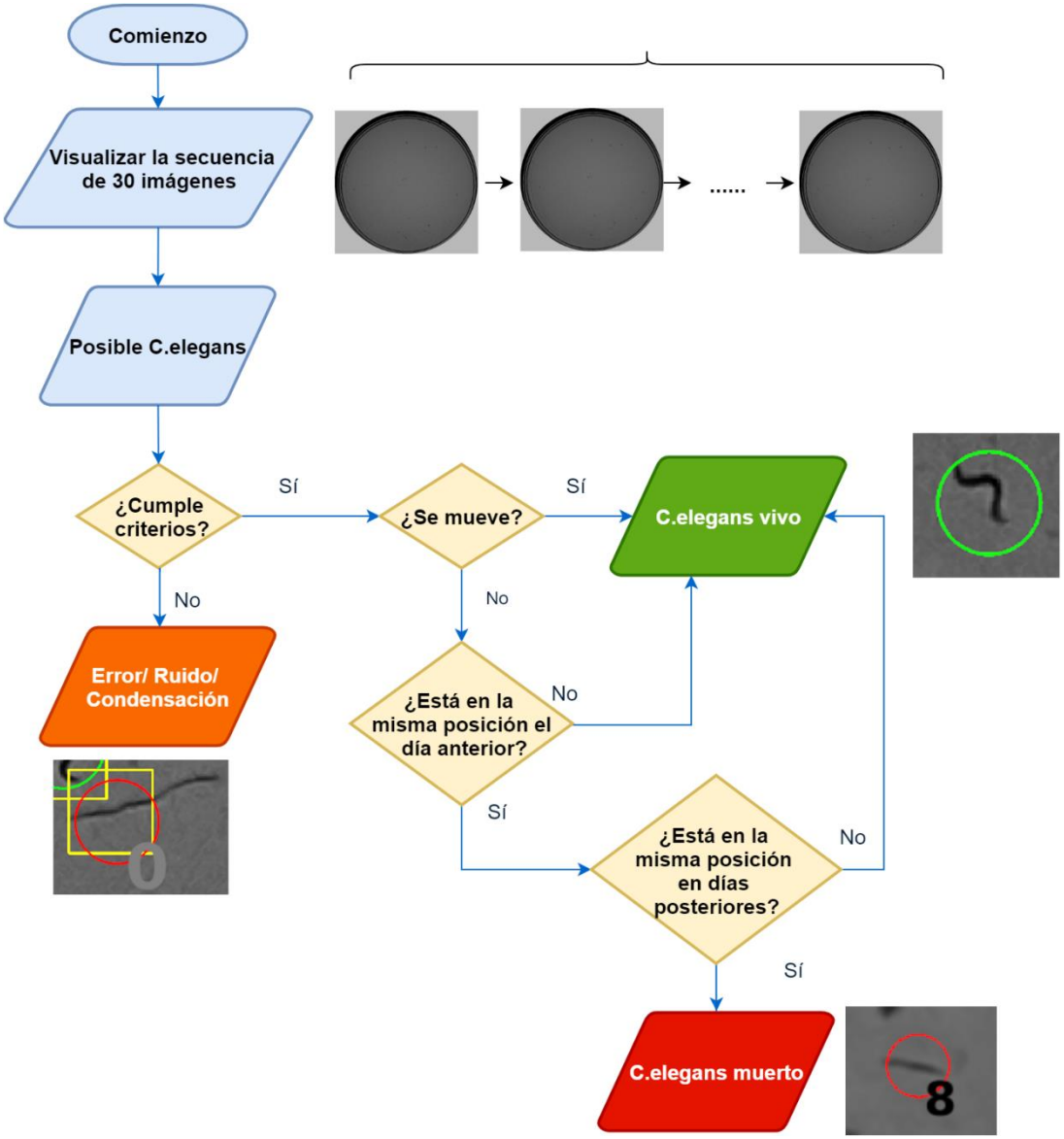

Fig. 16 Flujograma procedimiento de etiquetado

En síntesis, el etiquetado consiste en identificar el *C. elegans* y analizar su movimiento durante la secuencia de 30 imágenes. De esta forma, si un *C. elegans* no varía su posición respecto al día anterior y en días posteriores sigue en la misma posición se considera muerto, mientras que si se mueve se considera vivo.

Como resultado de este etiquetado, dentro de la misma carpeta donde se encuentra la secuencia de imágenes, se tiene un archivo xml llamado conteoManual.xml como el que se muestra en la figura 17. En este archivo están las coordenadas (x,y) de la posición del centro de la etiqueta de los *C. elegans* etiquetados como vivos en la última imagen de la secuencia.

```
<?xml version="1.0"?>
<opencv_storage>
<CE_Manuals>
  1228 594 545 890 1221 197 917 1845 978 1766</CE_Manuals>
</opencv_storage>
```

Fig. 17 Archivo xml de etiquetados

Como el objetivo de este TFM es que la red prediga el número de *C. elegans* vivos, no son necesarias las posiciones, sino el número de *C. elegans*. Para obtener el valor a partir del archivo xml etiquetado, simplemente se divide la longitud de la lista de coordenadas (x, y) entre 2, obteniendo así el conteo.

$$c.elegans\ vivos = longitud\ lista\ coordenadas\ /2$$

Ecuación 1 Cálculo C. elegans vivos

Una vez obtenido esto, se generan las imágenes en formato .jpg y se estructura el *dataset* por placas, tal y como se muestra en la figura 18:

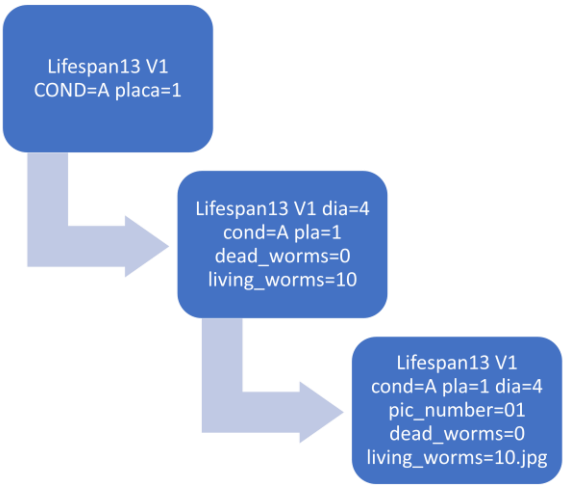

Fig. 18 Estructura de carpetas nuevo dataset

Dentro de la carpeta de una placa se encuentran las subcarpetas de cada uno de los días que ha durado el experimento, con las 30 imágenes capturadas en ese día.

En este caso no se necesita un archivo adicional con las etiquetas, ya que los nombres de las carpetas contienen la información del número de *C. elegans* vivos y muertos, permitiendo disponer de esta manera de las etiquetas. El número de *C. elegans* muertos no se va a utilizar para este proyecto, pero dado que se dispone de este conteo se ha incluido por si se empleara en algún proyecto futuro.

5.3 Método y arquitectura de red

Como se ha comentado en el apartado de objetivos, se ha propuesto un método que utiliza como entrada a la red neuronal una secuencia de imágenes. A la salida, se obtiene una predicción con el número de *C. elegans* vivos.

Una secuencia de imágenes o *frames* puede ser tratada como un video. En el estado del arte del tratamiento de video utilizando aprendizaje profundo, se encuentran técnicas de clasificación y de descripciones de escenas, tal y como se ve en la siguiente figura:

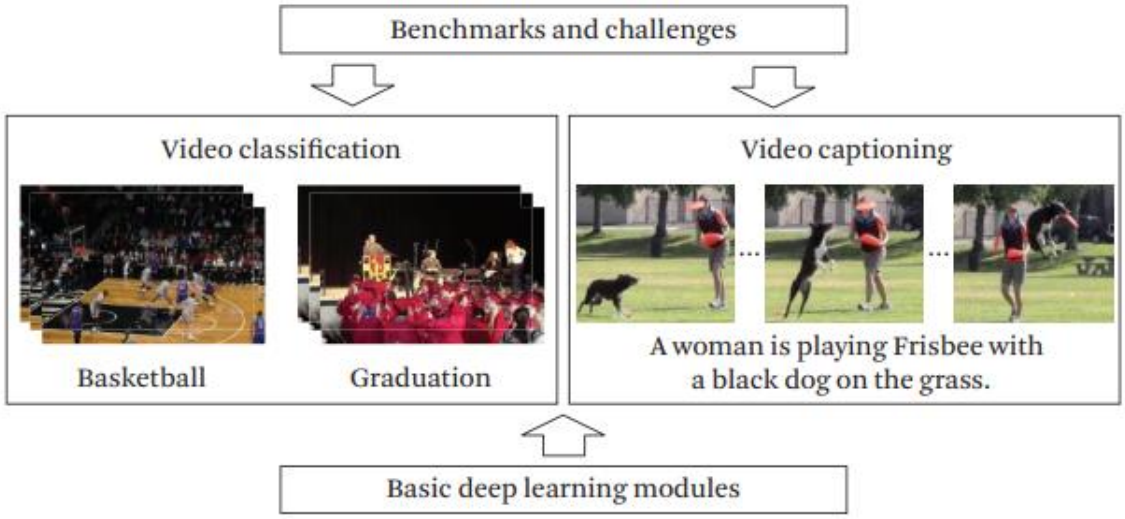

Fig. 19 Estado del arte video y aprendizaje profundo (Wu et al., 2017)

Una de las prácticas más comunes a la hora de diseñar las arquitecturas de redes neuronales para el análisis de video es la combinación de redes neuronales convolucionales con redes recurrentes, de forma que la red convolucional se encarga de la extracción de características y la recurrente de procesar la dinámica temporal.

Se han propuesto dos arquitecturas:

- Una red CNN-LSTM-FC propuesta por miembros del ai2.
- Una red LRCN extraída de un artículo que resuelve problemas de clasificación de videos.

A continuación, se describen las alternativas planteadas para este TFM.

5.3.1 Red CNN-LSTM-FC

La primera propuesta de arquitectura de red es la siguiente:

Conv2d_1 (1, 4, kernel_size=(5, 5), stride=(1, 1))
Batchnorm2D_1
LeakyReLu
Max_pooling2d_1 (kernel_size=(2, 2), stride=(2, 2))
Conv2d_2 (4, 8, kernel_size=(5, 5), stride=(1, 1))
Batchnorm2D_2
LeakyReLu
Max_pooling2d_2 (kernel_size=(4, 4), stride=(4, 4))
Conv2d_3 (8, 16, kernel_size=(5, 5), stride=(1, 1)
Batchnorm2D_3
LeakyReLu
Max_pooling2d_3 (kernel_size=(3, 3), stride=(3, 3))
Reshape (batch_size,seq_length,16 * size_after_cnn^2)
LSTM layer (16*size_after_cnn^2, hidden_size=1024)
Reshape (batch_size, hidden_size*seq_length)
FC Layer_1 (2000,2000)
Batchnorm1D_1
Dropout (50%)
LeakyReLu
FC Layer_2 (2000,750)
Batchnorm1D_2
Dropout (50%)
LeakyReLu
FC Layer_3 (750,seq_length)

Tabla 1 Arquitectura CNN-LSTM-FC

Esta arquitectura está formada por 3 capas convolucionales seguidas de una capa LSTM y de 3 capas densamente conectadas (*fully connected*).

Capas convolucionales

Cada una de las capas convolucionales están formadas por la etapa de convolución , la capa de *batch normalization*, la función de activación y la capa de *max pooling*.

Las redes convolucionales permiten mantener la información de las relaciones espaciales y aprender a detectar una serie de características. Esto se hace aplicando una serie de filtros o *kernels* a las imágenes de entrada.

En la siguiente figura se puede ver un ejemplo de la operación de convolución. En este caso, la imagen sería el cuadrado azul claro de 4x4, el filtro sería el cuadrado azul oscuro de 3x3 y el mapa de características el cuadrado verde de 2x2.

Como se puede ver, el filtro se va desplazando por la imagen realizando un producto de estas matrices, cuyo valor se guarda en el mapa de características.

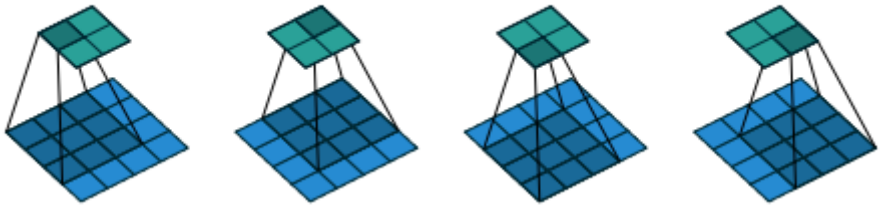

Fig. 20 Ejemplo operación de convolución

Como resultado de esta operación se obtienen distintas imágenes de salida (mapas de características), cada una de ellas puede detectar una característica, como por ejemplo bordes, orientaciones, patrones, etc. Estos filtros se inicializan con valores (pesos) aleatorios y se van actualizando a su valor óptimo durante el entrenamiento.

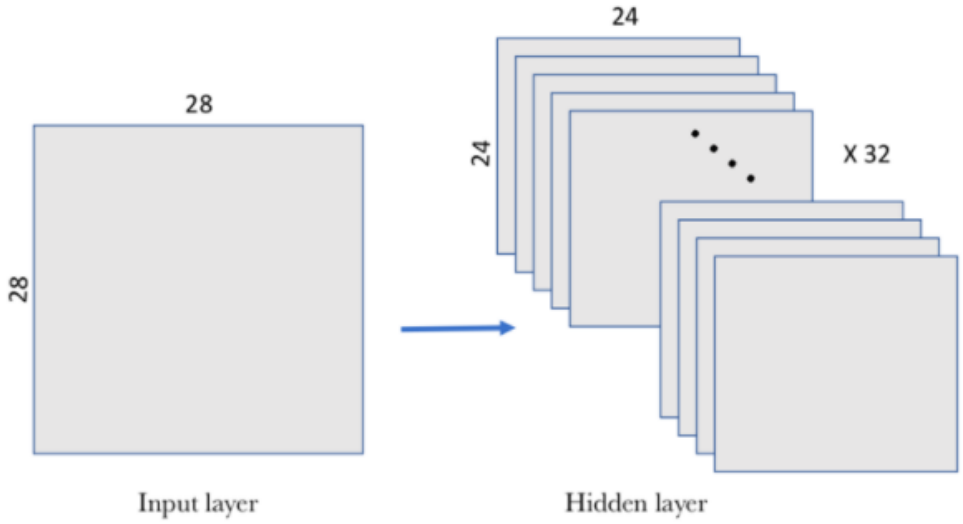

Fig. 21 Ejemplo salida de operaciones de convolución

En la **capa de convolución** existen un conjunto de parámetros que determinan la dimensión de los mapas de características resultantes:

- *Kernel size*. Es el tamaño de la ventana del filtro empleado.
- *Stride.* Es el paso o número de píxeles que se desplaza el filtro en cada paso.
- *Padding*. Consiste en rellenar la imagen alrededor de los bordes (normalmente con píxeles de valor 0) para así evitar perder dimensiones de la imagen al aplicar el filtro, ya que los *kernels* no pueden llegar a los bordes.

Teniendo en cuenta la combinación de estos parámetros, la dimensión del mapa de características resultante es el siguiente:

$$T = \frac{M - K + 2 \cdot P}{S} + 1$$

Ecuación 2 Tamaño mapa de características

Donde M es el valor del ancho de la imagen, K es el *kernel size*, P es el *padding* y S el valor de *stride*.

En el caso de esta red se ha escogido un *kernel* de 5x5 y un *stride* de 1.

Se ha utilizado la técnica de ***batch normalization*** (Ioffe & Szegedy, 2015), ya que durante el entrenamiento se producen cambios en la distribución de las entradas de las capas debido a la variación de los parámetros, ralentizando el aprendizaje. Por ello, con esta técnica se normaliza restando la media y dividiendo por la desviación estándar.

En la siguiente figura, extraída del artículo (Ioffe & Szegedy, 2015) se observa cómo se acelera el proceso de aprendizaje (a) y la diferencia en la distribución de las entradas (b) sin *batch normalization* y (c) con *batch normalization*, que es más estable en este caso.

