Segmentación de núcleos celulares en imágenes de microscopía ayudados por redes neuronales convolucionales

David García Seisdedos
Máster Universitario de Bioinformática y Bioestadística

Área de *Machine Learning*

Director:
Edwin Santiago Alférez Baquero

02/07/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	Segmentación de núcleos celulares en imágenes de microscopía ayudados por redes neuronales convolucionales.
Nombre del autor:	*David García Seisdedos*
Nombre del consultor/a:	*Edwin Santiago Alférez Baquero*
Nombre del PRA:	
Fecha de entrega :	06/2018
Titulación::	*Plan de estudios del estudiante*
Área del Trabajo Final:	*Machine Learning*
Idioma del trabajo:	***Español***
Palabras clave	*Microscopía, segmentación, red neuronal convolucional.*

Resumen del Trabajo:

La detección de núcleos celulares en imágenes de microscopía es una operación esencial en el análisis microscópico. Pero los métodos manuales de detección y segmentación de áreas consumen bastante tiempo del analista. Y por otra parte, los métodos clásicos de segmentación de imágenes no funcionan bien cuando existe alta celularidad.

El objetivo de este proyecto fue desarrollar una herramienta bioinformática para detectar y aislar núcleos celulares. El método desarrollado realiza la detección y segmentación en tres pasos. Primero, se lleva a cabo una segmentación inicial de la imagen en bruto. Segundo, los fragmentos obtenidos se clasifican mediante una red neuronal convolucional (CNN) en tres grupos: mono-núcleos, poli-núcleos o artefactos no-nucleares. Tercero, los mono-núcleos se almacenan, los artefactos se eliminan y los fragmentos polinucleares son segmentados y re-analizados desde el segundo paso. Así es posible subdividir por recursividad el elemento problema -acumulo de núcleos- en sub-problemas más sencillos.

La eficacia y robustez en la detección y segmentación se comprobó mediante el análisis de la base de datos de referencia BBBC020[25]. Los mejores resultados se alcanzaron con los modelos diseñados que utilizan conjuntamente un método de clasificación mediante CNN y la segmentación por aglomeración (obteniendo un error de contaje medio menor –ECM– del 1%), mejorando significativamente a los métodos de segmentación clásicos de *watershed* y Otsu (con un ECM de núcleos del 4% y 8% respectivamente).

Por tanto, el modelo propuesto detecta y segmenta núcleos celulares en imágenes de microscopía de forma efectiva, flexible (soportando bien el escalado) y rápida.

Abstract:

The detection of cell nuclei in microscopic imaging is a basic task in microscopic analysis. But manual methods of area detection and segmentation are a time consuming task for the analyst. On the other hand classic methods of image segmentation do not work well when there is a high density of touching cells.

Therefore, the aim of this project is to develop a bioinformatics tool to detect and isolate cell nuclei. The developed model performs detection and segmentation in three steps. First, an initial segmentation of the raw image is carried out. Second, the fragments obtained are classified by means of a convolutional neural network (CNN) into three groups: mono-nuclei, poly-nuclei or non-nuclear artifacts. Third, the artifacts will be removed and the polynuclear fragments will be segmented and will be analyzed from the second step again. Thus, the main problem - accumulation of nuclei - is subdivided by recursiveness into simpler sub-problems.

The effectiveness and robustness of detection and segmentation was proven by analysis of the reference database BBBC020[25]. It was observed that the best results were achieved with the models designed using a CNN classification method and agglomeration segmentation (obtaining an average count error of less than 1%), significantly improving the classical watershed segmentation and the Otsu method (with an average count error of 4% and 8% respectively).

Finally, the proposed model makes a robust segmentation and supports well the image scaling. In addition, the inference in nuclei detection is computationally fast.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

La microscopía de fluorescencia tiene un importante papel en el diagnóstico clínico, siendo una técnica ampliamente utilizada tanto en el laboratorio clínico (por ejemplo en laboratorios de inmunología, anatomía patológica, microbiología o citogenética), como en investigación biomédica. Las técnicas de fluorescencia más utilizadas en biomedicina son la inmunofluorescencia (IF) y la fluorescencia de hibridación in-situ (FISH). La técnica de IF (Figura 1 izquierda) es una técnica de inmunomarcación que hace uso de anticuerpos unidos químicamente a una sustancia fluorescente para demostrar la presencia de una determinada molécula. La técnica FISH (Figura 1 derecha) es utilizada en citogenética para el marcaje de cromosomas mediante hibridación por sondas fluorescentes.

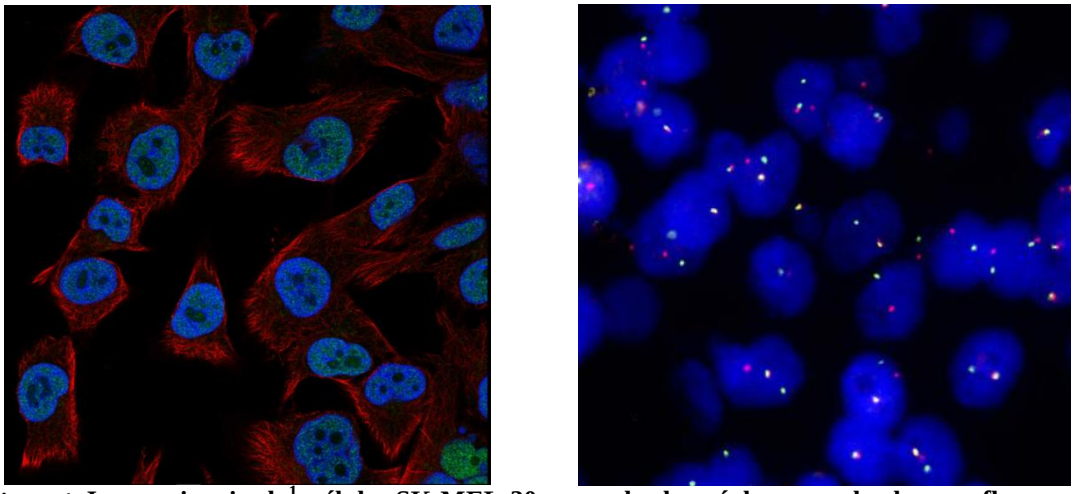


Figura 1. Imagen izquierda[1], células SK-MEL-30 marcados los núcleos con el colorante fluorescente DAPI (en azul) y se utilizó marcadores inmunofluorescentes para microtúbulos (en rojo) y para la proteína transportadora del colesterol 1 (en verde). Imagen derecha[2], técnica FISH con núcleos marcados con DAPI (azul) y sondas *break-apart* (verde y rojo) de la región del cromosoma 22q12.

El análisis de las imágenes de microscopía puede realizarse tanto manualmente como automáticamente. Pero ya que las células son elementos vivos y los procesos celulares son normalmente estocásticos[3], los análisis deben realizarse masivamente en cientos o miles de células para un determinado experimento con el fin de que sean resultados estadísticamente significativos.

El análisis manual de las imágenes de microscopía puede resultar tedioso y susceptible a errores. El error más común que se puede cometer en un análisis manual es el error de tendencia entre-ensayos o entre-analistas. Por consiguiente, los métodos de diagnósticos ayudados por ordenador -que mejoran la reproducibilidad y permiten una elevada velocidad de procesado de imágenes- han atraído gran interés por parte de la comunidad científica y médica[4].

Para llevar a cabo el diagnóstico ayudado por ordenador, primeramente es necesaria la detección y segmentación del núcleo celular o de la célula en su conjunto. Sin embargo, es difícil conseguir un método de segmentación robusto y preciso por varias

razones. Primero, la segmentación de células en imágenes requiere de identificar los múltiples objetos que ocupan la imagen[5]. Segundo, la distribución de intensidad en la célula no es homogénea -debido a la posición de los diferentes organelas- y por tanto se pueden dar gradientes de intensidad de señal no deseables. Tercero, el crecimiento celular habitualmente provoca agrupaciones celulares difíciles de aislar. Finalmente, el tratamiento experimental no suele ser perfectamente homogéneo en toda la preparación, por tanto puede que existan células con diferente intensidad de señal pero que fenotípicamente sean iguales.

La segmentación de núcleos se realiza en diferentes pasos. Por una parte, la detección de núcleos celulares pretende obtener la localización de un determinado núcleo sin que se tenga que delimitar perfectamente los límites del mismo. Este procedimiento sirve como punto de partida o semilla para una posterior segmentación. Existen varios métodos de detección de núcleos que han sido utilizados en la literatura y se pueden subdividir según el algoritmo principal utilizado. Así se pueden encontrar diferentes métodos que utilizan la transformada de distancia[6], operación morfológica[7], transformada H-mínima/máxima[8], filtro Laplaciano de la Gausiana[9], detección de la región extrema estable máxima[10], transformada de Hough[11], votación basada en simetría radial[12], así como también se pueden encontrar métodos de aprendizaje supervisado: Support Vector Machine[13], Random Forest[14], redes neuronales profundas, análisis por componentes principales, clasificación Bayesiana[15].

Por otra parte, en la segmentación de núcleos se intenta separar objetos diferentes mediante la delimitación de los límites. Generalmente existen tres estrategias para llevar a cabo la separación de los núcleos/células[16]: 1) separar el fondo del primer plano, 2) identificar las semillas de los objetos a identificar para posteriormente expandirlas hasta sus límites, y 3) generar regiones candidatas y después seleccionar los mejores candidatos para ser segmentados. Los algoritmos de segmentación más utilizados son: intensidad límite[17], transformación divisoria[18], métodos de agrupamiento (*clustering*) *(*K-means[19], *fuzzy c-means[20]*, maximización de la expectativa[21]), métodos basados en grafos[22] y clasificación supervisada (clasificación por pixeles[23], clasificación por super-pixeles[24]).