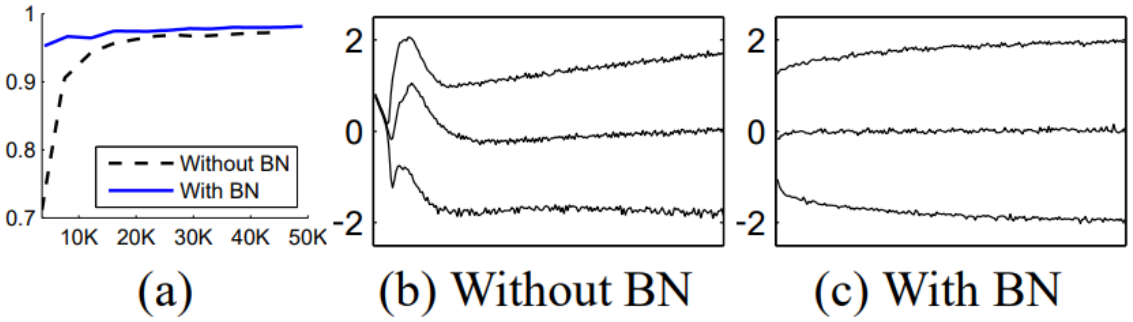

Fig. 22 Efecto batch normalization (Ioffe & Szegedy, 2015)

Como **función de activación** se ha empleado la LeakyReLu, que a diferencia de la función ReLu no es cero para los valores negativos, sino que tiene una pequeña pendiente que evita el problema de que algunas neuronas queden inactivas. En la siguiente figura se puede ver la comparativa entre ambas funciones:

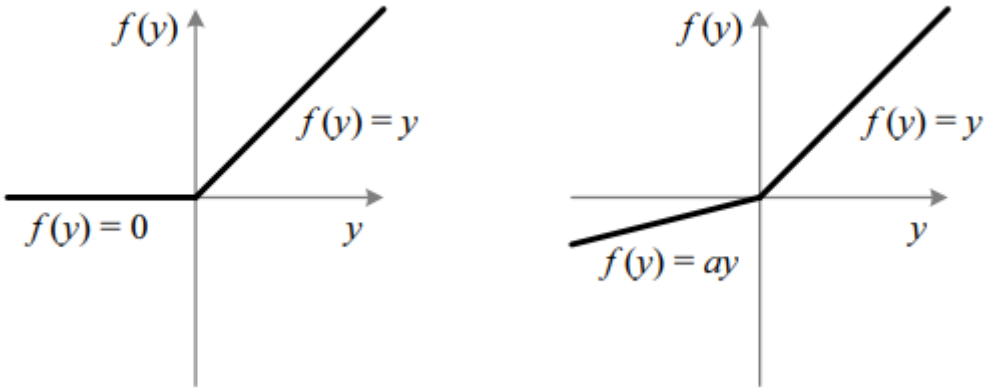

Fig. 23 ReLu vs LeakyRelu (He et al., 2015)

Tras esto, se pasa a una capa de **max poolling**, que permite reducir el tamaño de las imágenes, disminuyendo así el número de parámetros del modelo. Como se puede ver en la siguiente figura, se reduce el tamaño de las imágenes, pero se mantiene el número de canales. La idea es mantener las características más importantes detectadas por el filtro evitando que el número de parámetros sea excesivo.

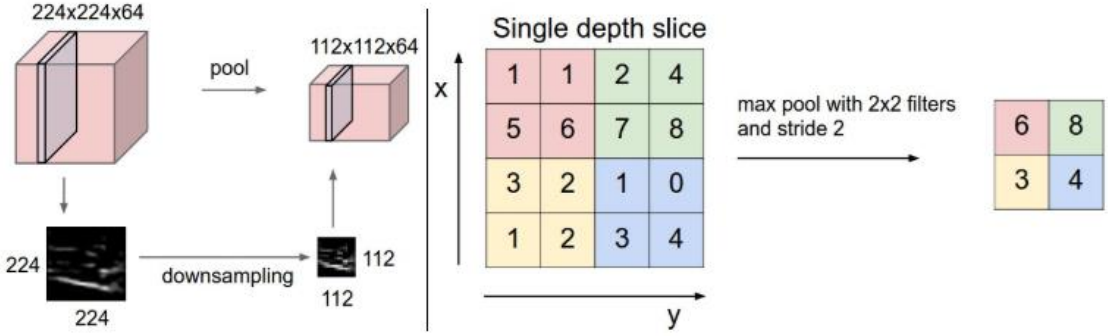

*Fig. 24 Operación de MaxPoolling (*CS231n Convolutional Neural Networks for Visual Recognition*, s. f.)*

Capa LSTM

Las características extraídas de las capas convolucionales son la entrada de una capa de red recurrente (RNN), que son las más apropiadas para trabajar con datos secuenciales. Las RNN funcionan de forma que la entrada de estas redes sea la entrada en el instante actual (X_t) y el estado del instante de tiempo anterior (h_{t-1}). En la siguiente figura podemos ver su esquema:

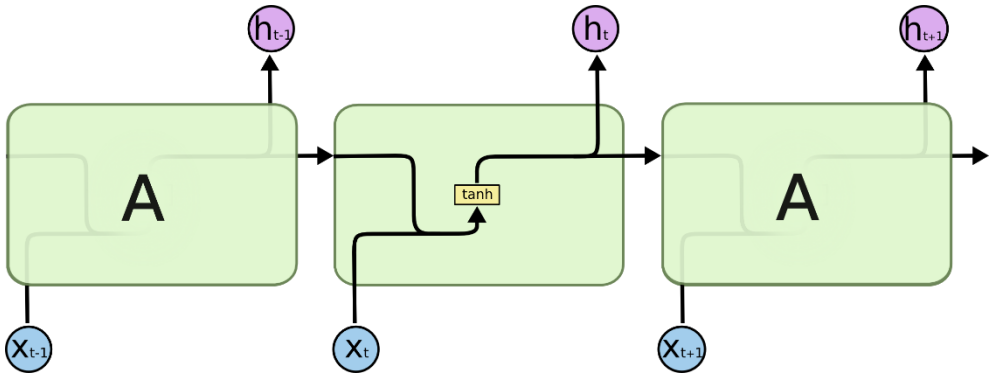

Fig. 25 Esquema RNN básica

Las RNN presentan problemas con las dependencias de tiempo a largo plazo. Por ello, surgieron las redes LSTM (*Long Short-Term Memory*) que, a diferencia de las redes recurrentes básicas, que presentan un módulo con una sola capa, tienen un módulo o red neuronal con cuatro capas.

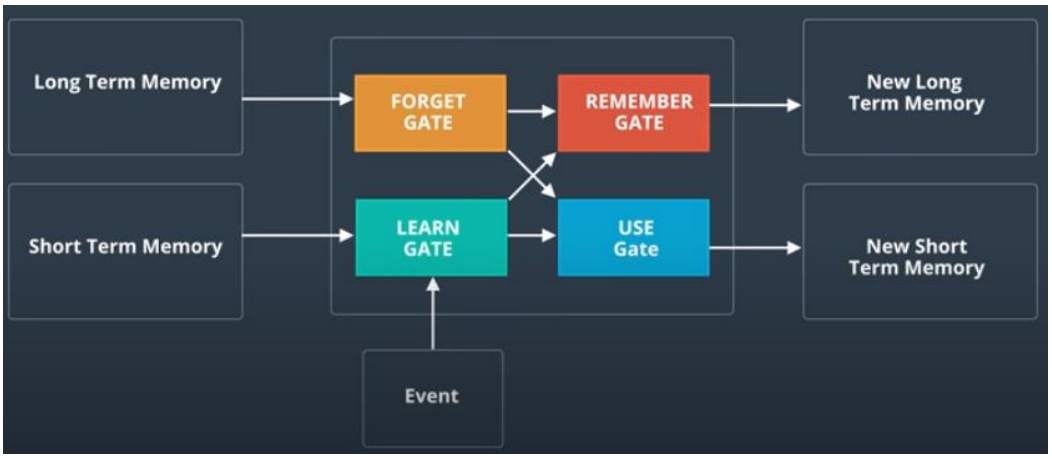

Fig. 26 Esquema LSTM

Como se puede ver en la figura 26, se tiene como entrada el evento actual, el *Long Term Memory*, que es la información a largo plazo almacenada y el *Short Term Memory* que es la salida anterior.

Dentro de la red existen 4 puertas (*gates)* que realizan lo siguiente:

- Puerta de aprendizaje (*learn gate*). Combina el evento actual y el *Short Term Memory*, quedándose solo con la información más importante.

- Puerta de olvidar (*forget gate*). Recibe el *Long Term Memory* y decide qué información es más relevante para mantener y cual se debe olvidar.
- Puerta de recordar (*remember gate*). Combina la información de las puertas de olvidar y aprender para obtener un nuevo *Long Term Memory*.
- Puerta de salida (*use gate*). Utiliza las salidas información de las puertas de olvidar y aprender para obtener una salida o predicción, que será el nuevo *Short Term Memory*.

Capas densamente conectadas

Las características de salida de la red LSTM pasan a una red densamente conectada (*fully connected*), en la que cada neurona está conectada con todas las salidas de la capa anterior y que permite obtener a la salida un vector de la dimensión de nuestra salida.

En estas capas se ha utilizado la técnica de *dropout,* cuya función es la de reducir el *overfiting*. Con esta técnica se realizan desconexiones al azar de un porcentaje de las neuronas (con un valor comprendido entre 0 y 1) en cada iteración de entrenamiento tal y como se ve en la siguiente comparativa (figura27).

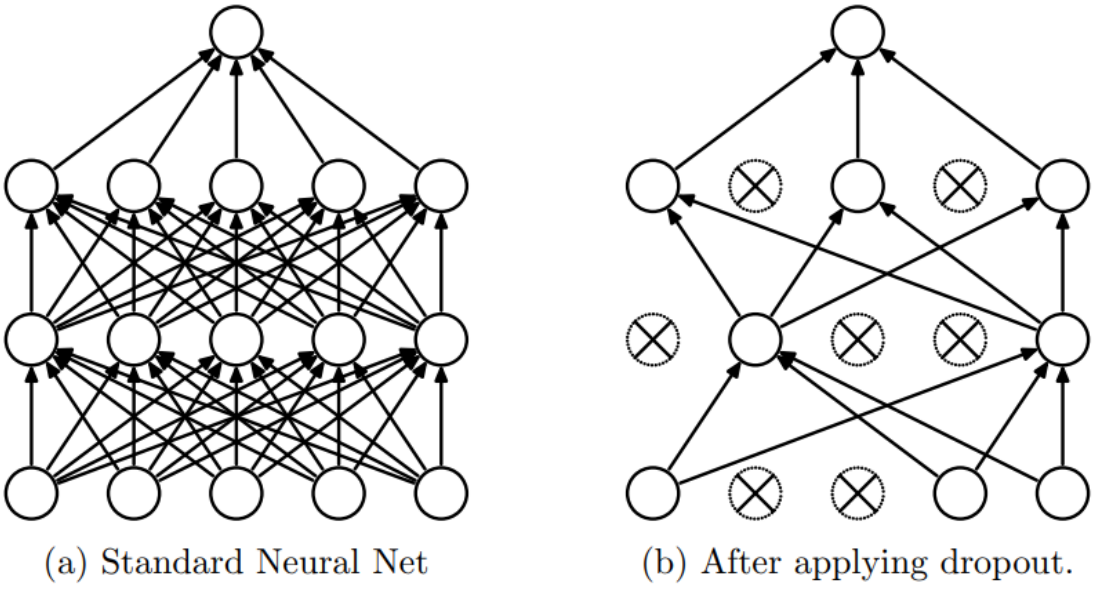

Fig. 27 Técnica de dropout (Srivastava et al., 2014)

En estas capas densamente conectadas, al igual que en las capas convolucionales, también se ha empleado como función de activación LeakyReLu, y se ha aplicado la técnica de *batch normalization*.

5.3.2 Red LRCNs

Esta red (*Long-term Recurrent Convolutional Network*) se ha obtenido del artículo (Donahue et al., 2017) en el que se propone utilizar una red convolucional preentrenada junto con una red LSTM.

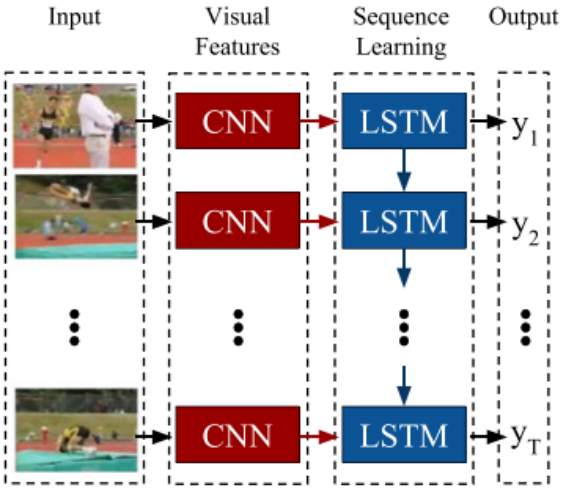

Fig. 28 LRCN (Donahue et al., 2017)

En este caso la arquitectura que se ha probado utiliza una red preentrenada ResNet18 junto con una red recurrente LSTM bidireccional.

Resnet 18 Preentrenada
LSTM bidirectional(512, 256, num_layers=2 bidirectional)
Linear(in_features=512, out_features=seq_length)

Tabla 2 Arquitectura red LRCN

ResNet o Deep Residual Convolutional Network

En general, cuanto más capas tenga una red mejor desempeño podrá tener en la resolución de la tarea a la que se enfrenta. Sin embargo, en la práctica llega un momento en que esto deja de ser así debido a lo que se conoce como *vanishing gradient problem*. Este problema se debe a que el gradiente del error se transmite desde las últimas capas a las primeras (*backpropagation*) y se va haciendo progresivamente menor. Si la red es muy profunda, llega un momento en que el gradiente es prácticamente nulo y por tanto no se produce actualización de los pesos, la red deja de aprender.

Para solucionar este problema surgen las redes residuales (ResNet), que lo que hacen es conectar una capa con la siguiente y además con otra capa que esté a N capas de distancia. Esto es lo que se conoce como *skip connections*. Se ha demostrado empíricamente que estas redes facilitan la optimización y aumentan la precisión de las redes profundas.

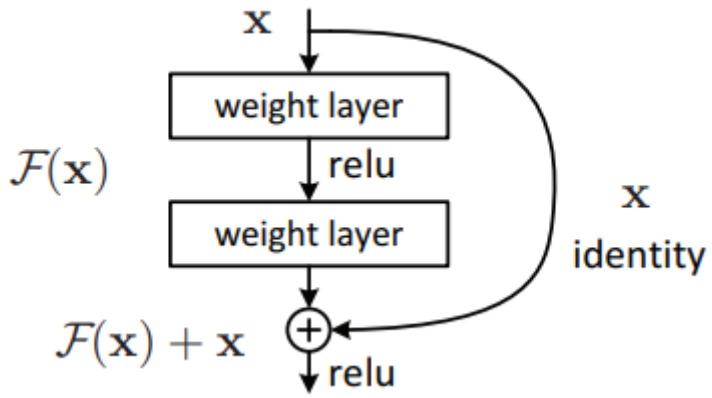

Fig. 29 Ejemplo bloque residual(He et al., 2016)

En la siguiente figura se puede ver la arquitectura ResNet18, que recibe ese nombre por las 18 capas que tiene.

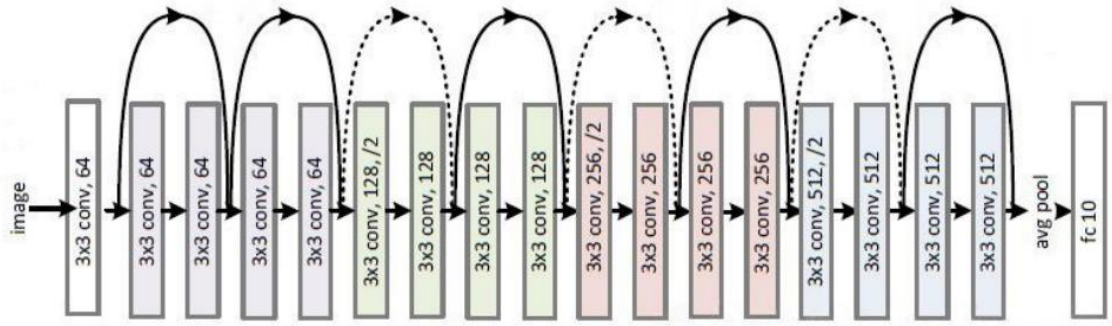

Fig. 30 Arquitectura Resnet18

LSTM bidireccional

Las redes recurrentes bidireccionales fueron propuestas por primera vez en 1997 por Schuster y Paliwal (Schuster & Paliwal, 1997). Esta arquitectura recorre las secuencias en dos direcciones temporales: de atrás hacia adelante y de adelante hacia atrás. Utilizando estás dos direcciones la capa de salida obtiene información de los estados pasados y futuros.

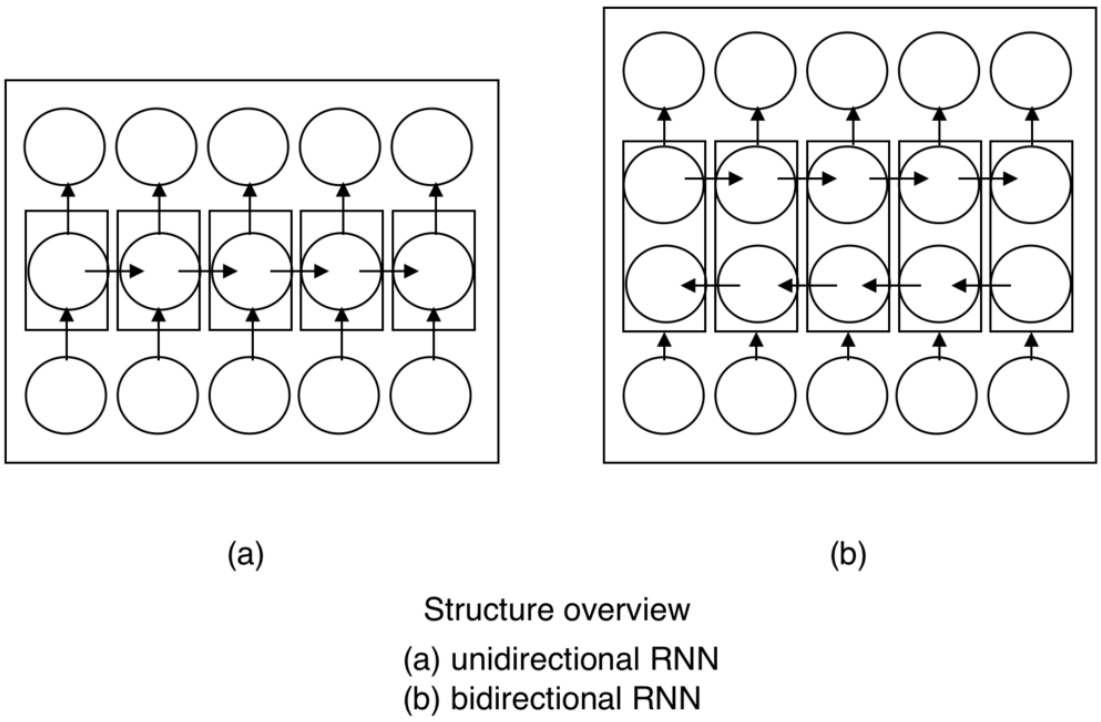

Fig. 31 Comparativa RNN unidireccional vs RNN bidireccional(«Bidirectional Recurrent Neural Networks», 2020)

5.4 Métodos de introducción de imágenes

En el apartado anterior, se ha definido una arquitectura de red neuronal que acepta como entrada una secuencia de imágenes. En este apartado se proponen distintas formas de introducir las secuencias. Estos métodos se pueden dividir en aquellos que realizan el conteo de *C. elegans* vivos en un día concreto y los que realizan el procesamiento completo del experimento de *Lifespan*.

5.4.1 Método de procesamiento del experimento completo de *Lifespan*

5.4.1.1 Método secuencia 57 imágenes, 1 img/día

Este método utiliza como entrada a la red una secuencia con una imagen de cada día del experimento. A la salida, se obtiene un vector con el número de *C. elegans* vivos en cada día.

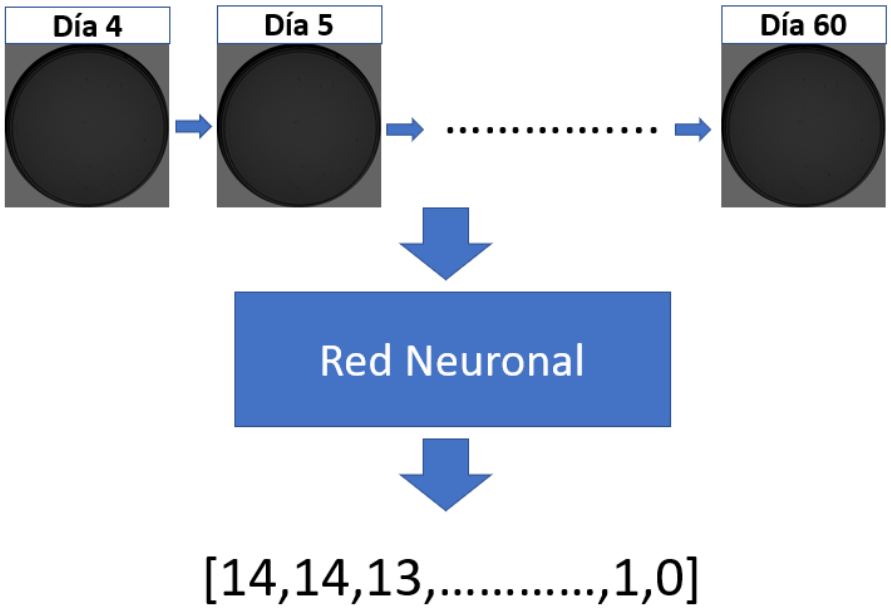

Fig. 32 Método 57 imágenes, 1 imagen/día

Uno de los problemas de este método, es que los experimentos pueden tener distinto número de días de duración, sin embargo, la longitud de la secuencia de entrada a la red neuronal debe ser fija. Por tanto, es necesario establecer una longitud de secuencia máxima.

En la base de datos disponible, el experimento más largo tiene una duración de 60 días.

El conteo se empieza a realizar a partir del día 4, que es aproximadamente cuando comienza la edad adulta de los *C. elegans*, por tanto, la longitud de la secuencia será:

$$longitud\ secuencia = día\ final - \ día\ inicial + 1 = 60 - 4 + 1 = 57$$

Ecuación 3 Cálculo longitud de secuencia máxima

En los experimentos que tienen una duración inferior, se replica la última imagen disponible hasta el final. Si esa imagen contiene algún *C. elegans* vivo, se utiliza la imagen, pero se modifica la etiqueta a muerto, ya que se va a mantener una imagen estática.

El otro inconveniente que presenta este método es que los experimentos presentan huecos en la secuencia de imágenes, ya que los días festivos y los fines de semana no se capturan imágenes.

Para resolver este problema se han propuesto 2 soluciones:

1. **Emplear un simulador que genere imágenes para completar los huecos de la siguiente manera**:
   Si hay un día sin imágenes capturadas, se utiliza una imagen de un día posterior como fondo y se simulan sobre ella un número de *C. elegans* en movimiento equivalente a la interpolación lineal entre el número de nematodos vivos antes y después del hueco.

2. Utilizar imágenes alternas

Este método de introducción de datos a la red utiliza secuencias con 1 imagen de cada día, tomando siempre la misma posición (entre 1 y 30). Si en uno de los días no hay datos, se pueden utilizar las imágenes del día anterior con índice separado en 14 pasos. De esta forma, se evita que la imagen sea estática y se genera la sensación de movimiento. Es decir, existe variación entre las imágenes con las que se ha completado el hueco. Si no se hiciera esto y se mantuviera la misma imagen podría interpretarse que los *C. elegans* están muertos al no haber movimiento.

En la siguiente figura se puede ver un ejemplo de este método. Como se puede ver, en los días 6 y 7 no hay imágenes capturadas. En esta secuencia se están utilizando las imágenes de índice 1 (la primera de las 30 capturadas ese día). Entonces, la imagen del día se completa con la imagen del día 5 de índice 15 y la imagen del día 7 con la imagen del día 5 con la imagen del día 5 de índice 1.

			Sin datos				
Día	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Día imagen empleada	**4**	**5**	**5**	**5**	**8**	**9**	**10**
Índice secuencia	1	1	1	1	1	1	1
	2	2	2	2	2	2	2
	3	3	3	3	3	3	3
	4	4	4	4	4	4	4
	5	5	5	5	5	5	5
	6	6	6	6	6	6	6
	7	7	7	7	7	7	7
	8	8	8	8	8	8	8
	9	9	9	9	9	9	9
	10	10	10	10	10	10	10
	11	11	11	11	11	11	11
	12	12	12	12	12	12	12
	13	13	13	13	13	13	13
	14	14	14	14	14	14	14
	15	15	15	15	15	15	15
	16	16	16	16	16	16	16
	17	17	17	17	17	17	17
	18	18	18	18	18	18	18
	19	19	19	19	19	19	19
	20	20	20	20	20	20	20
	21	21	21	21	21	21	21
	22	22	22	22	22	22	22
	23	23	23	23	23	23	23
	24	24	24	24	24	24	24
	25	25	25	25	25	25	25
	26	26	26	26	26	26	26
	27	27	27	27	27	27	27
	28	28	28	28	28	28	28
	29	29	29	29	29	29	29
	30	30	30	30	30	30	30

Fig. 33 Esquema completado de secuencia utilizando imágenes alternas

5.4.2 Método de procesamiento de un día del experimento

Para el conteo de los *C. elegans* vivos en un día concreto se han propuesto varios métodos, que se presentan a continuación.

5.4.2.1 Método 30 imágenes de 1 día

Este método tiene como entrada a la red las 30 imágenes capturadas en un día y a la salida devuelve el número de *C. elegans* vivos.

En la siguiente figura se puede ver un ejemplo de este método:

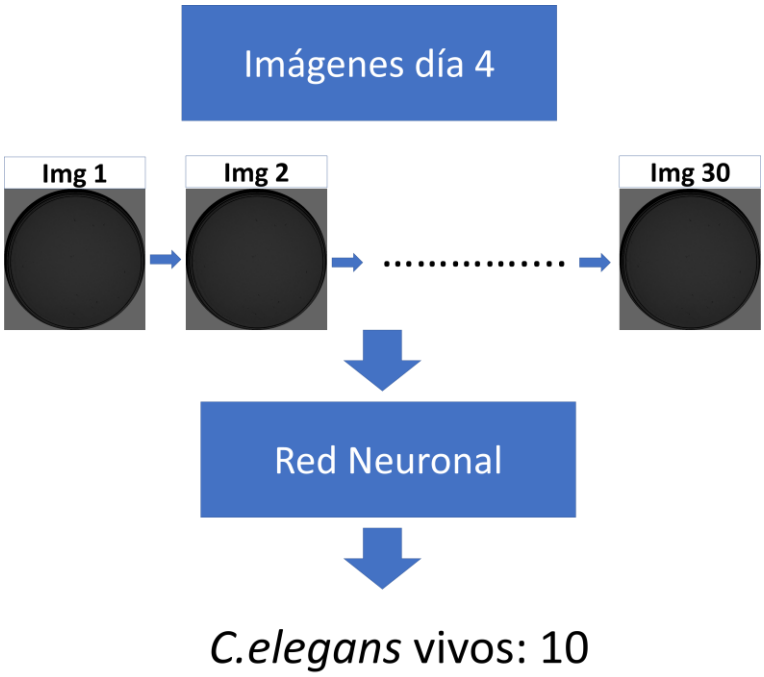

Fig. 34 Método 30 imágenes de 1 día

5.4.2.2 Método 30 imágenes de 1 día + imágenes de día anterior y posterior

La clasificación de un *C. elegans* como vivo o muerto se hace en base al desplazamiento y a los cambios de forma. El criterio seguido para el etiquetado tiene en cuenta esos cambios respecto al día anterior, los cambios que se producen durante la secuencia de 30 imágenes y los días posteriores.

El método anterior tiene el inconveniente de que puede que la información sea insuficiente para saber si los nematodos están vivos, porque, aunque no es lo habitual, puede que un *C. elegans* vivo no se mueva durante la secuencia de 30 imágenes.

Por este motivo, una mejora al método anterior sería incluir en la secuencia una imagen del día anterior y otra del día posterior, teniendo de esta forma más información.

No obstante, este método no se podría emplear si no se tuvieran datos de otros días.

5.4.2.3 Método 3 imágenes: Día actual, día anterior, día posterior

Otra posibilidad diferente a las anteriores y que permite reducir la longitud de la secuencia introducida a la red es utilizar una imagen del día actual, otra del anterior y otra del posterior. De esta forma, comparando la imagen del día actual con las del día anterior y posterior podría ser capaz de predecir cuantos *C. elegans* vivos hay en la placa en dicho día.

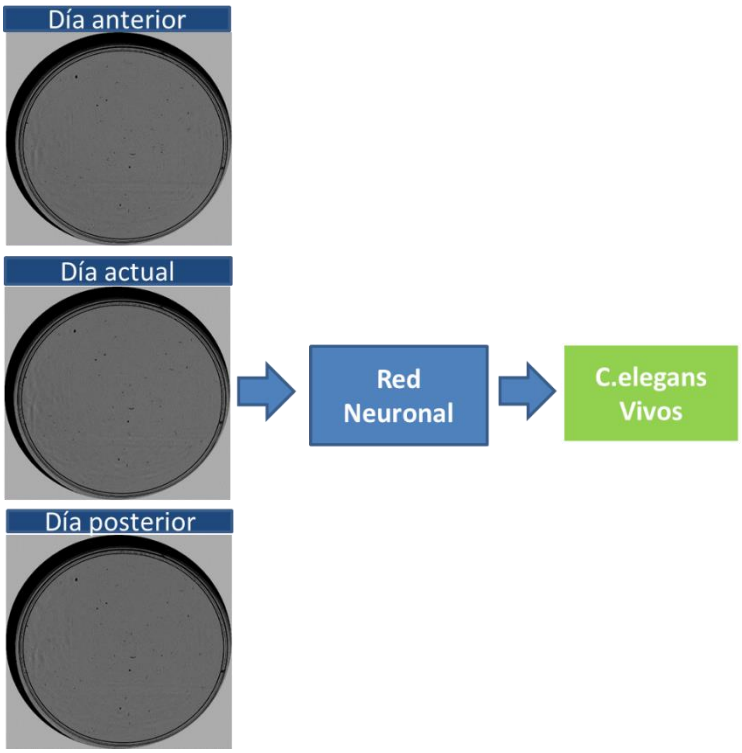

Fig. 35 Método 3 imágenes

5.4.2.4 Método de 4 imágenes: Día actual, día anterior, día posterior y día posterior 2

Este método es análogo al anterior, la diferencia es que añade una imagen más. Está inspirado en un método que se empleó en un trabajo anterior y que dio mejores resultados que utilizar 3 imágenes.

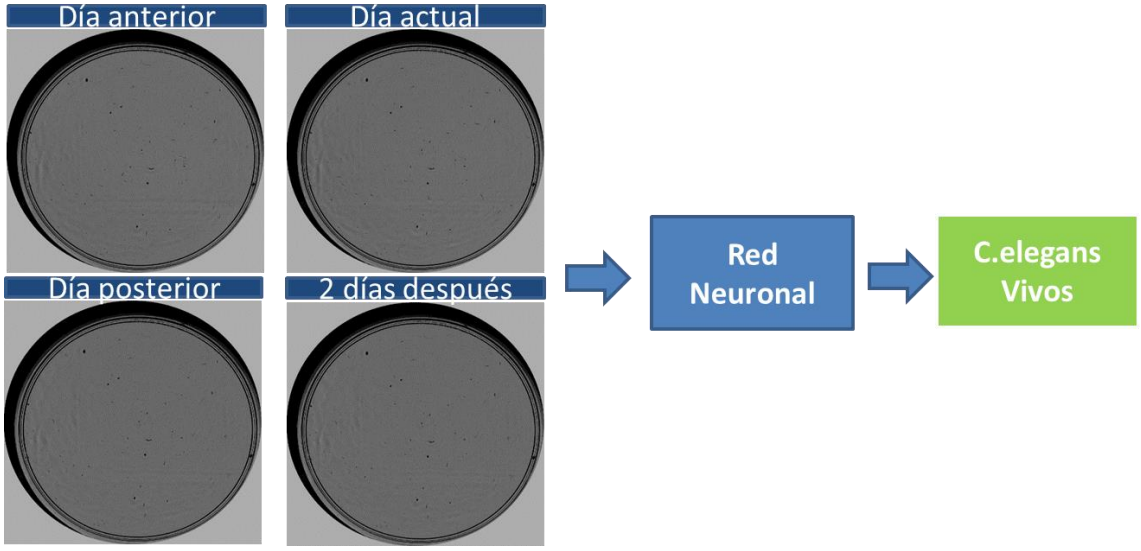

Fig. 36 Método de 4 imágenes

5.5 Aumento de imágenes

Para el correcto aprendizaje de la red es muy importante que el conjunto de datos de entrenamiento empleado sea lo más grande posible. En general, cuanto mayor sea el conjunto de datos utilizado, mejores resultados se pueden obtener, la red generaliza mejor y se reduce el riesgo de sobreentrenamiento (*overfiting*).