Muchas de las aproximaciones utilizadas en la bibliografía no dan unos resultados robustos en los diferentes *datasets*. Esto es debido a la poca flexibilidad que presentan estos métodos con respecto a la variabilidad intrínseca de la microscopía, dado que existe una gran variabilidad principalmente debida al tipo de muestra y al modo de preparación. Por tanto, la detección y segmentación de células de células/núcleos que están parcialmente superpuestas o pegadas entre sí sigue siendo un reto científico difícil de resolver.

Por tanto, el presente trabajo de investigación desarrollará una herramienta bioinformática de detección y segmentación de núcleos celulares mediante la utilización de redes neuronales convolucionales. Esta herramienta posteriormente será entrenada y evaluada con un conjunto de imágenes de microscopía públicas, como por ejemplo de la Broad Bioimage Benchmark Collection[25]. De este modo se comparará la eficacia de procesamiento en la detección y segmentación de células.

1.2 Objetivos del Trabajo

1.2.1. Objetivos principales:

Desarrollar una herramienta bioinformática para la detección de núcleos celulares utilizando redes neuronales convolucionales (CNN).

1.2.2. Objetivos específicos:

a. Desarrollar una herramienta para clasificar núcleos celulares utilizando CNN

i. Estudiar y comparar técnicas de segmentación de imágenes.

ii. Crear una base de datos con imágenes para el entrenamiento de la CNN.

iii. Desarrollar una CNN para clasificar las imágenes de entrada en tres clases de: un sólo núcleo, aglomerado nuclear y elemento no-nuclear.

b. Optimizar y flexibilizar la herramienta de detección y segmentación de núcleos celulares.

i. Optimizar la herramienta de detección de núcleos en agrupaciones celulares.

ii. Estudiar e implementar técnicas de escalado y de flexibilidad a la herramienta diseñada.

iii. Evaluar la herramienta de detección y segmentación de núcleos en diferentes datasets.

1.3 Enfoque y método seguido:

El objetivo final de este proyecto es desarrollar una herramienta bioinformática para detectar y aislar núcleos celulares en imágenes de microscopía de fluorescencia. En general, en los modelos desarrollados en la literatura primeramente se realiza la detección de los núcleos celulares para posteriormente utilizar dichas “semillas” como molde para la segmentación.

El modelo de este proyecto se realizará en tres pasos. Primero, se llevará a cabo una segmentación -de la imagen en bruto- para obtener fragmentos con núcleos celulares. Segundo, dichos fragmentos se evaluarán mediante la utilización de una CNN y se clasificarán en tres grupos, es decir, en regiones mono-nucleares, polinucleares o regiones con artefactos no-nucleares. Tercero, las regiones no-nucleares se eliminaran y si los fragmentos contienen dos o más núcleos, éstos serán segmentados en dos partes y cada parte será procesada por recursividad desde el segundo paso. De esta manera, se pueden separar agrupaciones celulares que los métodos simples de segmentación no son capaces de separar.

El modelo de CNN propuesto intenta simular la percepción que tiene el analista para discernir si un determinado elemento microscópico corresponde a uno o varios núcleos. Y por otra parte, el modelo propuesto intentará subdividir el elemento problema -acumulo de células no segmentadas- en subproblemas más sencillos mediante recursividad.

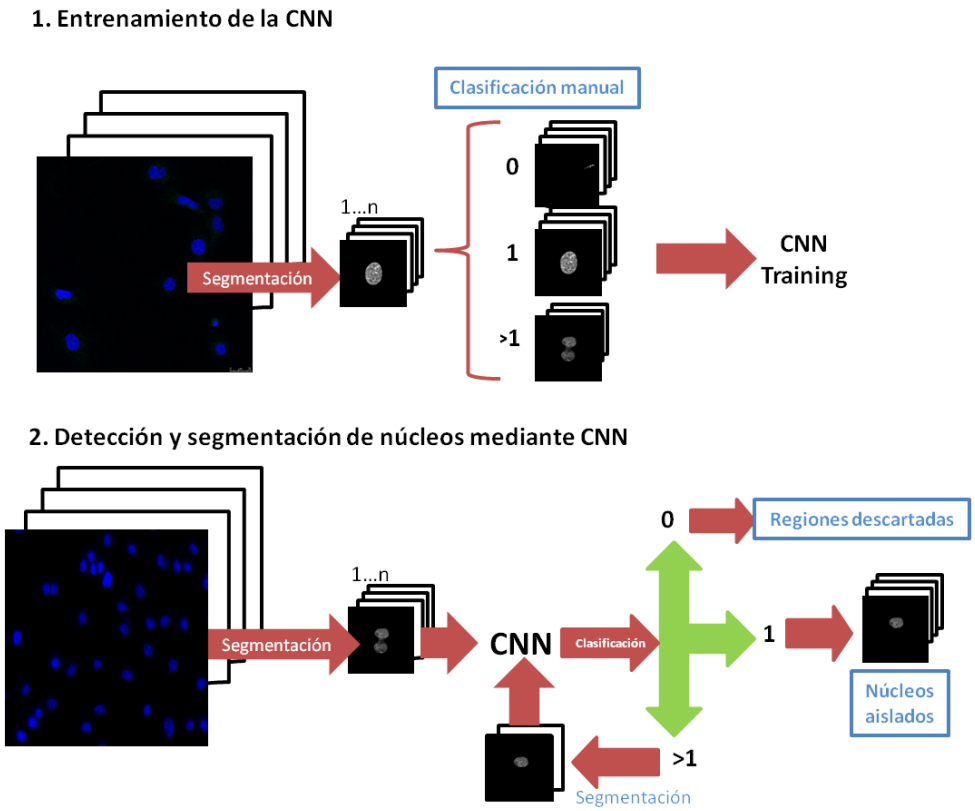


Figura 2: **Diagrama del modelo propuesto.**

1.4 Planificación del Trabajo:

1.4.1. Tareas:

Tareas para la Fase 1:

1.i.a. Comparar y seleccionar el mejor método de segmentación de núcleos celulares frente al fondo de la imagen. Duración: 5 días (15 horas).

1.ii. Desarrollar un pipeline para la creación de una base de datos con imágenes "recortadas" de elementos segmentados a partir de imágenes en bruto de microscopía. Dicha base de datos subdividirá las imágenes según el número de núcleos que existan por imagen. Duración: 3 días (9 horas).

1.i.b. Comparar y seleccionar el mejor método de segmentación de núcleos celulares a partir imágenes "recortadas" que contengan dos o más núcleos. Duración 2 días (6 horas).

1.iii.a. Estudiar y comprender el fundamento de las CNN. Duración: 4 días (12 horas).

1.iii.b. Desarrollar y optimizar una CNN para la clasificación de imágenes. Duración: 9 días (27 horas).

1.iv. Elaborar el informe de resultados de la primera fase. Duración: 4 días (12 horas).

Tareas para la Fase 2:

2.i. Desarrollar un *pipeline* utilizando un proceso recursivo para aislar núcleos de una agrupación de núcleos. Duración: 5 días (15 horas).

2.ii. Creación del software de detección y segmentación de núcleos a partir de los diferentes *pipelines* desarrollados. Duración: 10 días (30 horas).

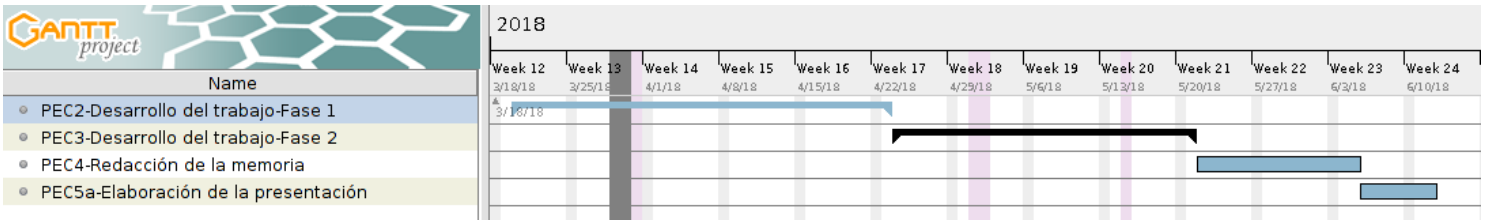
2.iii. Evaluar el software de análisis creado con *datasets* de imágenes públicas. Duración: 2 días (6 horas).
2.iv. Elaborar el informe de resultados de la segunda fase. Duración: 4 días (12 horas).

1.4.2. Calendario:

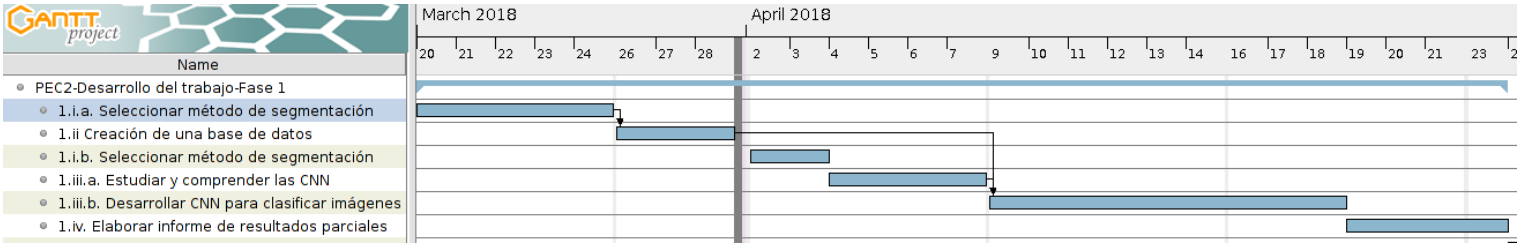
El calendario con las diferentes etapas del proyecto se ha establecido teniendo en cuenta los días festivos. Es decir, no se ha designado trabajo los domingos, la semana santa (desde el 29 de marzo hasta el 1 de abril), el uno de mayo, el dos de mayo ni en San Isidro (15 mayo). Por otra parte, el tiempo asignado a cada día de trabajo es de tres horas por cada jornada.