En el caso de este proyecto, el conjunto de datos está limitado a la base de datos etiquetada que se dispone. Se podrían capturar nuevas imágenes utilizando el sistema de visión artificial y etiquetarlas, sin embargo, esto es muy costoso y requiere una gran cantidad de tiempo.

Por este motivo, es necesario buscar alternativas para aumentar la base de datos evitando el coste de etiquetado. A continuación, se analizan dos métodos de aumento de imágenes, uno basado en transformaciones sobre las imágenes originales y otro empleando un simulador para generar nuevas muestras.

5.5.1 Aumento de imágenes transformando las imágenes originales

Esta es una de las técnicas más habituales para el aumento de imágenes de entrenamiento, debido a su sencillez. Consiste en aplicar transformaciones como pueden ser:

- Rotaciones
- Volteos verticales y horizontales
- Traslaciones horizontales y verticales
- Zoom
- Cambio de brillos y contrastes
- Recortes

Se pueden realizar de forma *online* y *offline*. Lo óptimo es realizarlo *online* durante el entrenamiento, ya que se evita el tener que almacenar en disco los nuevos datos.

Además, si las transformaciones son aleatorias, la red tendrá nuevas imágenes en cada época de entrenamiento, permitiendo esto una mejor generalización.

En este proyecto se han empleado las siguientes transformaciones para el aumento de imágenes:

- Rotaciones de 90, 180 y 270 grados
- Volteo vertical y volteo horizontal

A continuación, en la figura 37 se muestra un ejemplo de estas transformaciones. Se utiliza un zoom de dos *C. elegans* y se hace un seguimiento de ellos en todas las transformaciones, ya que en las imágenes de la placa completa es difícil apreciar los cambios de posición.

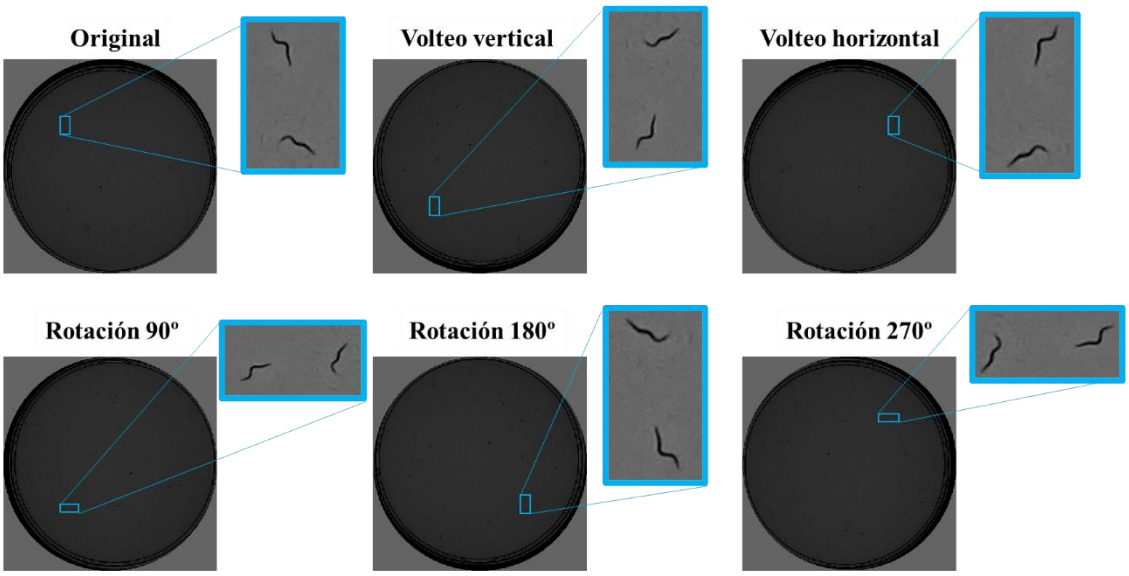

Fig. 37 Transformaciones para el aumento de datos

5.5.2 Aumento de imágenes con simulador

5.5.2.1 Generador de trayectorias

Como ya se ha comentado en el apartado anterior, se dispone de un algoritmo implementado en Matlab desarrollado por miembros del grupo de investigación del ai2 que permite generar trayectorias de *C. elegans* y dibujar estas sobre un fondo de una imagen real.

A continuación, se explica en que consiste el algoritmo y cómo se ha utilizado para crear experimentos nuevos.

El algoritmo recibe los siguientes parámetros:

- Imagen de **fondo**. Es la imagen sobre la que se insertan los *C. elegans*.
- Número de ***C. elegans*** a simular. Permite simular entre 0 y 31.
- **Color** del *C. elegans.* Valor en escala de grises comprendido entre 0 (negro) y 255 (blanco).
- **Velocidad** de movimiento de los *C. elegans.* Admite valores entre 1 y 4.

Con estos parámetros el algoritmo genera una secuencia de 30 imágenes en la que se desplazan el número de *C. elegans* especificado.

Este algoritmo puede ser aprovechado para implementar un simulador que genere nuevos experimentos *Lifespan*, permitiendo así aumentar el *dataset* disponible.

5.5.2.2 Generación de imágenes de fondo

Para poder crear nuevos experimentos utilizando el algoritmo de generación de trayectorias se necesitan imágenes de fondo, es decir, imágenes de placas de Petri sin *C. elegans,* sobre las que insertar los nematodos que genera el algoritmo.

Estas imágenes de fondo se pueden obtener de las imágenes del *dataset* original, eliminando los *C. elegans* presentes en la placa. Con esto se pretende simular haciendo que las imágenes sean lo más parecidas posible a las reales, manteniendo las condiciones que se pueden dar en un experimento de *Lifespan*, como pueden ser el ruido, cambios de iluminación, las condensaciones, etc.

En las imágenes de la placa existen dos zonas diferenciadas. Por un lado, la zona central de la placa o círculo donde se mueven los *C. elegans*. Por otro, la zona de las paredes o bordes de la placa, que presenta ruido y oclusiones. Como el movimiento de los *C. elegans* se produce en mayor parte en la zona central, se obtiene esa zona para comenzar a realizar el procesamiento.

Para el procesamiento en primer lugar, se obtiene el círculo central de la placa de Petri realizando un procesamiento de la secuencia de imágenes:

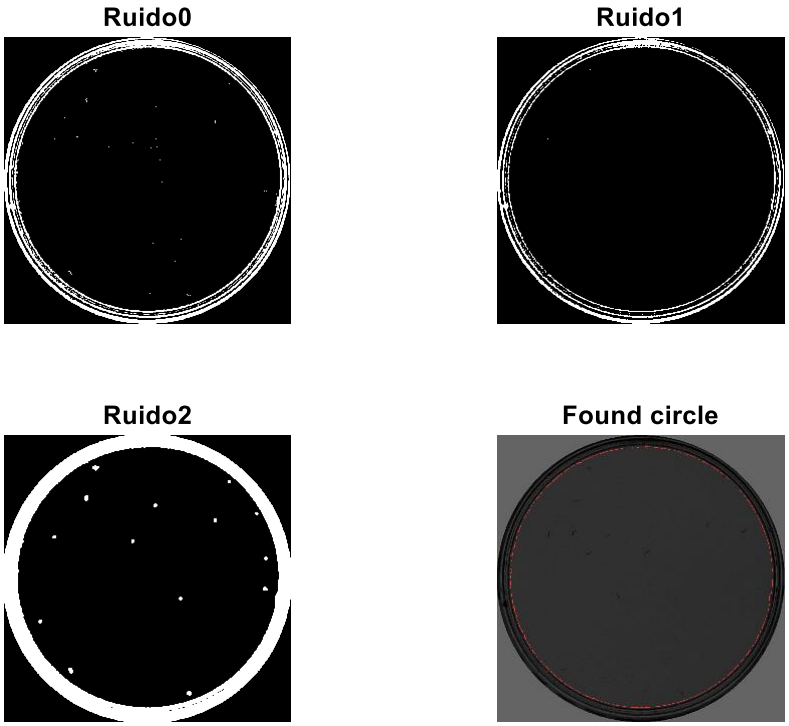

Fig. 38 Obtención del círculo de la placa de Petri

Una vez obtenida una máscara binaria del círculo interior de la placa (todo lo que está dentro vale 1 y lo que está fuera 0), ya se pueden realizar las operaciones deseadas. El objetivo es eliminar los *C. elegans* de la placa para obtener una placa que mantenga las características del ruido del experimento y que sea la imagen de base que utiliza el simulador para crear nuevos datos de entrenamiento.

Se extrae de la imagen original la zona del círculo donde se encuentran los *C. elegans* multiplicándola por la máscara del círculo binario:

Fig. 39 Obtención de la zona de movimiento de los C. elegans

Tras esto, se eliminan los *C. elegans* recorriendo los píxeles y cambiando aquellos que se encuentran en el rango de color del cuerpo de *C. elegans* a un valor de fondo. Este rango es variable, en este caso se ha utilizado [0,35]. Como valor de fondo se utiliza un número aleatorio comprendido entre [40,50].

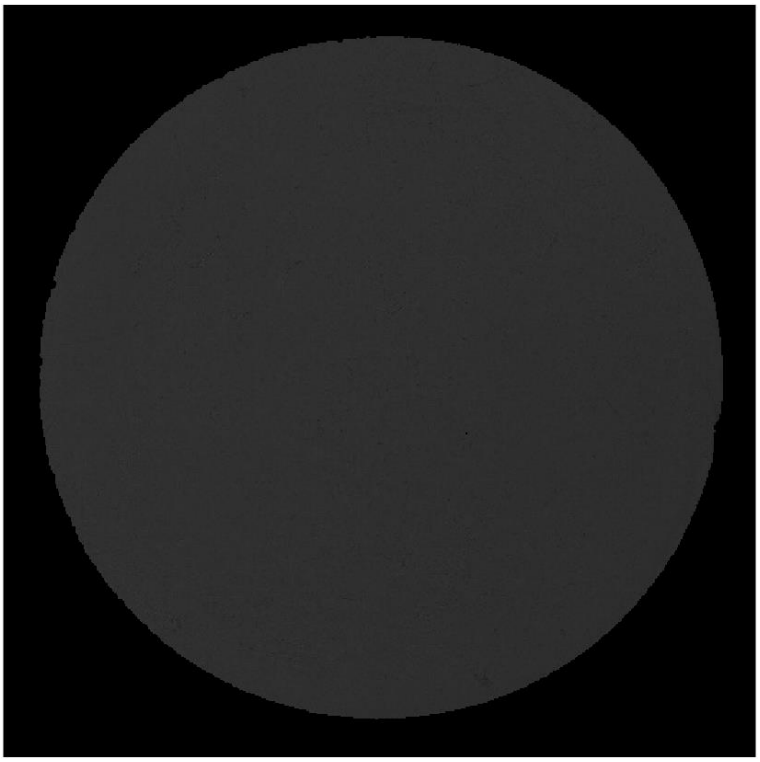

Fig. 40 Zona central de la placa de Petri sin C. elegans

Una vez se tiene la imagen de la zona central sin *C. elegans* (figura 40), solo falta añadir la parte del borde, que no se ha modificado durante el procesamiento.

Para obtener el borde, se invierte la máscara del círculo de la placa y se multiplica por la imagen original, tal y como se ve en la siguiente figura:

Fig. 41 Obtención del borde de la placa de Petri

Por último, se suman la zona exterior (borde) y la zona interior modificada para obtener la nueva imagen:

Fig. 42 Imagen de fondo final

A continuación, en las figuras 43 y 44 se muestra un ejemplo del resultado del procesamiento. En la figura 43, se ve la imagen original con los *C. elegans* en el *zoom*.

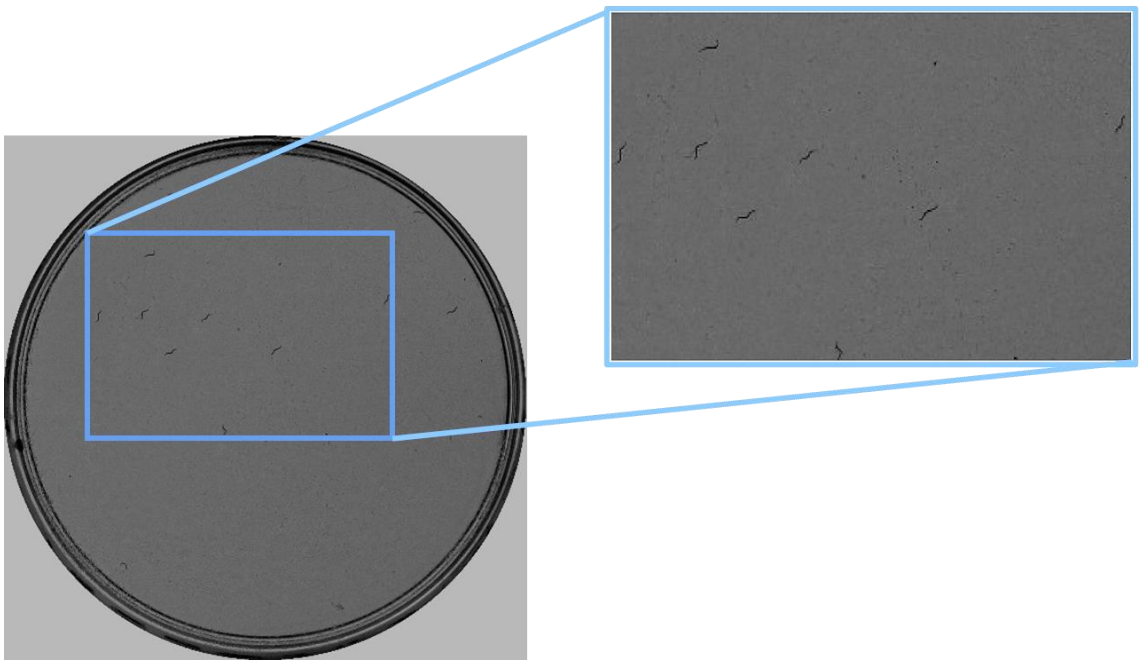

Fig. 43 Imagen de partida del procesamiento

En la figura 44, se ve en la ventana ampliada como tras el procesamiento no hay *C. elegans* en la placa de Petri.

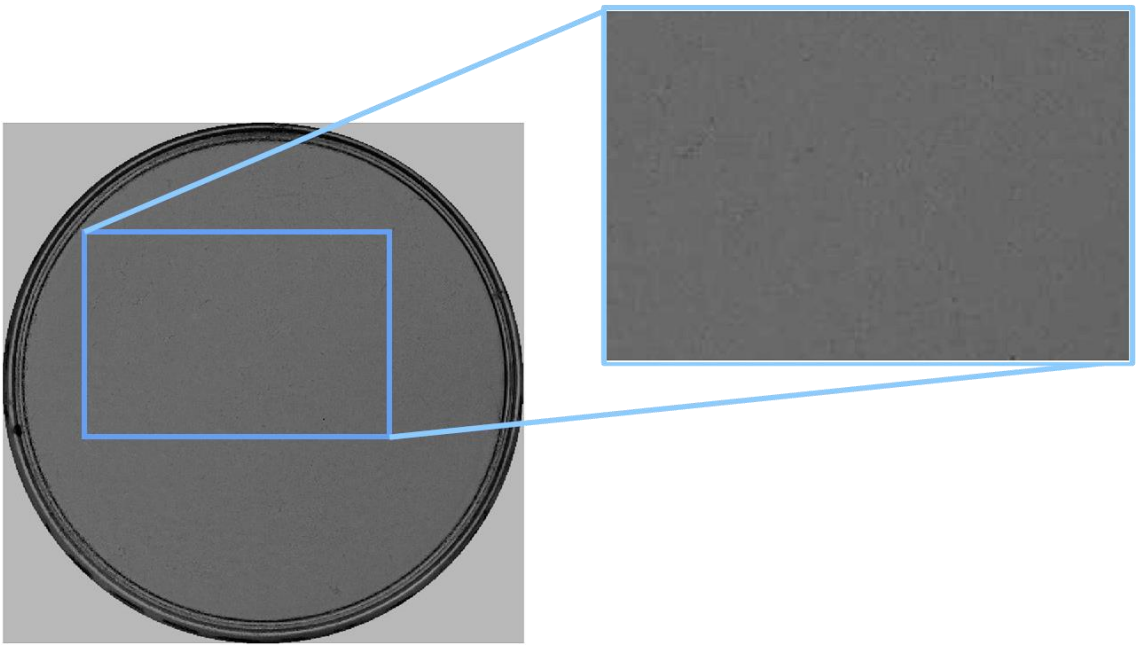

Fig. 44 Resultado final del procesamiento, con zoom para comprobar que no hay C. elegans

Este algoritmo se ha implementado en Matlab y se han obtenido los experimentos sin *C. elegans*.

5.5.2.3 Simulador

Con las imágenes de las placas de Petri generadas y el algoritmo de generación de trayectorias disponible, el siguiente paso es definir el algoritmo que genera las curvas de *Lifespan*, es decir, el número de *C. elegans* vivos en cada día. A continuación, se proponen dos métodos.

Generación de curvas de *Lifespan* utilizando los datos originales

Una posibilidad es hacer que el número de *C. elegans* vivos generados en cada día sea el mismo que el de las imágenes originales etiquetadas, pero utilizando un filtro de postprocesado (Puchalt et al., 2020) del conteo de forma que:

- El número de *C. elegans* contados el primer día es el máximo.
- Si en un día posterior hay más *C. elegans* etiquetados, se cambia al valor del primer día.
- Si en un día el número de *C. elegans* es superior al de algún día anterior, el número de *C. elegans* en ese día se modifica por el conteo del día actual.

Generación de curvas de *Lifespan* utilizando la fórmula de Weibull

En el anterior método de generación de curvas, se observa que los datos son muy parecidos, en cuanto al número de *C. elegans* y duración de los experimentos. Esto puede ser una posible causa de que la red acabe prediciendo siempre unos valores similares y no sea capaz de generalizar.

Para solucionar el inconveniente del método anterior se ha propuesto utilizar un método que permita generar curvas de *Lifespan* con diferentes duraciones de experimentos, con el fin de obtener una mayor variedad de datos que permita que la red neuronal aprenda a generalizar.

El modelo de *Lifespan* de los *C. elegans* ha sido ampliamente estudiado por la comunidad científica y existen diversos modelos y fórmulas matemáticas que permiten generar curvas.

En (Vanfleteren et al., 1998) se analizan cuatro modelos matemáticos de la curva de *Lifespan* para *C. elegans*:

- Gompertz
- Weibull
- 2p logistic
- 3p logistic

De estos métodos, el más interesante para el simulador es el de Weibull, ya que permite calcular el porcentaje de *C.elegans* vivos en el día del experimento t, utilizando como entrada 2 parámetros (a, b), que están relacionados con la pendiente de la curva de mortalidad y con la vida media respectivamente.

La función de mortalidad de Weibull es la siguiente:

$$m(t) = \left(\frac{a}{b^a}\right) \cdot t^{a-1}$$

Ecuación 4 Función de mortalidad de Weibull

Y la función de supervivencia:

$$\%\ supervivencia = 100 \cdot e^{-(\frac{t}{b})^a}$$

Ecuación 5 Función de supervivencia de Weibull

Donde t es el tiempo en días y los parámetros a y b son valores que determinan la pendiente de la curva de mortalidad y la vida media respectivamente.

Está función de probabilidad se ha utilizado para crear una función que recibe como parámetros el día en el que empieza el experimento, el número de *C. elegans* a simular y los parámetros a y b.

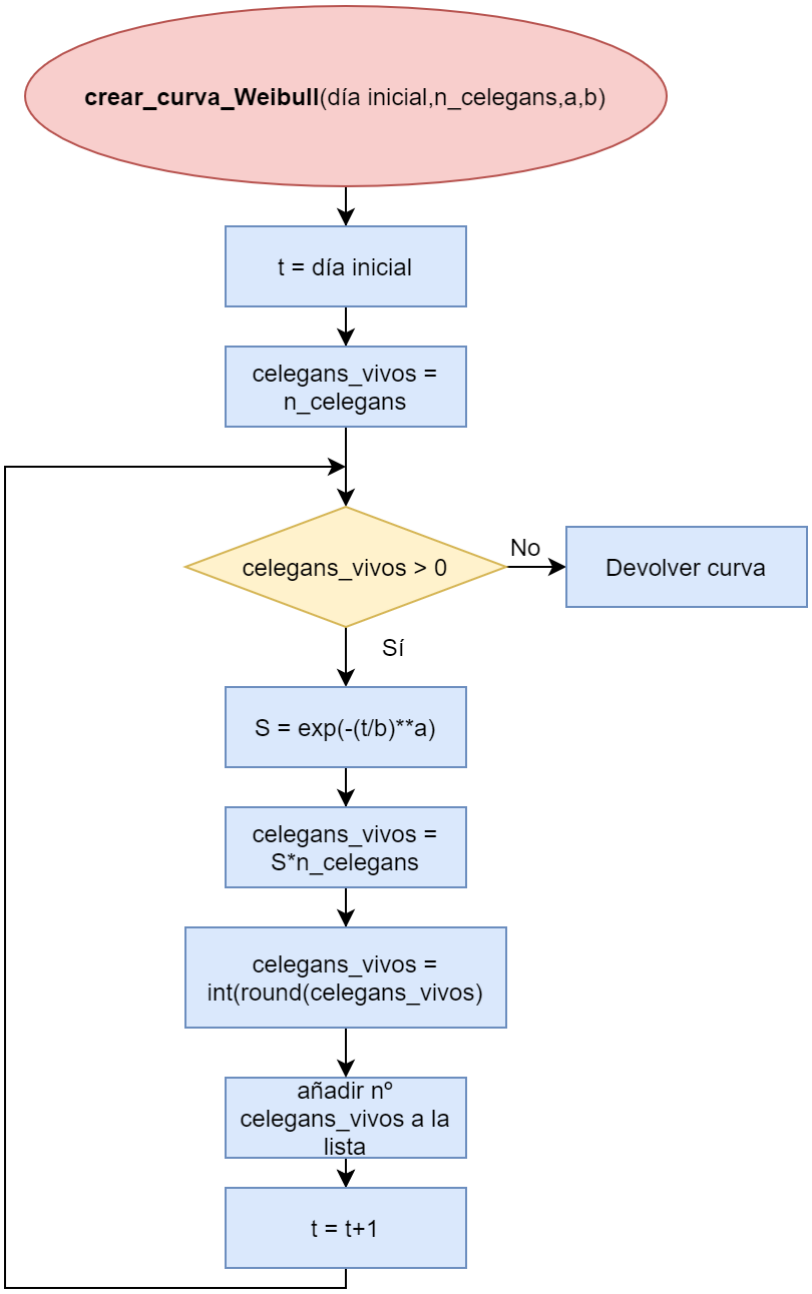

Fig. 45 Flujograma función generar curva Lifespan Weibull

En primer lugar, se iguala el número de *C. elegans* vivos al número de *C. elegans* del experimento y se iguala el tiempo al día inicial del experimento.

Mientras el número de *C. elegans* vivos sea mayor que cero, se calcula la probabilidad de supervivencia aplicando la fórmula de Weibull. Ese valor de probabilidad se multiplica por el número de C. elegans vivos al comienzo del día actual para calcular el número de *C. elegans* que quedan vivos tras ese día. Como el resultado es número decimal, se redondea a número entero. Ese valor se almacena en una lista que contiene los *C. elegans* vivos en cada día.

Cuando termina el experimento, se devuelve la lista que contiene el número de *C. elegans* vivos en cada día, que será la entrada del algoritmo simulador de trayectorias junto con las imágenes de las placas sin *C. elegans* para generar los nuevos experimentos.

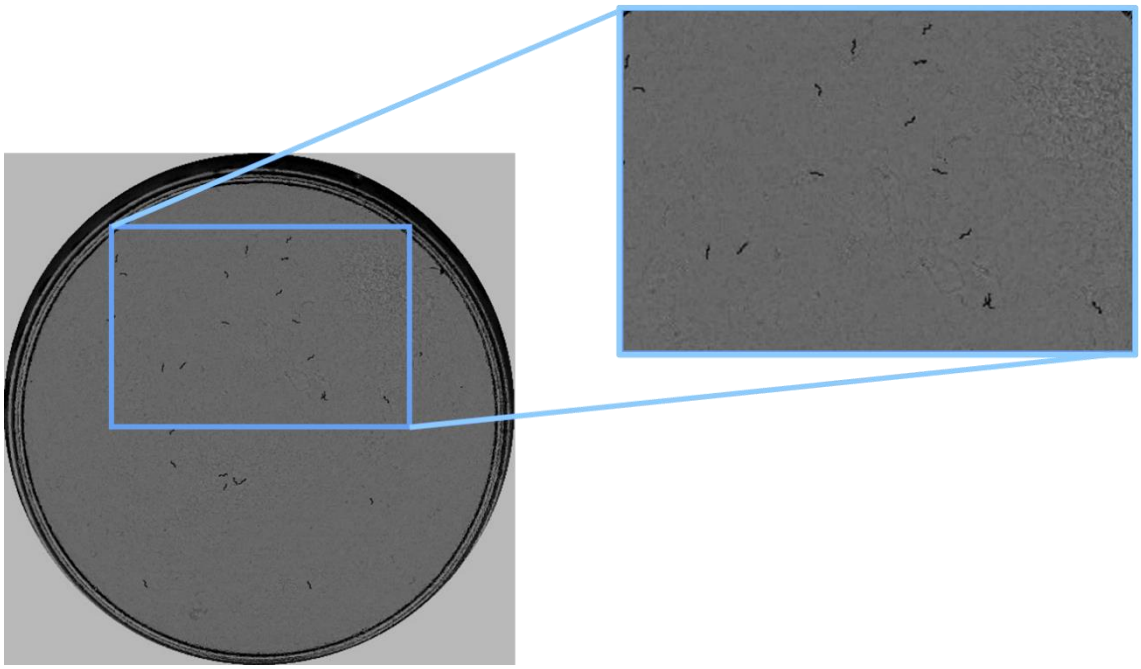

Fig. 46 Ejemplo de imagen generada con el simulador

En la siguiente figura se puede ver el aspecto de los *C. elegans* simulados. Existe una pérdida de resolución debido a que las imágenes generadas son de tamaño 1944x1944 píxeles y son reducidas a un tamaño de 524x524 ya que es el tamaño de entrada a la red tal y como se explica en el apartado 5.6.

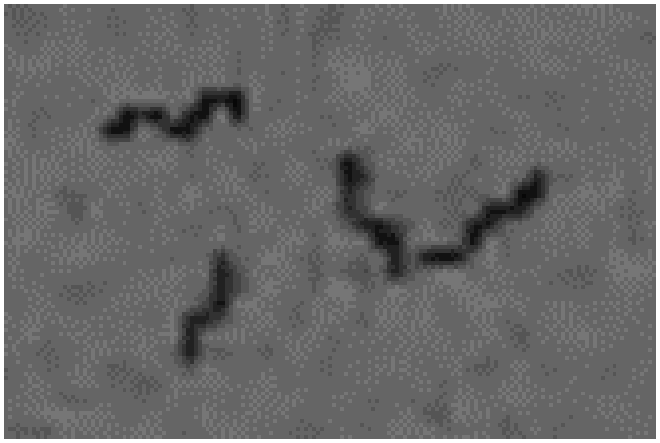

Fig. 47 Zoom C. elegans generados con el simulador

5.6 Implementación y entrenamiento

División de las imágenes en entrenamiento y validación

Para poder medir la bondad del modelo que se va a entrenar, es necesario evaluarlo con datos que no se han utilizado para el entrenamiento. El algoritmo puede ajustarse a los datos de entrenamiento y obtener buenos resultados de precisión, pero en realidad no ser capaz de generalizar. Por este motivo, el *dataset* completo se ha dividido en dos conjuntos, de entrenamiento (70%) y de validación (30%).

Definición y elección de hiperparámetros

- **Épocas**. Es el número de veces que el algoritmo recorre el conjunto completo de muestras de entrenamiento.
- ***Batch size*** (tamaño de lote). Es un hiperparámetro que hace referencia al número de muestras empleadas en cada iteración de aprendizaje, es decir el número de muestras que pasan por la red antes de realizar la actualización de los pesos. Los valores típicos de *batch size* suelen ser 32, 64 y 128. En nuestro caso se ha utilizado un *batch size* de 4, ya que es lo máximo que permitía la memoria disponible. Este valor es pequeño ya que los *batches* son secuencias de más de 30 imágenes y la memoria disponible no permite un valor superior.
- ***Learning rate*** (tasa de aprendizaje). Este hiperparámetro es un escalar que define cuanto afecta el gradiente a la actualización de los pesos de la red en cada iteración del entrenamiento. Este valor varía según el problema, pero en general, elegir un valor muy pequeño puede hacer que el aprendizaje sea muy lento y elegir un valor muy elevado puede hacer que el algoritmo no alcance el mínimo. Este efecto de la elección del *learning rate* se puede observar en la figura 48.
  En nuestro caso se ha empleado un *learning rate* de 0,01 y se ha ido modificado cada cierto número de épocas multiplicando por un factor de 0,1.

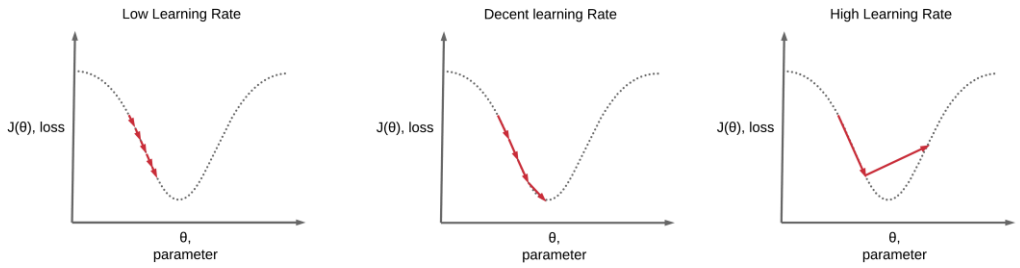

*Fig. 48 Efecto elección de learning rate (*Learning Rate Scheduling - Deep Learning Wizard*, s. f.)*

Elección de optimizador

Es el algoritmo de optimización empleado para calcular los pesos que permitan minimizar la función de coste.

El empleado ha sido el del descenso del gradiente utilizando *mini-batches*, es decir, que los parámetros se actualizan tras calcular el error de cada *mini-batch*.

Función de coste y métrica de error

La función de coste permite medir el error de las predicciones de la red neuronal respecto de las etiquetas de las entradas. En este caso las predicciones son secuencias de valores (la curva de *Lifespan*), por lo que se puede considerar como un problema de regresión lineal.

La función de coste elegida ha sido el error cuadrático medio (*MSE loss*: *mean squared error*), su fórmula es la siguiente:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f(x) - y)^2$$

Ecuación 6 Fórmula error cuadrático medio MSE

Donde *N* es el número de muestras*, f(x)* es la predicción de la red neuronal e *y* es el valor real o etiqueta.