La división de tiempo de cada proceso puede ser observada con los diagramas de Gantt.

Programación del tiempo del proyecto global:

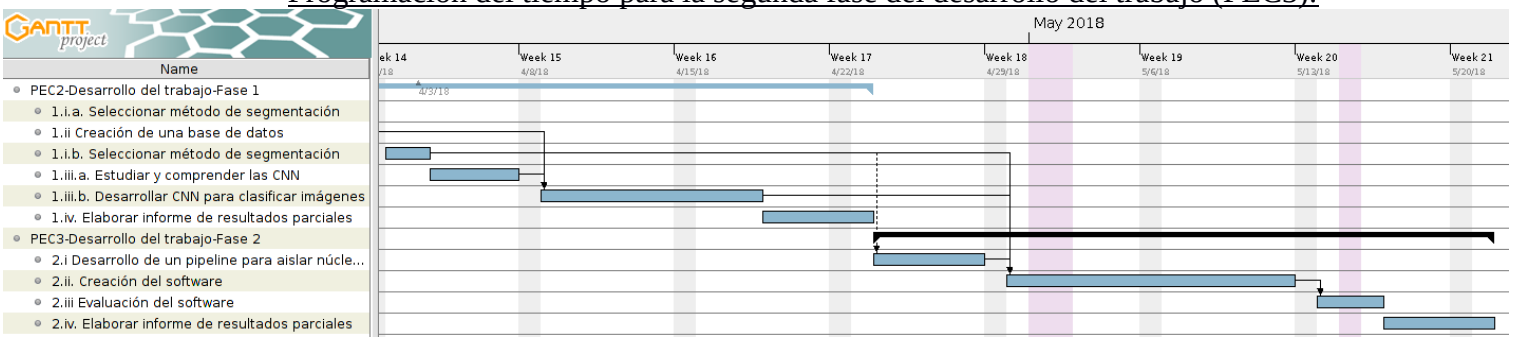


Programación del tiempo para la primera fase del desarrollo del trabajo (PEC2):



Se puede observar varias relaciones de interdependencia de tareas. La primera relación es la tarea de selección del método de segmentación (1.i.a) con la creación de la base de datos (1.ii). La segunda relación es la creación de la base de datos (1.ii.) y el estudio de las CNN (1.iii.a.) sobre la tarea de desarrollo de una CNN (1.iii.b).

Programación del tiempo para la segunda fase del desarrollo del trabajo (PEC3):



Se puede observar varias relaciones de interdependencia de tareas. La primera relación es la tarea de selección del método de segmentación (1.i.b) con la tarea de detección y

aislamiento de núcleos celulares en núcleos agrupados (2.ii). La segunda relación es las tareas 1.i.b, 1.iii.a y 2.i sobre la creación del software de detección de núcleos celulares (2.ii). La tercera relación es la tarea 2.ii sobre la evaluación del software.

1.4.3. Hitos:

Tabla 1: Hitos del desarrollo del trabajo de la Fase 1

Hito	***Fin de plazo***
Seleccionar un método de segmentación para separar núcleos celulares del fondo de la imagen.	24 de Marzo 2018
Desarrollar un *pipeline* para la creación de una base de datos con imágenes recortadas de núcleos celulares a partir de un *batch* de imágenes de microscopía.	28 de Marzo 2018
Seleccionar un método de segmentación para separar núcleos celulares entre sí.	3 de Abril 2018
Desarrollar un *pipeline* utilizando CNN para clasificar imágenes.	18 de Abril 2018
Elaborar informe de resultados de la Fase 1.	23 de Abril 2018

Tabla 2: Hitos del desarrollo del trabajo de la Fase 2

Hito	***Fin de plazo***
Desarrollar un *pipeline* utilizando un proceso recursivo para aislar núcleos a partir de una agrupación de núcleos.	28 de Abril 2018
Implementar un software de detección y segmentación de núcleos celulares utilizando los pipelines anteriormente desarrollados.	16 de Mayo 2018
Elaborar informe de resultados de la Fase 2	21 de Mayo 2018

Tabla 3: Hitos de la post-producción

Hito	***Fin de plazo***
Redacción de la memoria.	5 de Junio 2018
Elaboración de la presentación virtual.	12 de Junio 2018

1.4.4. Análisis de riesgos:

El desarrollo del proyecto fin de máster puede sufrir riesgos que afecten negativamente al curso y a la consecución del mismo. Por tanto, se realiza un estudio pormenorizado de los posibles riesgos que pueden surgir así como un plan de contingencia.

1. Problemas técnicos: debido a que todo el proyecto se realiza con ordenador, todos los posibles riesgos asociados a su uso pueden ser: cortes eléctricos inesperados, subida de tensión eléctrica o mal funcionamiento de algún componente del ordenador. Para minimizar los problemas eléctricos se realizarán recurrentes copias de seguridad en varios discos duro externos. Para el caso de que el ordenador personal deje de funcionar se deberá realizar la compra de otro equipo que incorpore una tarjeta gráfica de similares características. El coste total para un ordenador de sobremesa con una tarjeta gráfica de gama media-baja -por ejemplo, un ordenador de sobremesa Lenovo con CPU Intel i7 6700,12 GB de RAM y tarjeta gráfica NVIDIA GeForce GTX 750Ti- es de: 1.198,99 euros en Amazon.

2. Desajuste de la duración de los objetivos. En caso de descuadre del organigrama de trabajo se asignarán horas adicionales a las tareas deficitarias.

3. Enfermedad. Si ocurre una enfermedad o accidente leve-moderado se intentará recuperar las horas perdidas trabajando los días festivos.

1.4.5. Costes asociados a la realización del trabajo y producto final:

El proyecto fin de máster será realizado en el domicilio del autor, donde se dispone de un ordenador personal con una GPU de gama media-baja (Nvidia GeForce GTX 750Ti). Dicho equipo se prevé con capacidad suficiente para llevar a cabo los análisis previstos en el proyecto. Por otra parte, todo el software que se utilice en el proyecto será software libre.

Por tanto, el coste económico del proyecto solo está asociado al coste de la electricidad empleada.

1.4.6. Implicaciones legales y éticas del uso de datos:

En este trabajo fin de máster no se utilizarán datos ni muestras de pacientes. Las imágenes no públicas de cultivos celulares que se utilizarán en el proyecto son de células de ratón.

1.5 Breve sumario de productos esperados

Los productos obtenidos a la finalización del trabajo serán:

- Plan de trabajo.
- Informe de resultados parciales de la fase 1 y 2.
- Software escrito en Python 3.0 para la detección y segmentación de núcleos celulares utilizando como input un conjunto de imágenes de microscopía realizadas todas ellas en las mismas condiciones.
- Presentación virtual.

1.6 Breve descripción de los otros capítulos de la memoria

- Capitulo 1: **Antecedentes bibliográficos de las redes neuronales convolucionales.** Se realiza un estudio bibliográfico sobre las redes neuronales, haciendo especial hincapié en las redes neuronales convolucionales.
- Capitulo 3: **Materiales y métodos.** Describe los materiales y métodos utilizados durante el desarrollo del proyecto de investigación.
- Capitulo 4: **Desarrollo del software.** Describe paso por paso las técnicas y los scripts utilizados para el completo desarrollo del software de detección y segmentación de núcleos celulares.
- Capitulo 5: **Resultados**. En dicho capitulo se resume los resultados obtenidos del análisis realizado por los diferentes modelos desarrollados sobre dos bases de datos de imágenes de microscopía.
- Capitulo 6: **Conclusión**. Argumentos y afirmaciones obtenidos tras la consecución del proyecto.
- Capitulo 7: **Anexo**.
- Capitulo 8: **Bibliografía**.

2. Antecedentes bibliográficos de las redes neuronales convolucionales

2.1 Introducción:

La red neuronal artificial es un modelo de inteligencia artificial, inspirado en la red neuronal biológica. Se fundamenta en la utilización de una colección de nodos, llamados neuronas artificiales, conectados con otros nodos de diferente capa neuronal. En cada conexión entre neuronas artificiales se transmite información de forma unidireccional. La neurona receptora recibe las señales de una o más neuronas y las procesa para enviar una nueva señal procesada a las neuronas que están en una capa neuronal superior.

En la red neuronal artificial la señal transmitida entre conexiones es un número real y el procesado de dichas señales se calcula mediante una función suma no lineal de todas ellas. Para que el algoritmo pueda "aprender" es necesario la utilización y el ajuste de constantes modificadoras -también conocidos como pesos (*weights*)- sobre las neuronas o conexiones neuronales, que incrementen o disminuyan la señal procesada por la neurona.

Generalmente la red neuronal artificial está organizada en capas. Las diferentes capas llevan a cabo diferentes transformaciones sobre las señales de entrada. En una red con n capas la conformación es la siguiente: la primera capa de *input*, n-2 capas neuronales ocultas (*hidden neural layers*) y una última capa de clasificación. La señal viaja desde la primera capa hasta la última después de atravesar las diferentes capas ocultas.

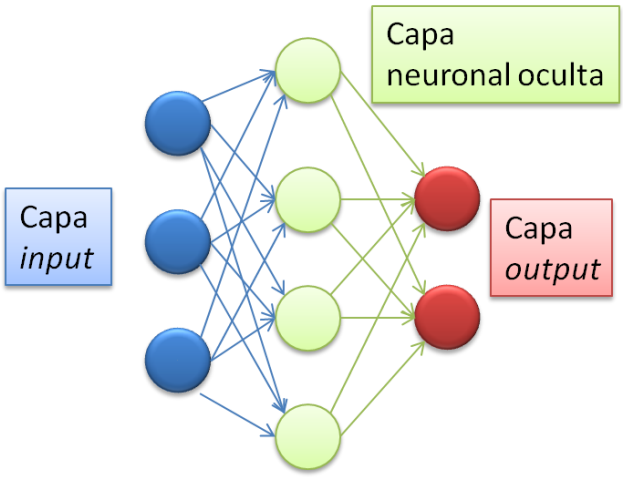


Figura 3: Conformación de una red neuronal simple con tres capas: capa input, capa neuronal oculta y capa output.