Como métrica del error durante la fase de entrenamiento se ha empleado la raíz del error cuadrático medio (RMSE) que permite tener una medida con una interpretación más intuitiva del error cometido. Su fórmula es:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x) - y)^2}$$

Ecuación 7 Fórmula RMSE

Preprocesamiento

Uno de los objetivos de este TFM es intentar que el preprocesamiento de las imágenes sea mínimo, buscando una metodología *end to end*. El preprocesamiento que se ha realizado es cambiar el tamaño de las imágenes, que originalmente tienen un tamaño de 1944x1944 píxeles, reduciéndolo a un tamaño de 524x524 píxeles. Este tamaño se ha elegido teniendo en cuenta el máximo tamaño que la GPU es capaz de procesar y que además se pueda distinguir el cuerpo de los *C. elegans* presentes en la placa de Petri.

También se normaliza la imagen entre valores de 0 a 255 en escala de grises y se hace la operación lógica *not* utilizando la función de OpenCV *cv2.bitwise_not(img, img)*. Con esto, se consigue aumentar el contraste entre los valores de fondo y los valores de los *C. elegans*. En las imágenes originales el fondo tiene un valor de intensidad en torno a 50 y los *C. elegans* de 0 a 10, con este cambio (normalización y negada) el fondo pasa a ser de 140 y los *C. elegans* de valores superiores a 200. Aunque esta normalización no afecta al aprendizaje de la red, se ha realizado para permitir visualizar mejor las imágenes en caso de querer analizarlas.

En la figura 49 se pueden ver los pasos del preprocesamiento:

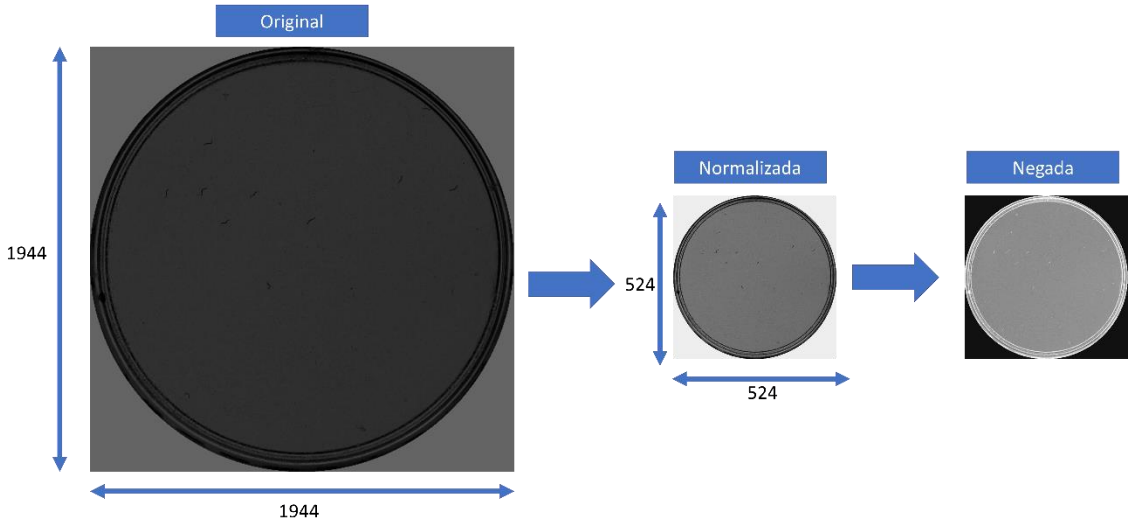

Fig. 49 Preprocesamiento imágenes de entrada a la red

En la siguiente figura se puede ver la imagen resultante del preprocesamiento y que será la entrada a la red neuronal:

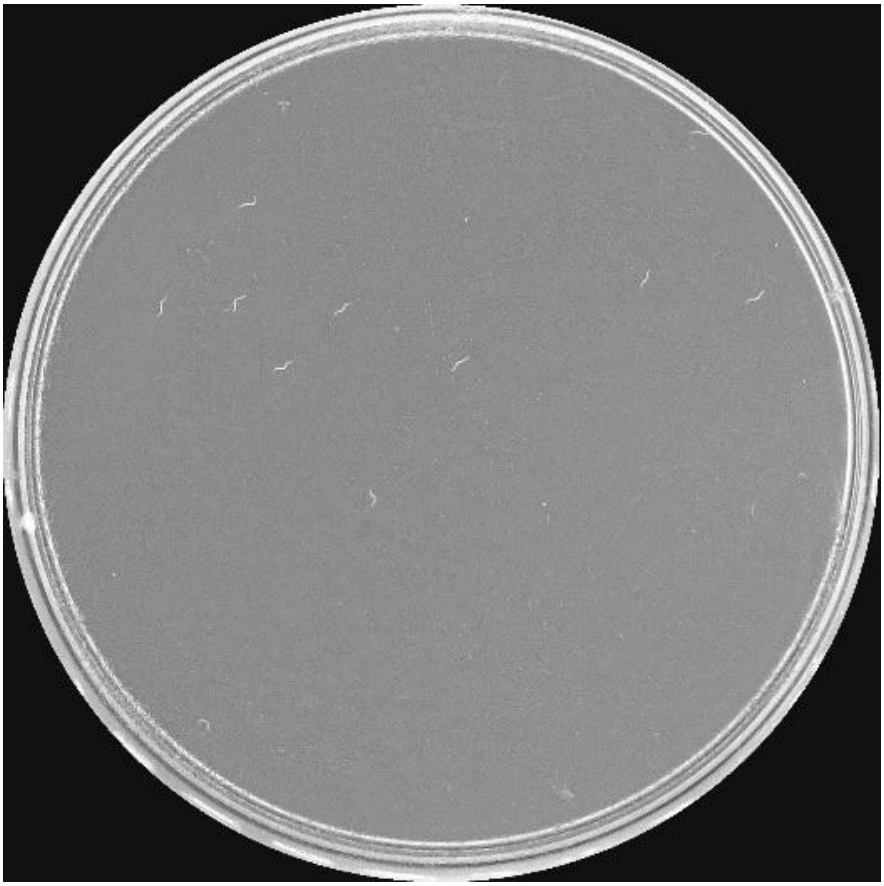

Fig. 50 Imagen resultado del preprocesamiento

La implementación se ha realizado empleando la plataforma de *Deep Learning* Pytorch. Esta plataforma ha sido desarrollada por el grupo de inteligencia artificial de Facebook. Se ha elegido esta opción, ya que presenta una buena documentación con gran cantidad de ejemplos y su uso en la comunidad dedicada a la investigación está creciendo mucho en los últimos años. Además, dispone de soporte para su ejecución mediante GPU utilizando la API CUDA de NVIDIA.

En el siguiente diagrama se resumen los pasos seguidos para la implementación del código:

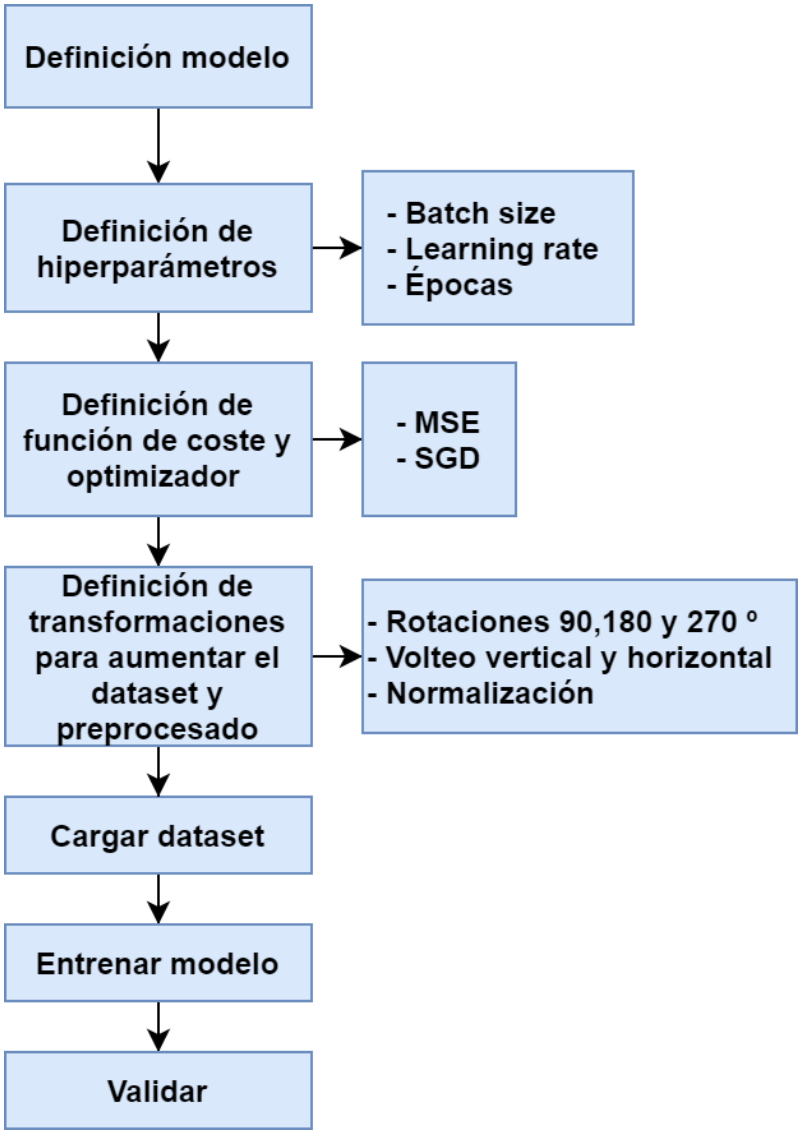

Fig. 51 Esquema implementación y entrenamiento red neuronal

6 Resultados

En este apartado se muestran diferentes experimentos realizados y los resultados obtenidos al evaluar los modelos entrenados.

6.1 Elección de arquitectura y método de introducción de imágenes

El primer paso ha sido elegir la combinación de arquitectura y método de introducción de imágenes más apropiado de todas las alternativas que se han propuesto.

Tras evaluar las ventajas e inconvenientes de cada alternativa y de realizar pruebas para comprobar qué método obtiene mejores resultados se han obtenido las siguientes conclusiones:

- Los métodos que predicen el número de *C. elegans* vivos en un día (apartado 5.4.2) no han sido capaces de generalizar y a la salida siempre devuelven valores similares, la red no es capaz de diferenciar entre las distintas secuencias.
- La arquitectura LRCN que utiliza una red convolucional preentrenada devuelve a la salida valores similares tanto para los métodos de un solo día como el método que utiliza imágenes de todos los días.
- Al no disponer de suficiente memoria en la red LRCN, solo se ha podido evaluar la red preentrenada resnet18, en lugar de la red resnet50 que es la que se empleaba originalmente en el artículo.

Por estos motivos se ha decidido emplear la siguiente combinación de arquitectura CNN-LSTM-FC y el método que utiliza una imagen de cada día del experimento, que sí consiguen obtener salidas diferentes para las distintas entradas.

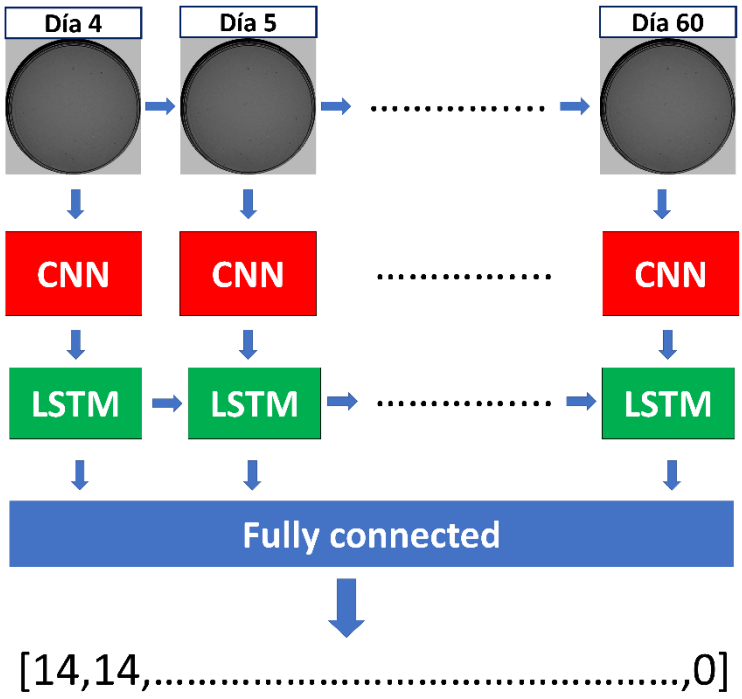

Fig. 52 Esquema arquitectura CNN-LSTM-FC

Una vez elegido esta combinación de arquitectura y método de introducción de las imágenes, a continuación, se presentan los resultados obtenidos con este método.

6.2 Modelo entrenado con imágenes originales

El *dataset* empleado consta de los siguientes experimentos:

Experimento	Placas /cond	Condiciones	Días	Imágenes/día	Total
Lifespan 13V1	10	2	20	30	12000
Lifespan 13V2	10	2	20	30	12000
Lifespan 14V1	10	2	20	30	12000
Lifespan 14V2	10	2	20	30	12000
Lifespan 18	4	4	18	30	8640
LS ensayo1	4	2	27	30	6480
Lifespan Daf2	4	7	39	30	32760
				Total	95880

Tabla 3 Información dataset original disponible

Tal y como se ha comentado en el apartado anterior, los huecos de los días festivos se completan utilizando imágenes ya existentes. Además, para que todas las secuencias de entrada sean de la misma longitud, la imagen del último día se replica hasta alcanzar el valor de secuencia máxima (en este caso es de 57 días). Por tanto, el número de imágenes empleadas para el entrenamiento y validación es el siguiente:

Experimento	Placas /cond	Condiciones	Total placas	Días	Imágenes/día	Total imágenes	Total secuencias
Lifespan 13V1	10	2	20	57	30	34200	600
Lifespan 13V2	10	2	20	57	30	34200	600
Lifespan 14V1	10	2	20	57	30	34200	600
Lifespan 14V2	10	2	20	57	30	34200	600
Lifespan 18	4	4	16	57	30	27360	480
LS ensayo1	4	2	8	57	30	13680	240
Lifespan Daf2	4	7	28	57	30	47880	840
Total			132			225720	3960

Tabla 4 Información dataset aumentado completando huecos

En la tabla 4 se muestra el número total de placas disponibles (132). Como ya se ha comentado en el apartado 5.4.1, las secuencias están formadas por una imagen de cada día del experimento, por tanto, por cada placa se pueden utilizar 30 secuencias. Esta práctica no es la más apropiada, ya que existe poca diferencia entre las imágenes del

mismo día (sobre todo en los días finales), y esto puede dar lugar a *overfiting*. Sin embargo, al disponer de un conjunto de imágenes tan reducido se ha optado por hacerlo de esa manera.

De este *dataset*, el 70% se ha empleado para el entrenamiento y el 30% para validar. Se ha empleado las técnicas de aumento de datos comentadas en el apartado anterior (rotaciones de 90, 180 y 270 grados, y volteos vertical y horizontal). Tras realizar varios entrenamientos y observar que el modelo deja de mejorar se ha obtenido un RMSE de 1,5 en el conjunto de validación.

Para analizar mejor los resultados y poder comparar con los resultados del método que emplea técnicas tradicionales de visión por computador se ha calculado el error medio absoluto (MAE) de los porcentajes de *C. elegans* vivos.