Los primeros científicos que desarrollaron un modelo computacional de redes neuronales fueron Warren McCulloch y Walter Pitts[26] utilizanhhdo el cálculo lógico. Este modelo sentaba las bases de la utilización de redes neuronales en la inteligencia artificial. Más tarde, el psicólogo D. O. Hebb[27] describió la hipótesis de la plasticidad neuronal. Esta hipótesis sirvió de base para que en 1948 Alan Turing[28] desarrollara el concepto de máquinas desorganizadas en modelos computacionales. El primer modelo de máquina desorganizada -conocido como máquina de Turing tipo A- conectaba al azar redes de puertas lógicas NAND. El segundo modelo desarrollado –conocido

como máquina de Turing Tipo B- podría ser creado tomando la máquina de Turing tipo A y reemplazando cada conexión inter-nodo con un modificador, éste permitiría a la máquina de tipo B llevar a cabo la "educación" de la red.

En 1975, el científico Paul J. Werbos desarrolló eficazmente un algoritmo de entrenamiento de las capas de redes neuronales mediante un entrenamiento hacia atrás (*backpropagation*). Éste distribuye el término error por las diferentes capas neuronales de manera reversa, mediante la modificación de los pesos de cada nodo.[29]

A pesar de que la comunidad científica mostró un gran interés por las redes neuronales, éstas no fueron muy utilizadas debido a la complejidad y el alto coste computacional que presentaban. Por ello, otros modelos de *machine learning* más simples, como: *support vector machines*, *random forest* o árboles de decisión, han tenido un mayor éxito en la clasificación de datos. Aunque por otra parte, dichos algoritmos obtienen un pobre resultado en la clasificación de imágenes.

Los problemas intrínsecos al uso de redes neuronales artificiales han podido ser solventados debido tanto a una mejora en el hardware como a una optimización de los algoritmos. La mejora a nivel de hardware vino por la utilización GPUs, dada la posibilidad de poder llevar a cabo multi-paralelización. La mejora de los algoritmos vino por el uso de modelos neuronales pre-entrenados, que han podido disminuir el tiempo de entrenamiento y el número de información necesaria para llevar a cabo el entrenamiento. Por otra parte, la utilización de la función *max-pooling* (función no-lineal que reduce la resolución) mejora la reproducibilidad de la red neuronal.

Entre los diferentes tipos de redes neuronales, la red neuronal convolucional (CNN) ha sido la que más éxito ha tenido debido al análisis de imágenes. La CNN es una subclase de red neuronal centrada en la estructura espacial de los *inputs*. La configuración de la CNN alterna capas convolucionales con capas *max-pooling* y por encima de ellas hay varias capas completamente o escasamente conectadas (*fully/sparsely connected layers*) seguidas de una capa final de clasificación.

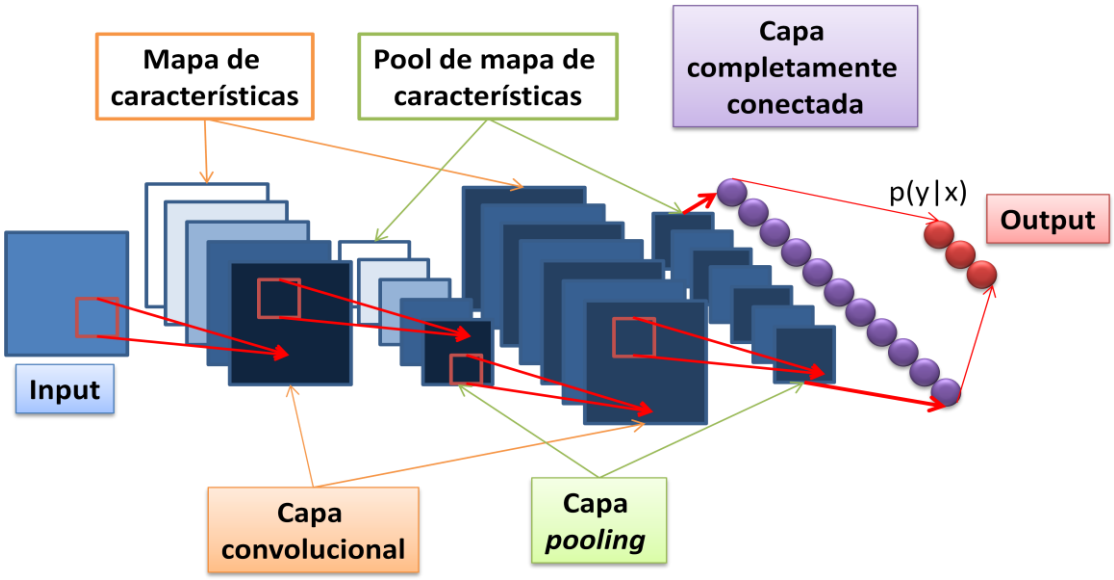


Figura 4: Conformación de una red neuronal convolucional[30].

2.2 Estructura de las redes neuronales convolucionales:

Las CNN son redes *feedforward* (prealimentadas) en las cuales la información fluye de manera unidireccional, desde los *inputs* hasta los *outputs*. Así como, las redes neuronales artificiales están inspiradas biológicamente, también lo están las CNNs. Las arquitecturas de las CNNs están basadas en el cortex visual del cerebro, que consiste en capas simples y complejas alternantes.[31]

La composición de las CNNs presenta muchas variantes. Sin embargo, en general éstas consisten en agrupaciones de módulos que contienen capas convolucionales y de reducción de resolución (*pooling*). En el caso del modelo de CNN más sencillo (modelo secuencial), los módulos se apilan unos encima de otros para formar un modelo profundo.

La capa convolucional está compuesta por varios *kernels* convolucionales los cuales son utilizados para calcular los mapas de características, con el fin de extraer información de las éstas[32]. Específicamente, cada neurona del mapa de características está conectada con otras neuronas de regiones vecinas de la siguiente capa. Dicha región es considerada como el campo receptivo de la neurona de la capa previa. El nuevo mapa de características puede ser obtenido primeramente por convolución del *input* con el *kernel* aprendido, para posteriormente aplicar una función no-lineal de activación. El mapa de características completo es obtenido mediante la utilización de diferentes *kernels*. El valor de cada característica en una determinada localización (i, j) en el k mapa de características de la capa l, $z_{i,j,k}^{l}$, es calculado por:

$$z_{i,j,k}^{l} = {w_k^l}^T x_{i,j}^{l} + b_k^l \text{ [1]}$$

Donde w_k^l y b_k^l representa al vector de pesos y a la tendencia del filtro k de la capa l respectivamente. La región input centrada en la localización (i, j) de la capa l se representa como: $x_{i,j}^{l}$.

Por otra parte, la función de activación introduce la no-linealidad al CNN, lo cual es deseable para detectar características no lineales. Siendo a(…) la función de activación. Las funciones de activación más utilizadas son: las sigmoideas, la tangente hiperbólica[33] y la unidad lineal de rectificación (ReLU)[34]. El valor de activación $a_{i,j,k}^{l}$ de la característica convolucional $z_{i,j,k}^{l}$ puede ser calculada como:

$$a_{i,j,k}^{l} = a(z_{i,j,k}^{l}) \text{ [2]}$$

Posteriormente, se suele utilizar una capa de *pooling* para disminuir la variabilidad (frente a la distorsión y translación de los datos de entrada) reduciendo la resolución espacial del mapa de características. Habitualmente el procedimiento de *pooling* es realizado entre dos capas convolucionales. El resultado de aplicar la función *pool* al valor de activación es:

$$y_{i,j,k}^{l+1} = pool\left(a_{m,n,k}^{l}\right), \forall (m,n) \in R_{ij} \text{[3]}$$

Donde R_{ij} representa a la vecindad local alrededor de la localización (*i,j*). Las funciones *pooling* más utilizadas son *average pooling*[35] y *max-pooling*[36,37]. En

especial, la capa de agregación por *max-pooling* mejora sustancialmente la generalización -y por ende la robustez del modelo- mediante la propagación del máximo valor del campo receptivo a la siguiente capa.

Generalmente, los *kernels* convolucionales de las capas más bajas están diseñados para detectar características básicas como curvas y límites, mientras que los *kernels* de capas más altas se centran en detectar características más abstractas.

Después de varias capas convolucionales y de reducción de resolución, puede haber una o más capas completamente conectadas con el fin de interpretar las características extraídas y de llevar a cabo un alto nivel de razonamiento computacional.[38]

La última capa de las CNNs es una capa de *output*. Si la finalidad del modelo es la clasificación, se llevará a cabo una operación *softmax*[39], es decir se comprime un vector K-dimensional, z, en un vector K-dimensional $\sigma(z)$ en el rango de (0,1). Siendo θ todos los parámetros del CNN (vectores de pesos y la constantes de tendencia). El parámetro óptimo para una tarea específica puede ser obtenido por la minimización de la función *loss* definida para cada tarea. Si se tienen N relaciones input-output {($x^{(n)}$, $y^{(n)}$); n ∈ [1,…,N] donde $x^{(n)}$ son los n datos de entrada e $y^{(n)}$ es la correspondiente etiqueta para dichos datos, el $o^{(n)}$ es el output del CNN. Por tanto, la función *loss* del CNN puede ser calculada:

$$L = \frac{1}{N} \sum_{n=1}^{N} l(\theta; y^{(n)}, o^{(n)}) \text{ [4]}$$

El entrenamiento de la CNN es un problema global de optimización. El mejor ajuste de los parámetros de la CNN puede ser realizado por minimización de la función *loss*.

Por otra parte, si la capa final es un clasificador *softmax*, la probabilidad de pertenecer a cada una de las clases viene dado por:

$$y_i = \frac{exp\,(-z_i)}{\sum_{j=i}^{K} exp\,(z_j)} \text{ [5]}$$

2.3 Modelos de CNN:

2.3.1 Modelo AlexNet

El campo del aprendizaje profundo resurgió en el 2006[40] debido al gran éxito obtenido en una variedad de tareas, como: la clasificación, reconocimiento de imágenes y objetos[41,42], reconocimiento facial[43] y segmentación de imágenes[44]. A pesar de éstos avances, las CNNs no habían sido muy utilizadas ni en el ámbito de la visión computacional ni el del aprendizaje profundo. Esto cambió después del congreso ILSVRC en el 2012, cuando una CNN desarrollada por Krizhevsky *et al.*[45] (AlexNet) consiguió el mejor resultado en la clasificación de imágenes de conjunto de imágenes de ImageNet. Este trabajo revolucionó el campo de la visión computacional y desde entonces las CNNs han sido la arquitectura que mejores resultados ha obtenido en la mayoría de tareas visuales y en particular en la clasificación de imágenes.

El éxito del modelo de Krizhevsky fue debido a la utilización de métodos novedosos. El modelo consiste de cinco capas convolucionales: tres de las cuales están seguidas

de capas de *max-pooling* y tres capas completamente conectadas. Como función de activación fue utilizada la función ReLU, que permite un rápido entrenamiento. Para no llegar al *overfitting* los autores emplearon la técnica *dropout*[46], que consiste en silenciar unas determinadas neuronas al azar, antes de que cada caso sea presentado a la CNN en la fase de entrenamiento. De esta manera, se previenen co-adapataciones superfluas con los datos de entrenamiento. Por otra parte, el *overfitting* también fue reducido por un aumento artificial de los datos de entrada, modificando las imágenes input mediante la aplicación de translaciones, reflexiones y alteraciones de intensidades.

2.3.2 Modelo VGG

El modelo Visual Geometry Group (VGG) desarrollado por Simonyan y Zisserman[47], consiguió el segundo mejor resultado en la clasificación de imágenes del ILSVRC en 2014. El modelo (VGG-16) utilizaba una red convolucional profunda, que consistía en 16 capas apiladas secuencialmente. Esto fue posible utilizando un pequeño filtro convolucional (3x3) por toda la red, minimizando de este modo los campos receptivos. El resultado es que había menos parámetros y más no-linealidades, facilitando la discriminación de la función de decisión y por tanto haciendo más fácil su entrenamiento.

2.3.3 Modelo GoogLeNet

El modelo que ganó el concurso ILSVRC del 2014 fue GoogLeNet[48], desarrollado por Szegedy *et al*. Introduciendo una novedosa arquitectura: el módulo *Inception*, que se basa en la utilización de convoluciones con diferente tamaño de filtro de forma paralela para posteriormente concatenar los resultados. Dicho modelo utilizaba 22 capas. Para superar el alto coste de utilizar un gran número de parámetros –ya que la red es más propensa al *overfitting*- se diseñó una nueva arquitectura de la red basada en los principios de Hebbian, que permitía cambiar de una red convolucional totalmente conectada a una escasamente conectada. Específicamente, su arquitectura utilizaba convoluciones 1×1 que servían como bloques de reducción de la dimensionalidad, para posteriormente llevar a cabo convoluciones 3×3 y 5×5 computacionalmente más costosas. De esta manera, se podía incrementar la profundidad y la anchura de la red sin aumentar considerablemente el coste computacional.

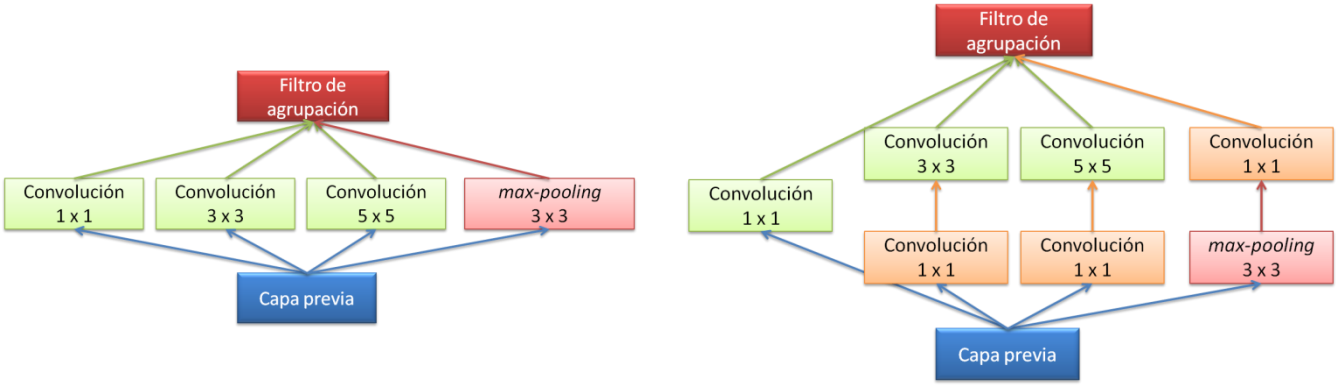


Figura 5: Modulo *inception*. A) versión básica. B) con reducción de la dimensionalidad[48].

2.4 Segmentación semántica:

Hasta ahora los modelos descritos fueron ideados para la **clasificación de imágenes en categorías**, realizando una predicción global de las clases de elementos que están presentes en el *input*. Es decir, pueden predecir el contenido global pero no el número ni la posición.

Las siguientes evoluciones lógicas en la inferencia de una imagen son mediante: **localización de objetos, segmentación semántica y segmentación semántica de instancias**[49]. La localización de objetos obtiene información de las clases presentes y de la localización de las mismas. La segmentación semántica realiza inferencias por pixel de las diferentes clases posibles. Por otra parte, la segmentación semántica de instancias puede segmentar elementos que pertenecen a la misma clase y que están pegados entre sí.

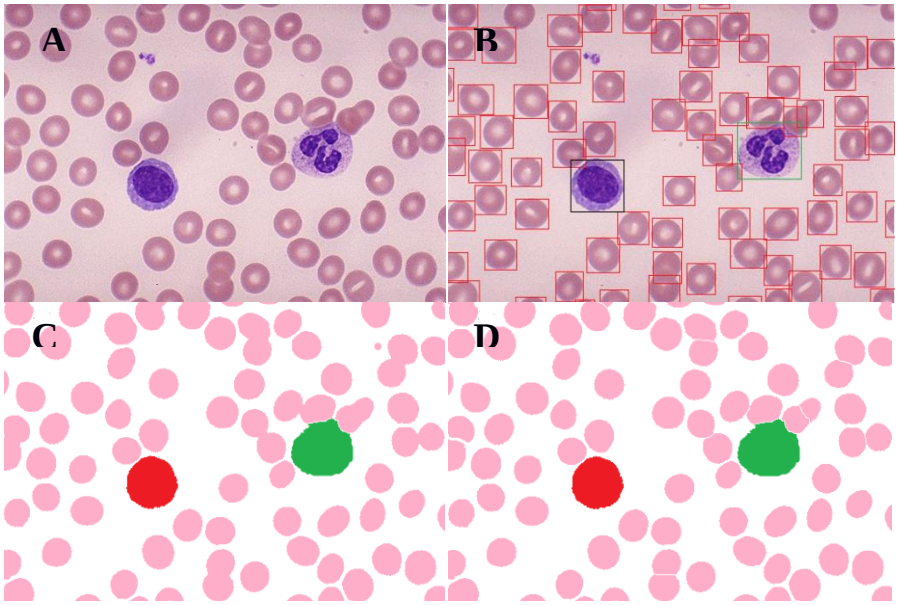


Figura 6: Evolución del reconocimiento de objetos en imágenes. Imagen: frotis sanguíneo. A) Clasificación de elementos en la imagen: hematíe, neutrófilo y linfocito. B) Localización de elementos en recuadros: hematíes (rojo), neutrófilo (verde) y linfocito (negro). C) Segmentación semántica: hematíes (rosa), neutrófilo (verde) y linfocito (rojo). D) Segmentación semántica de instancias: hematíes (rosa), neutrófilo (verde) y linfocito (rojo).

El precursor de la segmentación semántica fue el modelo de Red Neuronal Completa (FCN) desarrollado por Long *et al*[50]. La aproximación del modelo fue mediante el uso de CNNs ya conocidas y ampliamente usadas en la clasificación de imágenes. Para ello, transformaron los modelos de clasificación - `AlexNet`, `VGG`, `GoogLeNet` y `ResNet` – en modelos completamente convolucionales remplazando las últimas capas completamente conectadas con capas convolucionales cuyo *output* son mapas espaciales en vez de una puntuación para la clasificación de clases. Estos mapas de características son aumentados de resolución mediante el uso de convoluciones de pasos fraccionados (también conocido como deconvolución) para producir un output con un aumento de resolución, en el que cada pixel es etiquetado según la clase inferida. Éste modelo es considerado un hito en el aprendizaje profundo ya que muestra como las CNNs pueden ser entrenadas de forma *end-to-end*.

A pesar de la efectividad y flexibilidad del modelo FCN presenta varias carencias como son: la invariancia espacial que presenta no tiene en cuenta información global

que forma parte del contexto, no separa instancias similares que no presenta un discontinuo entre éstas y el entrenamiento de estos modelos es computacionalmente muy costosos (aproximadamente entre uno y tres días utilizando un equipo de altas prestaciones como es la GPU NVIDIA Tesla K40c).