El porcentaje de *C. elegans* vivos en cada día del experimento se calcula de la siguiente manera:

$$\%\ C.elegans\ vivos = \frac{C.elegans\ vivos\ día\ actual}{C.elegans\ vivos\ \ primer\ día} \cdot 100$$

Ecuación 8 Cálculo % C. elegans vivos

Una vez calculados los porcentajes, el MAE se calcula obteniendo el promedio del error en cada uno de los días, siendo el error la diferencia entre el porcentaje de *C. elegans* vivos del modelo y el porcentaje de *C. elegans* vivos reales etiquetados:

$$MAE = promedio(|\%vivos_{modelo} - \%vivos_{real}|)$$

Ecuación 9 Cálculo del MAE de los porcentajes

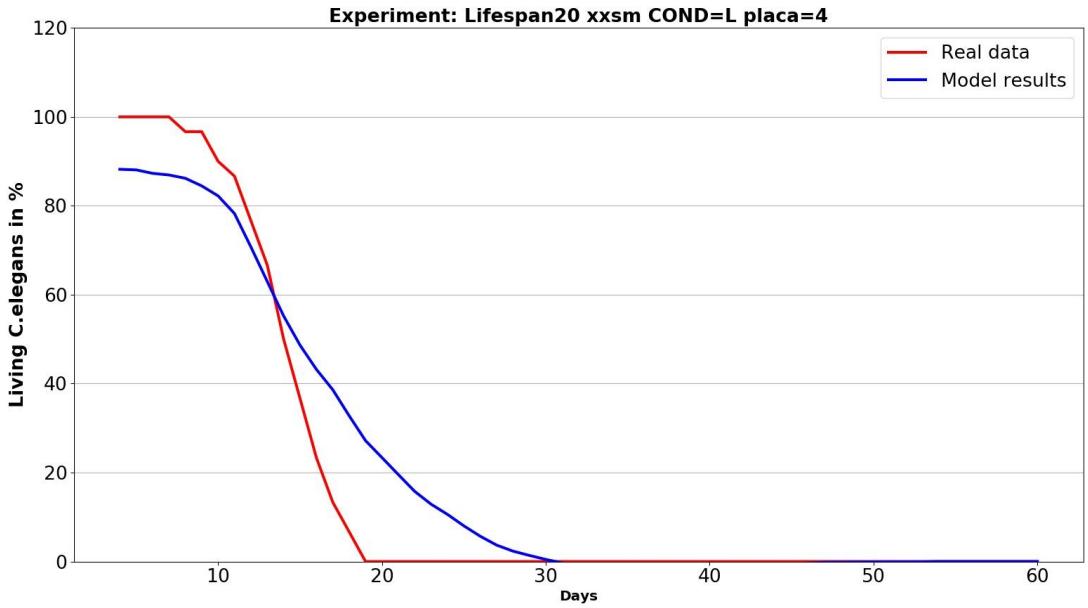

Fig. 53 Gráficas de curvas de Lifespan modelo y real

En la siguiente tabla se muestran los resultados obtenidos en cada una de las placas, incluyendo también el valor del MAE en termino de número de *C. elegans:*

Nombres de los experimentos	MAE de los %:	MAE
Lifespan14 V1 COND=B placa=3	9,65 ± 7,53%	1,06 ± 0,83
Lifespan13 V1 COND=A placa=2	5,76 ± 8,28%	0,58 ± 0,83
Lifespan14 V2 COND=B placa=2	6,31 ± 5,08%	0,69 ± 0,56
Lifespan14 V2 COND=B placa=6	9,34 ± 8,31%	1,12 ± 1,0
Lifespan13 V2 COND=A placa=2	7,61 ± 8,94%	0,76 ± 0,89
Lifespan14 V2 COND=B placa=8	9,63 ± 9,56%	1,06 ± 1,05
Lifespan13 V1 COND=A placa=6	5,14 ± 6,21%	0,51 ± 0,62
Lifespan13 V1 COND=A placa=4	5,61 ± 6,13%	0,56 ± 0,61
Lifespan18 COND=D placa=3	7,09 ± 8,19%	0,92 ± 1,06
LS ensayo1 COND=B placa=2	15,02 ± 11,27%	1,95 ± 1,47
Lifespan14 V1 COND=B placa=9	6,15 ± 4,03%	0,61 ± 0,4
lifespan daf2 y bebidas Ens2.0 COND=G placa=4	7,86 ± 6,29%	0,94 ± 0,75
lifespan daf2 y bebidas Ens2.0 COND=D placa=2	23,93 ± 11,32%	3,59 ± 1,7
Lifespan18 COND=B placa=3	8,18 ± 9,78%	1,06 ± 1,27
Lifespan18 COND=B placa=4	6,97 ± 7,09%	0,7 ± 0,71
Lifespan13 V2 COND=B placa=1	5,53 ± 6,05%	0,55 ± 0,6
Lifespan13 V2 COND=A placa=6	6,23 ± 6,77%	0,62 ± 0,68
Lifespan18 COND=A placa=2	8,01 ± 10,09%	1,12 ± 1,41
Lifespan13 V2 COND=A placa=9	8,15 ± 5,84%	0,9 ± 0,64
lifespan daf2 y bebidas Ens2.0 COND=G placa=2	5,81 ± 4,58%	0,7 ± 0,55
Lifespan14 V2 COND=B placa=3	8,76 ± 10,18%	0,96 ± 1,12
Lifespan14 V2 COND=A placa=6	6,62 ± 7,22%	0,73 ± 0,79
lifespan daf2 y bebidas Ens2.0 COND=F placa=2	25,33 ± 6,19%	3,55 ± 0,87
Lifespan13 V2 COND=B placa=7	6,21 ± 8,37%	0,62 ± 0,84
Lifespan18 COND=D placa=4	8,43 ± 10,55%	1,01 ± 1,27
Lifespan18 COND=A placa=1	9,24 ± 12,0%	1,2 ± 1,56
Lifespan13 V2 COND=B placa=4	5,15 ± 5,67%	0,51 ± 0,57
Lifespan14 V2 COND=B placa=5	5,96 ± 8,32%	0,66 ± 0,91
Lifespan13 V2 COND=B placa=8	5,81 ± 5,86%	0,58 ± 0,59
lifespan daf2 y bebidas Ens2.0 COND=A placa=1	11,72 ± 7,35%	1,41 ± 0,88
lifespan daf2 y bebidas Ens2.0 COND=C placa=1	11,29 ± 5,99%	1,58 ± 0,84
Lifespan13 V1 COND=A placa=7	5,45 ± 6,28%	0,54 ± 0,63
Lifespan18 COND=A placa=4	12,59 ± 14,04%	1,38 ± 1,54
Lifespan13 V1 COND=B placa=10	4,29 ± 5,09%	0,47 ± 0,56
Lifespan13 V2 COND=B placa=2	6,67 ± 4,82%	0,67 ± 0,48
Lifespan14 V1 COND=A placa=7	10,14 ± 9,53%	1,12 ± 1,05
Lifespan14 V1 COND=A placa=9	7,23 ± 9,15%	0,72 ± 0,91
Lifespan13 V1 COND=B placa=9	5,23 ± 6,5%	0,47 ± 0,58
	MAE de los %: 8,53 ± 4,43%	**MAE : 1,01 ± 0,69**

Tabla 5 Errores por placas experimento modelo entrenado con imágenes originales

Estos resultados obtenidos (8,53 ± 4,43%) se encuentran lejos de los obtenidos por el sistema de tradicional de visión por computador (2,24 ± 0,55%).

Análisis de causas de error

Como se ha mostrado en la tabla 4, el número de placas disponible es muy reducido (132). El método de introducción de imágenes utiliza secuencias con una imagen de cada día, y por cada día hay 30 imágenes, entonces, el total de secuencias pasa a ser de 3960. A pesar de este incremento, al ser tan similares puede que se produzca un sobreentrenamiento que impida a la red generalizar.

Además, los experimentos disponibles en la base de datos son muy parecidos en cuanto a la duración (días desde que se comienza a contar hasta que muere el último *C. elegans*) y al número de *C. elegans* utilizados en los ensayos. Esta variabilidad puede que no sea suficiente para hacer que la red sea capaz de generalizar.

$$duración = día\_último\_celegans - día\_inicio\_conteo$$

Ecuación 10 Duración experimentos

En la figura 54 se puede apreciar como el *dataset* presenta un gran desequilibrio en la distribución de muestras de las distintas duraciones de los experimentos. Balancear sería una opción, pero al haber tan pocos experimentos no sería posible tener muestras tanto en entrenamiento como validación de todas las duraciones.

Fig. 54 Duración experimentos dataset original

Además, el *dataset* original presenta imágenes que están muy contaminadas o con problemas de adquisición.

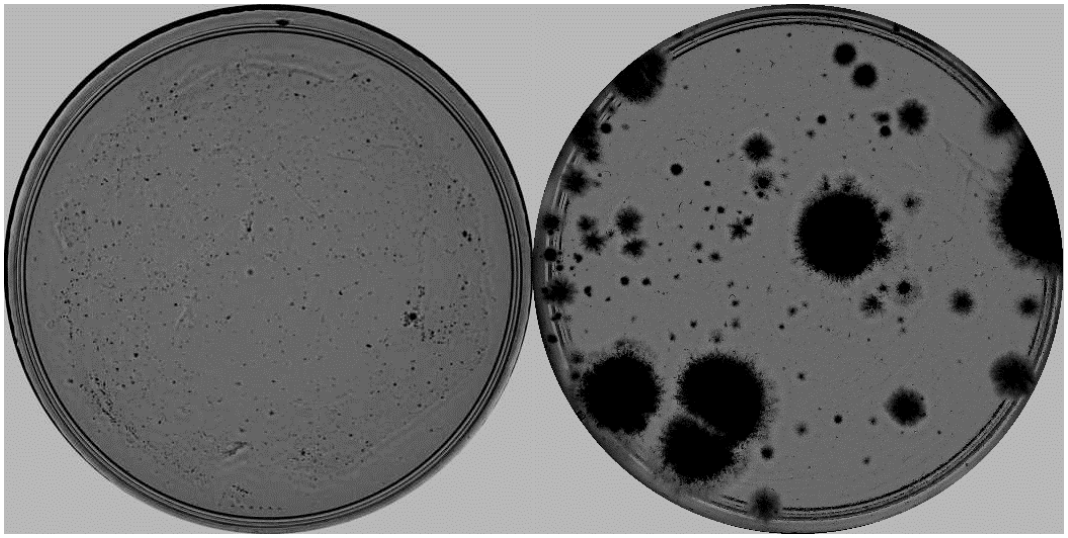

Fig. 55 Ejemplos imágenes contaminadas

La reducción del tamaño de la imagen original, que es de 1944x1944 píxeles a un tamaño de 524x524 píxeles hace que se pierda resolución y en algunos casos puede complicar mucho la tarea dificultando la visualización de los *C. elegans*.

6.3 Modelo entrenado con imágenes simuladas

En este experimento se ha entrenado un modelo utilizando imágenes simuladas para posteriormente validarlo con el *dataset* de imágenes originales. El *dataset* simulado contiene experimentos con distintas duraciones que varían entre 7 y 50 días con la misma frecuencia para todos, es decir, está balanceado en base a las duraciones de los experimentos.

Además de la variabilidad en el número de días, también se ha incorporado variabilidad en el número de *C. elegans* siendo la mitad de los experimentos de 10 y la otra mitad de 30.

El código del simulador requiere de mucho tiempo para generar muestras, por lo que no se han podido obtener muchas. Finalmente se han simulado 210 placas de ensayo.

Total duraciones distintas	Total muestras	Total secuencias
35	210	6300

Tabla 6 Información datos simulados

Con estos datos se ha entrenado un modelo desde cero y se ha validado tanto con las imágenes simuladas como originales.

Validación de imágenes simuladas utilizando modelo entrenado con imágenes simuladas

MAE del % de *C. elegans* vivos	11,31 ± 4,64%
MAE del nº de *C. elegans* vivos	2,18 ± 1,36

Tabla 7 Resultados validación con imágenes simuladas del modelo entrenado con imágenes simuladas

El error obtenido es superior al del modelo entrenado con imágenes originales, pero hay que tener en cuenta que este *dataset* presenta una mayor variabilidad tanto en la duración de los experimentos como en el número de *C. elegans*, por lo que el hecho de que el error sea mayor no significa que generalice peor.

Validación de imágenes reales utilizando modelo entrenado con imágenes simuladas

MAE del % de *C. elegans* vivos	47,48 ± 27,45%
MAE del nº de *C. elegans* vivos	5,14 ± 2,65

Tabla 8 Resultados validación con imágenes reales del modelo entrenado con imágenes simuladas

Al validar utilizando las imágenes reales se observa que este modelo entrenado con datos simulados no funciona con los datos originales. Esto se puede deber a los siguientes problemas que presentan las imágenes simuladas:

- No aparecen *C. elegans* muertos, y en las imágenes reales sí los hay. Es decir, con este simulador solo se generan *C. elegans* vivos y no muertos, cuando un *C.elegans* muere desaparece de la placa.
- No hay continuidad en los días, las secuencias de cada día son independientes. El simulador calcula y dibuja las trayectorias de un número de *C. elegans* en 30 imágenes sin tener en cuenta las posiciones de los días anteriores. Para los primeros días, esto no afecta mucho, ya que en la realidad los *C. elegans* se mueven más y cambian mucho su posición de un día a otro. Sin embargo, en los últimos días se mueven muy poco y si existe continuidad entre días.

6.4 Modelo entrenado con imágenes originales y simuladas

En este experimento se ha confeccionado un nuevo conjunto de entrenamiento mezclando tanto imágenes reales como simuladas. Se ha balanceado por duraciones de forma similar al caso anterior. La idea de este experimento es comprobar si la red es capaz de mezclar el aprendizaje de ambos dataset y generalizar mejor.

Total duraciones distintas	**Total muestras**	**Total secuencias**
24	144	4320

Tabla 9 Información dataset mezclado entrenamiento

Los resultados obtenidos son los siguientes:

Validación con imágenes reales y simuladas:

MAE del % de *C. elegans* vivos	14,5 ± 6,29%
MAE del nº de *C. elegans* vivos	2,22 ± 1,27

Tabla 10 Resultados validación con imágenes mezcladas del modelo entrenado con imágenes mezcladas

Validación con imágenes reales:

MAE del % de *C. elegans* vivos	15,5 ± 6,92%
MAE del nº de *C. elegans* vivos	1,66 ± 0,76

Tabla 11 Resultados validación con imágenes reales del modelo entrenado con imágenes mezcladas

Estos resultados muestran que entrenando un modelo mezclando imágenes reales y simuladas se mejora al validar con respecto al experimento anterior (modelo entrenado con imágenes simuladas) pero no respecto al primero que entrena solo con imágenes reales (8,53 ± 4,43%). Con este experimento se confirma que el dataset simulado no ayuda a mejorar la precisión del algoritmo. Esto quiere decir que las imágenes simuladas no son suficientemente representativas de los datos reales, por tanto, se debe modificar el simulador desarrollado para que los experimentos generados sean útiles para el entrenamiento.

7 Presupuesto

En este apartado se realiza una estimación económica de los costes de la realización de este TFM. Para la estimación del número de horas empleadas se ha tenido en cuenta la carga académica del TFM, que es de 15 créditos ECTS. Cada crédito equivale a 25-30 horas de formación, por tanto, el número de horas aproximado para el cálculo es de 450 horas.

A continuación, se realiza el desglose del presupuesto en dos partes: los costes de mano de obra y los costes de equipamiento (hardware y software) y, por último, se calcula el presupuesto total.

7.1 Costes de mano de obra

En el cálculo de los costes de la mano de obra se ha tenido en cuenta la labor del alumno, graduado en ingeniería electrónica industrial y automática cuyo salario aproximado es de 20 €/hora. Por otro lado, también se ha incluido la labor de dirección y supervisión del tutor, cuyo salario aproximado es de 40 €/hora.

MANO DE OBRA				
Ud	**Descripción**	**Precio(€/h)**	**Cantidad(h)**	**Total(€)**
h	Alumno ingeniero en electrónica industrial y automática	20	450	9000
h	Tutor	40	20	800
SUBTOTAL MANO DE OBRA			**470**	**9800**

Tabla 12 Costes de mano de obra

7.2 Costes de equipamiento

En cuanto al hardware, el equipamiento está formado por el ordenador del laboratorio y el ordenador personal descritos en el apartado 5.1 del proyecto.

Los equipos empleados no se han adquirido para este proyecto en concreto. Por tanto, se ha calculado su amortización durante el tiempo que se ha desarrollado el proyecto multiplicando el precio por el periodo de uso y dividiendo entre la vida útil.

COSTES DE HARDWARE						
Ud	**Descripción**	**Cantidad**	**Vida útil (años)**	**Periodo de uso (años)**	**Precio (€)**	**Coste (€)**
ud	PC lab	1	3	0,6	1500	300
ud	GPU lab	1	3	0,6	1000	200
ud	PC personal	1	3	0,6	500	100
SUBTOTAL COSTES DE HARWARE						**600**

Tabla 13 Costes de hardware

En cuanto al software, la mayoría de los programas empleados son de licencia libre, por lo que no han tenido coste alguno. Tan solo se ha tenido en cuenta el coste de la licencia de Windows 10 y de Matlab.