A partir de la arquitectura FCN se han descrito otras variantes que han tenido bastante éxito. En general, todos ellas utilizan una red pre-entrenada para la clasificación eliminando las capas completamente conectadas, a esta parte de la arquitectura se la conoce como "codificador": que produce mapas de características o imágenes con baja resolución. Por otra parte, la parte específica de cada modelo es el "decodificador", cuya función es mapear las imágenes de baja resolución para crear imágenes de mayor resolución con predicciones de clase por pixel.

Uno de los modelos que más éxito ha tenido es `SegNet`[51]. En el modelo `SegNet` el decodificador está compuesto por un conjunto de capas de aumento de resolución y de capas convolucionales que finalmente están seguidas por un clasificador *softmax* para predecir las etiquetas por pixel para un *output* que tiene la misma resolución que la imagen *input*. Cada aumento de resolución de la parte decodificadora corresponde con la función *max-pooling* utilizada en la parte codificadora.

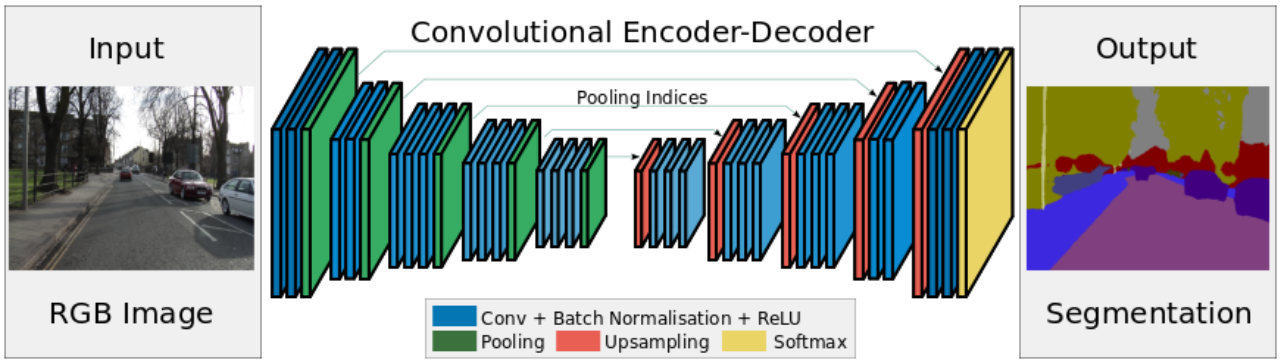


Figura 7: Arquitectura `SegNet`[51] con una parte codificadora y otra parte decodificadora, con una última capa de clasificación *softmax* por pixel.

3. Materiales y métodos

3.1 Cultivos celulares

La línea celular utilizada fue la línea C6, que son células gliales tumorales derivadas de la rata inducida con N-nitrosometilurea. Ésta línea celular fue adquirida a Sigma-Aldrich. El mantenimiento del cultivo celular se realizó utilizando medio Ham F12, añadiendo 2mM de glutamina y 10% de suero fetal bobino. Por otra parte, las condiciones ambientales utilizadas para el mantenimiento del cultivo fueron con 5% de CO_2 y a 37ºC.

3.2 Procedimiento de tinción

Para la visualización del núcleo celular, las células fueron lavadas previamente con PBS, posteriormente se añade colorante Hoecht 33342 1µg/mL en PBS y se incuba durante 10 minutos a temperatura ambiente. Una vez terminado se lava con PBS y se monta la preparación con glicerol al 70%.

3.3 Adquisición de imágenes

Las imágenes fueron adquiridas utilizando un microscopio confocal espectral Leica TCS SP5. Para la detección de núcleos fue utilizado un láser diodo azul a 405nm. El objetivo utilizado en todas las imágenes procesadas fue 63X/1,4-0,6 Oil. Para la adquisición digital de las imágenes se utilizó el programa LCS. Las imágenes obtenidas tienen una resolución de 1024 x 1024 pixeles.

3.4 Hardware

El ordenador utilizado presenta dichas características:

- CPU – Intel Core i3-4130, 3.40GHz
- RAM – 8Gb
- SSD – 256Gb
- GPU – GeForce GTX 750Ti

3.5 Base de datos de las imágenes de entrenamiento:

Se ha elaborado un *pipeline* en Python con el fin de crear una base de datos con imágenes de núcleos (subido al repositorio de GitHub como: `Creación_de_la_base_de_datos.ipynb`). En total fueron procesadas 20 imágenes de microscopía obteniéndose 1513 imágenes segmentadas, de las cuales 1392 corresponden a un único núcleo y 121 corresponden a agregados nucleares. Dada la disparidad entre las clases se diseñó un *pipeline* para crear imágenes de agregados nucleares a partir de células individuales (`Synthetic_DB_v3.py`).

3.6 Análisis estadístico:

La evaluación de la eficacia en la segmentación de núcleos celulares se realiza mediante la evaluación del error relativo en el contaje de núcleos por imagen y mediante la evaluación de la exactitud en la segmentación de imágenes con respecto a una imagen de referencia.

El **error relativo del contaje de núcleos** podemos formularlo como:

$$\boldsymbol{\varepsilon_{i,m}} = \frac{\mathbf{n_{i,m} - n_{i,r}}}{\mathbf{n_{i,r}}} \quad [6]$$

Si $\varepsilon_{i,m} > 0$ el método tiende a sobre-segmentar los núcleos.

Si $\varepsilon_{i,m} < 0$ el método tiende a infra-segmentar los núcleos.

Siendo $\varepsilon_{i,m}$ el error relativo del método *m* en la imagen *i*, $n_{i,m}$ el número de núcleos segmentados en la imagen *i* utilizando el método de segmentación *m* y $n_{i,r}$ el número de núcleos reales en la imagen *i*.

Por otra parte, los **métodos de evaluación de la exactitud** más utilizados en el análisis de técnicas de segmentación semántica son variaciones de la exactitud por pixel (PA) y intersección sobre unión (IoU).

Para poder realizar la evaluación de la exactitud se asumirán que existen un total del *k + 1* clases (es decir k número de regiones segmentadas más el fondo), p_{ij} es la cantidad de pixeles de la clase *i* que son inferidas a pertenecer a la clase *j*. Es decir, p_{ii} representa el número de verdaderos positivos, mientras p_{ij} y p_{ji} son interpretados como falsos positivos y falsos negativos respectivamente.

Las funciones de evaluación de exactitud que utilizaremos serán:

Exactitud por pixel (PA): es la métrica más sencilla que calcula el ratio entre la cantidad de pixeles clasificados correctamente frente al número total de ellos.

$$\mathbf{PA} = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}} \text{[7]}$$

Exactitud media por pixel (MPA): calcula el ratio entre la cantidad de pixeles clasificados correctamente frente al número total de cada clase y se promedia frente al número total de clases.

$$\mathbf{MPA} = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij}} \text{ [8]}$$

Intersección media sobre unión (MIoU): es considerada como la medida estándar para evaluar métodos de segmentación. Calcula el ratio entre la intersección y la unión entre dos conjuntos, en el caso que nos ocupa sería: la imagen preclasificada frente a la imagen segmentada y etiquetada por el método de segmentación elegido. Este ratio puede ser reformulado como el número de verdaderos positivos (intersección) sobre la suma de verdaderos positivos, falsos negativos y falsos positivos (unión). La IoU es calculada por cada clase y posteriormente se promedia.

$$\boldsymbol{MIoU} = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}} \text{ [9]}$$

4. Desarrollo del software

4.1 Introducción

La detección de núcleos celulares en imágenes de microscopía puede ser una tarea compleja si existe un solapamiento entre núcleos celulares, dado que los métodos utilizados para segmentar imágenes no son capaces de discernir si una región corresponde a un solo núcleo celular o varios núcleos muy cercanos entre sí. Por ello, se ha ideado un método de detección y segmentación que usa métodos cásicos de segmentación de imágenes, como el método de *watershed*[54] o los métodos de agrupaciones, junto a un clasificador de regiones por inteligencia artificial mediante redes neuronales convolucionales (CNN).

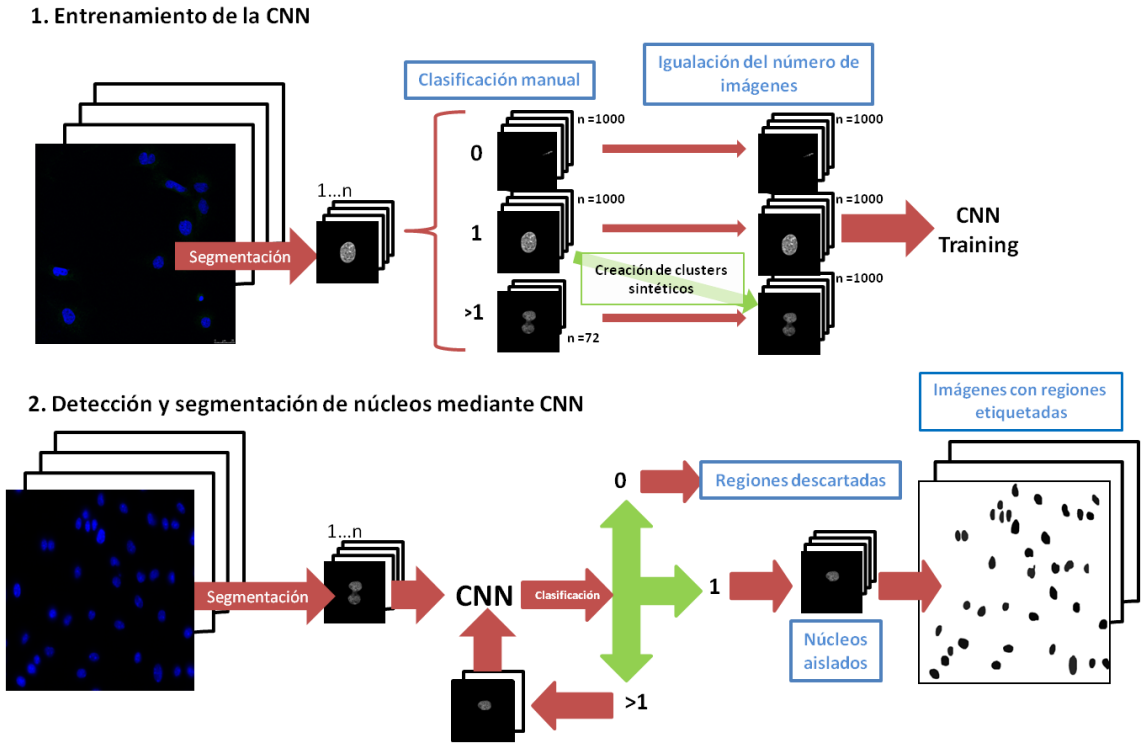


Figura 8: Modelo desarrollado para la detección y segmentación de núcleos celulares. (a) Entrenamiento de la CNN. (b) Detección y segmentación de núcleos mediante CNN.