COSTES DE SOFTWARE						
Ud	**Descripción**	**Cantidad**	**Vida útil (años)**	**Periodo de uso (años)**	**Precio (€)**	**Coste (€)**
ud	Windows 10 home	1	3	0,6	145	29
ud	Matlab R2019b	1	3	0,6	70	14
ud	Pycharm	1	3	0,6	0	0
ud	Python 3.7	1	3	0,6	0	0
ud	Librerías Pytorch, CUDA, OpenCV	1	3	0,6	0	0
SUBTOTAL COSTES DE SOFTWARE						**43**

Tabla 14 Costes de software

7.3 Presupuesto total

En este apartado se recogen los gastos totales del proyecto. En primer lugar, se suman los subtotales de los costes de mano de obra y de equipamiento, obteniendo así los gastos de ejecución material. A este valor, se le aplican los porcentajes correspondientes a gastos generales (13%) y beneficio industrial (6%), y se suman obteniendo el presupuesto de ejecución por contrata. Por último, se aplica el IVA (21%) y se suma obteniendo el presupuesto total del proyecto.

COSTE DEL PROYECTO	
Concepto	**Importe (€)**
Mano de obra	9800
Costes de equipamiento	643
Presupuesto de ejecución material	10443
Gastos generales (13%)	1357,59
Beneficio industrial (6%)	6265,8
Presupuesto de ejecución por contrata	18066,39
IVA (21%)	3793,94
TOTAL	**21860,33**

Tabla 15 Desglose presupuesto total

El coste total del proyecto asciende a **VEINTIÚN MIL OCHOCIENTOS SESENTA EUROS CON TREINTA Y TRES CÉNTIMOS.**

8 Conclusiones y mejoras futuras

8.1 Conclusiones

El objetivo principal de este trabajo fin de máster era el de investigar el uso de un algoritmo basado en técnicas de aprendizaje profundo que permitiese la automatización de ensayos *Lifespan* con nematodos *C. elegans*.

Revisando el estado del arte en el uso de algoritmos de aprendizaje profundo para la automatización de experimentos con *C. elegans*, se han encontrado diferentes proyectos, pero ningún precedente a la hora de resolver el problema del conteo automático de *C. elegans* vivos, que es el problema que se intenta abordar en este trabajo.

El primer paso para resolver el problema ha sido el de confeccionar el *dataset*, etiquetando las imágenes de los ensayos disponibles con el número de *C. elegans* vivos en cada día.

En segundo lugar, se han planteado distintas arquitecturas de redes neuronales que combinan redes convolucionales y redes recurrentes LSTM, ya que han demostrado ser efectivas a la hora de captar tanto relaciones espaciales como temporales.

Una vez definidas las arquitecturas, se han planteado métodos para introducir la secuencia de imágenes a la red. Por un lado, los métodos que realizan el conteo de *C. elegans* vivos en un día concreto y por otro, los que realizan el procesamiento completo del experimento de *Lifespan.*

Seguidamente se ha analizado las ventajas e inconvenientes de cada arquitectura y distintas formas de introducción de imágenes, eligiendo finalmente la red CNN-LSTM-FC y el método de procesamiento completo que utiliza una imagen de cada día del experimento.

Con este método se han realizado distintas pruebas encontrando como obstáculo la poca cantidad de datos y la poca variabilidad de los datos disponibles.

Para intentar mejorar los resultados, se han utilizado técnicas de aumento de datos transformando las imágenes originales (rotaciones y volteos) y, además, se ha implementado un simulador que ha permitido incrementar el *dataset* con nuevos experimentos.

Sin embargo, los experimentos realizados han mostrado que el incremento de datos a partir de las imágenes reales no es suficiente, ya que no resuelve el problema del desequilibrio en la distribución de muestras respecto de la duración de los experimentos.

Los datos generados a partir del simulador tampoco han permitido mejorar, ya que, aunque sí que incorporan la variabilidad de las duraciones, las imágenes generadas no eran lo suficientemente similares a las reales.

Por tanto, se ha cumplido con el objetivo de diseñar, desarrollar y evaluar algoritmos basados en aprendizaje profundo para la automatización del conteo de *C. elegans* vivos, sin embargo, los problemas anteriormente comentados han impedido conseguir una solución con mayor precisión que el algoritmo tradicional de visión por computador.

8.2 Mejoras futuras

En base a la investigación y experimentos realizados en este trabajo fin de máster, y los resultados obtenidos se plantean algunas posibles mejoras.

Uno de los grandes problemas, es conseguir una base de datos lo suficientemente grande y variada para que la red sea capaz de generalizar. En este TFM se ha intentado resolver este problema utilizando un simulador, sin embargo, este simulador solo genera imágenes con *C. elegans* vivos y no existe continuidad entre las imágenes por lo que puede que no sea apropiado.

Por tanto, una mejora posible es modificar el simulador para que sea capaz de generar experimentos más parecidos a los casos que se dan en la realidad. Esta modificación consistiría en hacer que además de simular *C. elegans* vivos también se incluyeran muertos y que las secuencias de imágenes de cada día no sean independientes, sino que se generen trayectorias tomando como referencia las posiciones del día anterior.

Otro inconveniente ha sido el elevado tiempo necesario para la generación de imágenes con el simulador, por tanto, este código también debe mejorarse en términos de tiempo de computación para poder generar más experimentos en menos tiempo.

En algunos casos, la contaminación junto con la pérdida de resolución de las imágenes al reducir el tamaño hace que se pierda información. Una mejora sería utilizar alguna técnica de preprocesado que permita mejorar el método utilizado.

Si con estos aumentos de datos y mejoras del *dataset* no se consiguiera mejorar los resultados obtenidos, otra opción sería plantear otras arquitecturas de redes neuronales (redes convolucionales preentrenadas, GRU y los Transformers) y evaluarlas.

9 Bibliografía

Amrit, F. R. G., Ratnappan, R., Keith, S. A., & Ghazi, A. (2014). The C. elegans lifespan assay toolkit. *Methods*, *68*(3), 465-475. https://doi.org/10.1016/j.ymeth.2014.04.002

Benlloch, J. V., Agusti, M., Sanchez, A., & Rodas, A. (1995). Colour segmentation techniques for detecting weed patches in cereal crops. *Proc. of Fourth Workshop on Robotics in Agriculture and the Food-Industry*, 30–31.

Berti, E. M., Salmerón, A. J. S., & Viala, C. R. (2017). 4-Dimensional deformation part model for pose estimation using Kalman filter constraints: *International Journal of Advanced Robotic Systems*. https://doi.org/10.1177/1729881417714230

Berti, E., Nina, O., Sánchez-Salmerón, A.-J., & Viala, C. (2017). *Optimized 4D DPM for Pose Estimation on RGBD Channels using Polisphere Models*. 281-288. https://doi.org/10.5220/0006133702810288

Berti, E., Sánchez-Salmerón, A.-J., & Benimeli, F. (2012, enero 30). *Kalman Filter for Tracking Robotic Arms Using low cost 3D Vision Systems*. ACHI 2012 - 5th International Conference on Advances in Computer-Human Interactions.

Bidirectional recurrent neural networks. (2020). En *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Bidirectional_recurrent_neural_networks&oldid=948549158

Corsi, A. K., Wightman, B., & Chalfie, M. (2015). A Transparent Window into Biology: A Primer on Caenorhabditis elegans. *Genetics*, *200*(2), 387-407. https://doi.org/10.1534/genetics.115.176099

CS231n Convolutional Neural Networks for Visual Recognition. (s. f.). Recuperado 4 de agosto de 2020, de https://cs231n.github.io/convolutional-networks/#pool

Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., & Darrell, T. (2017). Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 677-691. https://doi.org/10.1109/TPAMI.2016.2599174

Feng Ning, Delhomme, D., LeCun, Y., Piano, F., Bottou, L., & Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, *14*(9), 1360-1371. https://doi.org/10.1109/TIP.2005.852470

García Garví, A. (2019). *Diseño, desarrollo y evaluación de un sistema de clasificación de objetos en imágenes que permita la monitorización de C. elegans mediante redes neuronales convolucionales*. https://riunet.upv.es/handle/10251/126302

¿GPU vs. CPU? ¿Qué es la computación por GPU? | NVIDIA. (s. f.). Recuperado 21 de julio de 2020, de https://www.nvidia.com/es-la/drivers/what-is-gpu-computing/

Grau, R., Sánchez-Salmerón, A.-J., Girón, J., Ivorra, E., Fuentes, A., & Barat, J. (2011). Nondestructive assessment of freshness in packaged sliced chicken breasts using SW-NIR spectroscopy. *Food Research International*, *44*, 331-337. https://doi.org/10.1016/j.foodres.2010.10.011

Hakim, A., Mor, Y., Toker, I. A., Levine, A., Neuhof, M., Markovitz, Y., & Rechavi, O. (2018). WorMachine: Machine learning-based phenotypic analysis tool for worms. *BMC Biology*, *16*. https://doi.org/10.1186/s12915-017-0477-0

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. https://doi.org/10.1109/CVPR.2016.90

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026-1034. https://doi.org/10.1109/ICCV.2015.123

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*. http://arxiv.org/abs/1502.03167

Ivorra, E., Sánchez, A. J., Camarasa, J. G., Diago, M. P., & Tardaguila, J. (2015). Assessment of grape cluster yield components based on 3D descriptors using stereo vision. *Food Control*, *50*, 273-282. https://doi.org/10.1016/j.foodcont.2014.09.004

Ivorra, Eugenio, Amat, S. V., Sánchez, A. J., Barat, J. M., & Grau, R. (2014). Continuous monitoring of bread dough fermentation using a 3D vision Structured Light technique. *Journal of Food Engineering*, *130*, 8-13. https://doi.org/10.1016/j.jfoodeng.2013.12.031

Ivorra, Eugenio, Sánchez-Salmerón, A.-J., Amat, S., Barat, J., & Grau, R. (2016). Shelf life prediction of expired vacuum-packed chilled smoked salmon based on a KNN tissue segmentation method using hyperspectral images. *Journal of Food Engineering*, *178*. https://doi.org/10.1016/j.jfoodeng.2016.01.008

Javer, A., Brown, A. E. X., Kokkinos, I., & Rittscher, J. (2019). Identification of C. elegans Strains Using a Fully Convolutional Neural Network on Behavioural Dynamics. En L. Leal-Taixé & S. Roth (Eds.), *Computer Vision – ECCV 2018 Workshops* (pp. 455-464). Springer International Publishing. https://doi.org/10.1007/978-3-030-11024-6_35

Jeremy Howard, & Sylvain Gugger. (2020). *Deep Learning for Coders with fastai and PyTorch*. https://learning.oreilly.com/library/view/deep-learning-for/9781492045519/

Kong, S., Florendo, E., Mou, Z., Lister, E., Cinquin, O., & Fowlkes, C. (s. f.). *Automated Biological Image Analysis using Computer Vision and Machine Learning*. 25.

Learning Rate Scheduling—Deep Learning Wizard. (s. f.). Recuperado 7 de julio de 2020, de

https://www.deeplearningwizard.com/deep_learning/boosting_models_pytorch/lr_scheduling/

Li, K., Javer, A., Keaveny, E. E., & Brown, A. E. X. (2017). *Recurrent Neural Networks with Interpretable Cells Predict and Classify Worm Behaviour* [Preprint]. Animal Behavior and Cognition. https://doi.org/10.1101/222208

Lin, J.-L., Kuo, W.-L., Huang, Y.-H., Jong, T.-L., Hsu, A.-L., & Hsu, W.-H. (2020). Using Convolutional Neural Networks to Measure the Physiological Age of Caenorhabditis elegans. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1-1. https://doi.org/10.1109/TCBB.2020.2971992

Martínez Bertí, E., Sánchez Salmerón, A. J., & Benimeli, F. (2012). Human robot interaction and tracking using low cost 3D vision systems. *Romanian Journal of Technical Sciences - Applied Mechanics*, *7*, 151-168. https://riunet.upv.es/handle/10251/47564

Martinez-Berti, E., Ai, I., Snchez-Salmern, A. J., & Ricolfe-Viala, C. (2016). *Human Pose Estimation for RGBD Imagery with Multi-Channel Mixture of Parts and Kinematic Constraints*. *15*, 8.

Martinez-Berti, E., Sánchez-Salmerón, A.-J., & Ricolfe-Viala, C. (2017). Dual Quaternions as Constraints in 4D-DPM Models for Pose Estimation. *Sensors (Basel, Switzerland)*, *17*(8). https://doi.org/10.3390/s17081913

Puchalt, J., Sánchez-Salmerón, A.-J., Ivorra, E., Genovés Martínez, S., Martínez, R., & Martorell, P. (2020). Improving lifespan automation for Caenorhabditis elegans by using image processing and a post-processing adaptive data filter. *Scientific Reports*, *10*. https://doi.org/10.1038/s41598-020-65619-4

Puchalt, J., Sánchez-Salmerón, A.-J., Martorell, P., & Genovés Martínez, S. (2019). Active backlight for automating visual monitoring: An analysis of a lighting control technique for Caenorhabditis elegans cultured on standard Petri plates. *PLOS ONE*, *14*, e0215548. https://doi.org/10.1371/journal.pone.0215548

¿Qué es la inteligencia artificial? (Spanish). (2019, mayo 7). *Jordi TORRES.AI*. https://torres.ai/que-es-la-inteligencia-artificial/

Ricolfe-Viala, C., & Sanchez-Salmeron, A. (2011). Optimal conditions for camera calibration using a planar template. *2011 18th IEEE International Conference on Image Processing*, 853-856. https://doi.org/10.1109/ICIP.2011.6116691

Sanchez, A. J., & Marchant, J. A. (2000). *Fusing 3D information for crop/weeds classification.* In Proceedings 15th International Conference on Pattern Recognition.

Sánchez, A. J., Albarracin, W., Grau, R., Ricolfe, C., & Barat, J. M. (2008). Control of ham salting by using image segmentation. *Food Control*, *19*(2), 135-142. https://doi.org/10.1016/j.foodcont.2007.02.012

Sánchez, A., & Ramos, C. (2000). SEGUIMIENTO VISUAL DE OBJETOS UTILIZANDO TÉCNICAS DE PREDICCIÓN. *XXI Jornadas de Automatica*, 6.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673-2681. https://doi.org/10.1109/78.650093

Silva, C. A., Magalhaes, K. M. C., & Neto, A. D. D. (2003). An intelligent system for detection of nematodes in digital images. *Proceedings of the International Joint Conference on Neural Networks, 2003.*, *1*, 612-615 vol.1. https://doi.org/10.1109/IJCNN.2003.1223431

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929-1958.

Supercomputación, corazón de Deep Learning. (2018, junio 11). *Jordi TORRES.AI*. https://torres.ai/supercomputacion-corazon-de-deep-learning/

Tissenbaum, H. A. (2015). Using C. elegans for aging research. *Invertebrate Reproduction & Development*, *59*(sup1), 59-63. https://doi.org/10.1080/07924259.2014.940470

Vanfleteren, J. R., De Vreese, A., & Braeckman, B. P. (1998). Two-Parameter Logistic and Weibull Equations Provide Better Fits to Survival Data From Isogenic Populations of Caenorhabditis elegans in Axenic Culture Than Does the Gompertz Model. *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, *53A*(6), B393-B403. https://doi.org/10.1093/gerona/53A.6.B393

Verdú, S., Ivorra, E., Sánchez, A. J., Barat, J. M., & Grau, R. (2015a). Relationship between fermentation behavior, measured with a 3D vision Structured Light technique, and the internal structure of bread. *Journal of Food Engineering*, *146*, 227-233. https://doi.org/10.1016/j.jfoodeng.2014.08.014

Verdú, S., Ivorra, E., Sánchez, A. J., Barat, J. M., & Grau, R. (2015b). Study of high strength wheat flours considering their physicochemical and rheological characterisation as well as fermentation capacity using SW-NIR imaging. *Journal of Cereal Science*, *62*, 31-37. https://doi.org/10.1016/j.jcs.2014.11.002

Wang, Z., Wang, D., Li, C., Xu, Y., Li, H., & Bao, Z. (2018). Deep reinforcement learning of cell movement in the early stage of C.elegans embryogenesis. *Bioinformatics*, *34*(18), 3169-3177. https://doi.org/10.1093/bioinformatics/bty323

Wu, Z., Yao, T., Fu, Y., & Jiang, Y.-G. (2017). Deep Learning for Video Classification and Captioning. *arXiv:1609.06782 [cs]*, 3-29. https://doi.org/10.1145/3122865.3122867