Primeramente se re-entrena una CNN con arquitectura `VGG-16` para clasificar tres tipos de regiones: regiones mono-nucleares (etiquetadas como 1), regiones poli-nucleares (etiquetadas como >1) o regiones con artefactos no-nucleares (etiquetadas como 0). Una vez re-entrenada la CNN se diseñó el software (`SegNu`) que muestra el diagrama de la Figura 8.b. Primero, se parte de las imágenes de microscopía en bruto y se segmenta utilizado un método de segmentación límite. Segundo, las regiones segmentadas son clasificadas por la CNN, almacenando las regiones mono-nucleares y regiones con aglomerados nucleares y eliminando las regiones etiquetadas como regiones no-nucleares. Posteriormente, las regiones con aglomerados nucleares se segmentan y se vuelven a clasificar de manera recursiva. Finalmente, sólo tendremos regiones mono-nucleares, los cuales se utilizaran para generar una imagen de etiquetas.

En las subsiguientes secciones se detallará cada paso seguido por el software `SegNu`, desde la segmentación inicial de la imagen en bruto hasta la generación de los datos de

salida. El script del software `SegNu` se puede encontrar en el repertorio de `GitHub` (https://github.com/davco6/Trabajo-fin-de-Master/tree/master/SegNu).

4.2 Métodos de segmentación

Una de las partes principales del proyecto es la segmentación inicial de la imagen. Dicha segmentación se realiza utilizando un método de segmentación límite dada su robustez. Idealmente si se representa un histograma de intensidades de señal de una imagen de microscopía con núcleos celulares, se obtendrán dos máximos locales: como ocurre en la Figura 9 izquierda. Dichos máximos corresponden a los máximos de dos distribuciones: la distribución de intensidad del ruido de fondo y la distribución de las intensidades de las señales de los núcleos. En el caso de que las dos distribuciones estén lo suficientemente separadas se puede encontrar un mínimo por el cual se pueden separar en dos grupos las dos distribuciones. Este punto se puede hallar mediante la función `threshold_minimum` de la librería `scikit-image`.

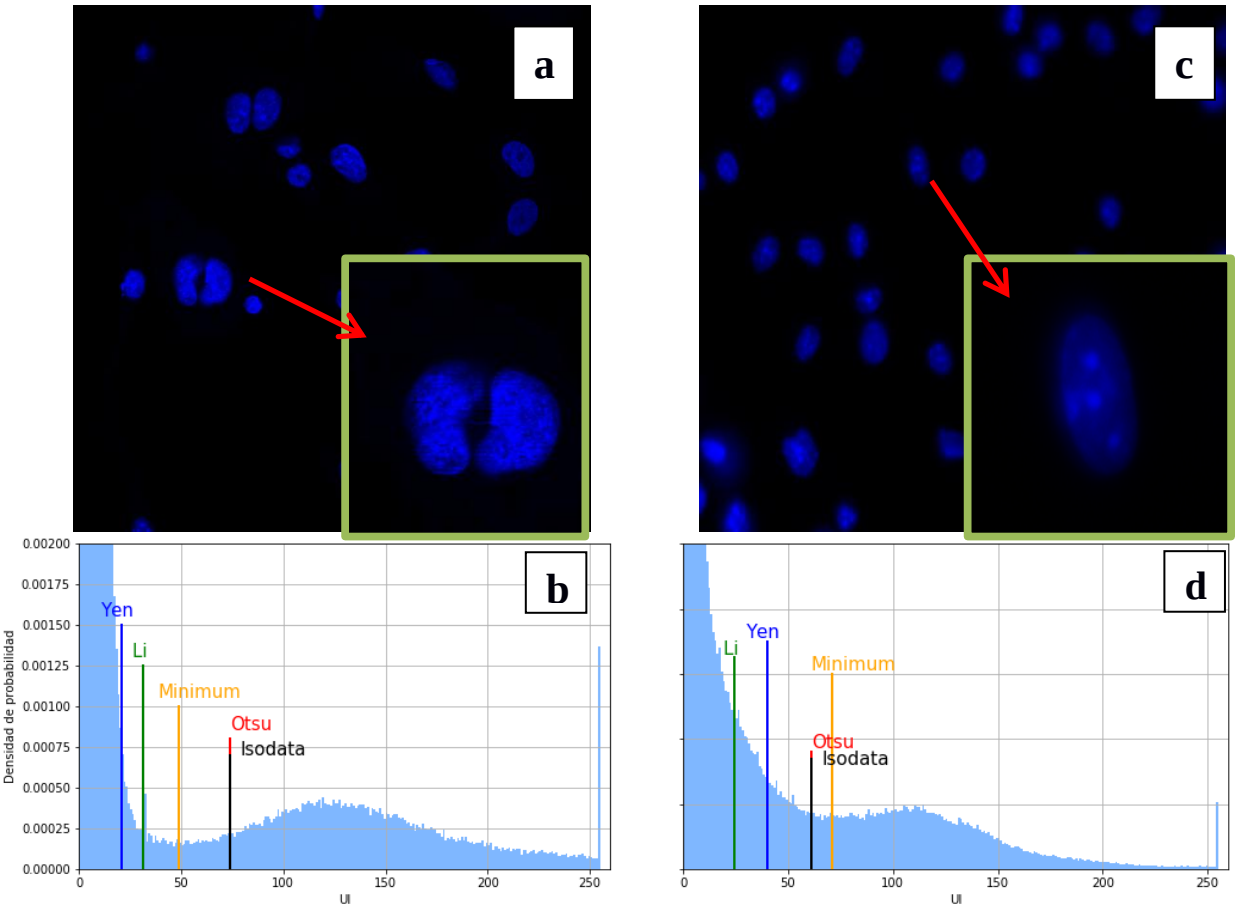


Figura 9: Imágenes de núcleos celulares y sus histogramas de intensidad.
Imagen de alta calidad (a) y bajo ruido de fondo y el histograma de intensidades (b) del canal azul (el mínimo está en 0 y el máximo en 255), se puede observar una separación entre la intensidad de ruido de fondo y la intensidad de los núcleos. (c) imagen con alto ruido de fondo y (d) histograma de intensidades, en el que no hay una separación clara entre el ruido de fondo y la distribución de intensidades de los núcleos.

Pero en el caso de que la imagen tenga mucho ruido de fondo –como por ejemplo en la Figura 9 derecha- no existe una separación clara entre la señal ruido y la señal de los núcleos. La función `threshold_minimum` puede tener problemas en encontrar un mínimo local y si lo encuentra no tiene porqué separar de manera óptima las dos

clases. Para este último caso, una solución es utilizar el método límite de Otsu[53] (función `threshold_otsu` del módulo `scikit-image`), éste asume que la imagen contiene dos clases de pixeles (ruido de fondo y señal de interés) y calcula el límite óptimo que separa las dos clases para que sus varianzas intra-clase sea mínima. En la práctica el método de Otsu es bastante agresivo en la segmentación pero en el caso de imágenes con mucho ruido tienen un buen resultado.

En el Pipeline 1 se muestra que valor límite usar dependiendo de si la distribución de ruido de fondo solapa o no con la distribución de intensidad de los núcleos. En resumen, se determina los valores límites mínimo y de Otsu. En el caso de que no se pueda hallar el valor límite mínimo o que éste sea mayor que el valor límite de Otsu estaremos en un caso con mucho ruido de fondo y por tanto se utilizará como valor límite el de Otsu. Por otra parte, si el valor límite mínimo es menor que el de Otsu se utilizará el valor límite mínimo como el valor para la segmentación.

```
1.  def initial_segmentation(image,...):
2.      from skimage.filters import threshold_otsu, threshold_minimum
3.      t_otsu = threshold_otsu(image)
4.      try:
5.          #Intentamos hallar el límite mínimo
6.          t_min = threshold_minimum(image)
7.      except RuntimeError:
8.          #Si no encuentra el mínimo, arbitrariamente hacemos t_min>t_otsu
9.          t_min = t_otsu + 1
10.
11.     if t_min <= t_otsu:
12.         threshold = t_min
13.     else:
14.         threshold = thresh_otsu
```

Pipeline 1. Inicio de la segmentación de la imagen.

4.3 Red Neuronal Convolucional

Una vez segmentadas las regiones candidatas por la segmentación inicial, se clasifican mediante CNN dichas regiones en tres grupos: poli-nucleares, mono-nucleares o artefactos no-nucleares (digitales o biológicos).

La CNN debe ser entrenada antes de llevar a cabo la clasificación de regiones. Y dado que entrenar dichas arquitecturas por primera vez requiere de una gran cantidad de imágenes y de elevados recursos del sistema, se utilizaron modelos de CNN pre-entrenados.

El entrenamiento se realizó mediante la utilización de la librería `Keras` sobre 3000 imágenes de núcleos celulares[i]: 1000 imágenes de cada grupo. Para el grupo de núcleos aislados y de regiones no-nucleares, se utilizaron 1000 imágenes obtenidas en bruto, y para el grupo que contiene aglomerados nucleares fueron utilizadas 71 imágenes obtenidas en bruto más 929 imágenes generadas sintéticamente. Por otra parte, para la validación de los modelos se utilizaron 210 imágenes: 70 imágenes de cada grupo.

El entrenamiento de la CNN debe hacerse con el mismo número de imágenes por clase para no cometer errores de tendencia. Dado que la clase que contiene agrupaciones

[i] Dichas imágenes fueron obtenidas por microscopía confocal como se describe en el apartado de métodos.

celulares tienen muy pocas imágenes en comparación con las otras dos clases, se diseñó un script por el cual se formaban imágenes de aglomerados nucleares a partir de imágenes con un solo núcleo celular.

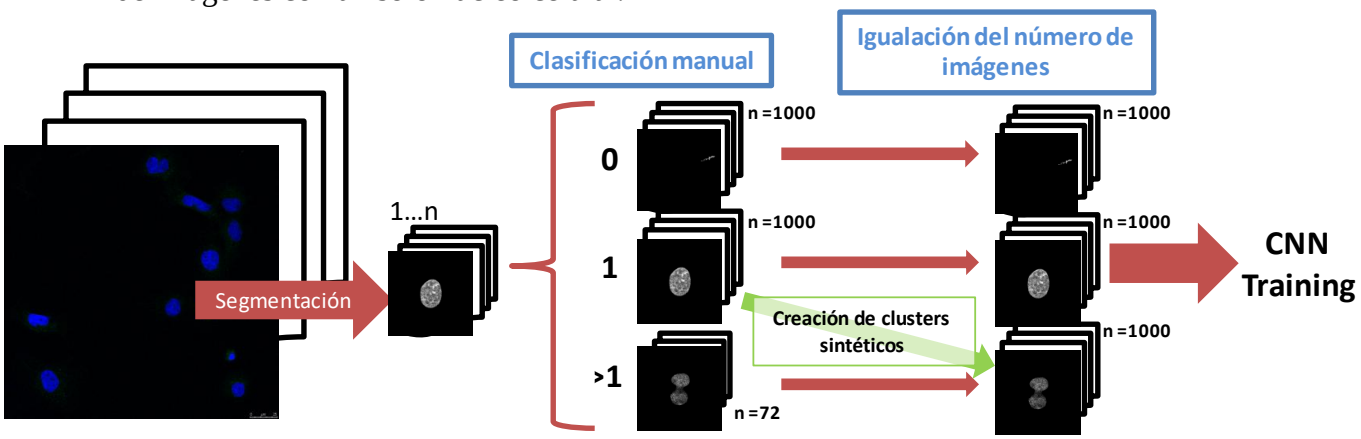


Figura 10. Entrenamiento de la CNN.

Generación de imágenes sintéticas con agregados nucleares:

Dado la importancia de obtener un mayor número de imágenes de agregados nucleares, se creó un script que generara sintéticamente estas agrupaciones (se puede encontrar en el repositorio de `GitHub` en https://github.com/davco6/Trabajo-fin-de-Master/blob/master/Synthetic_DB_v3.py). En resumen, primero se seleccionan al azar dos imágenes de núcleos mono-nucleares. Posteriormente, éstas se modifican al azar girando su eje, volteando la imagen, introduciendo un cizallamiento, cambiando su escalado y introduciendo un error gaussiano de señal. Una vez modificadas las dos imágenes unicelulares éstas se combinan, según el esquema de la Figura 11. Primeramente se selecciona al azar dos núcleos de la base de datos de imágenes con un solo núcleo (pasos **a** y **b**), cada imagen se modifica aplicando técnicas de *augmentation* como: rotación, cizallamento y giro al azar de los núcleos. En el paso **e**, se calcula el centroide del núcleo B y se traslada hasta el centro del parche. Posteriormente, en el paso **f** se elige al azar un punto limítrofe del núcleo B y se calcula la recta que pasa por el centroide y por el punto limítrofe seleccionado (paso **g**). El paso **h** calcula el punto donde se va a situar el centroide del núcleo A (marcado en naranja). Dicho punto se calcula de tal manera que se encuentre sobre la recta calculada a una distancia del punto limítrofe igual o menor al radio del núcleo A. Finalmente, en el paso **i** se superponen los núcleos A y B, situando el centroide del núcleo B en el punto calculado.

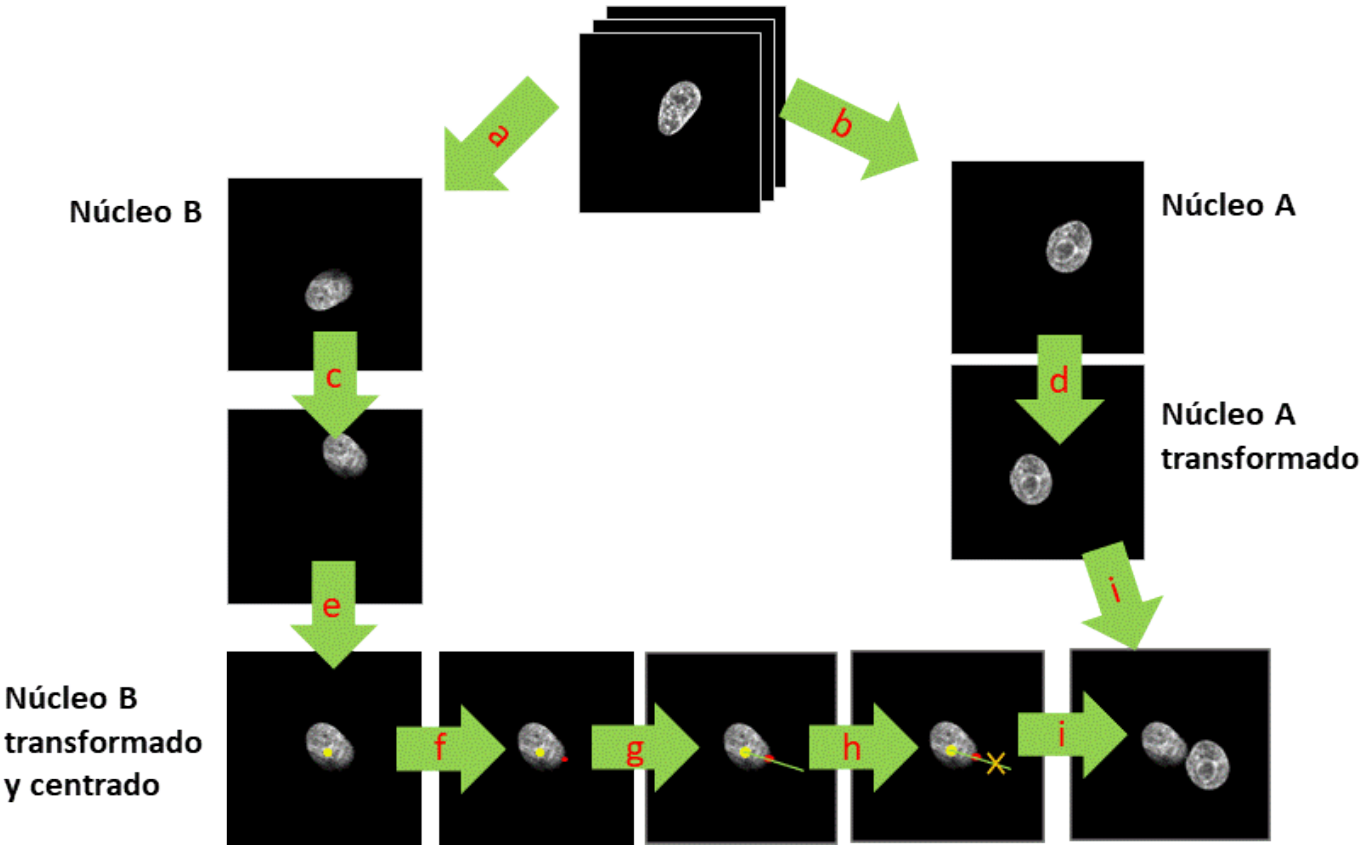


Figura 11. Pasos para la creación de imágenes sintéticas con agregados celulares.

Entrenamiento de la CNN:

El entrenamiento de la CNN se realizó mediante la utilización de una arquitectura `VGG-16` pre-entrenada, dado el alto coste computacional y la gran cantidad de imágenes que se requieren para entrenar una CNN por primera vez. La arquitectura `VGG-16` fue elegida debido al éxito que ha tenido dicha arquitectura en la clasificación de imágenes y por la sencillez en la modificación de los módulos por los que se compone.

Se usaron tres maneras diferentes para entrenar la CNN (el script desarrollado se muestra en el Anexo en el Pipeline 2), esquematizados en la Figura 12. El primer método de re-entrenamiento (Figura 12.A), se realizó mediante el "congelado" de todas las capas superiores de la red, conocido como el cuello de botella (*bottleneck*), y el entrenando de la última capa completamente conectada para la clasificación *softmax* de tres clases. El segundo método de re-entrenado (Figura 12.B), se realizó mediante *fine-tuning* del último módulo (que consta de tres capas convolucionales seguidas de una capa *max-pooling*), es decir se re-entrenó el ultimo modulo partiendo de los pesos obtenidos por el entrenamiento de la capa completamente conectada y utilizando una velocidad baja de aprendizaje (10^{-4}) con un gradiente estocástico descendente. Se utilizó una velocidad baja de aprendizaje para no cambiar en exceso los pesos aprendidos ya que se requiere su optimización.

El tercer método de re-entrenado (Figura 12.C), se realizó mediante *fine-tuning* de los dos últimos módulos, partiendo de los pesos obtenidos por el entrenamiento del último

modulo convolucional y utilizando una velocidad baja de aprendizaje (10^{-4}) con un gradiente estocástico descendente.

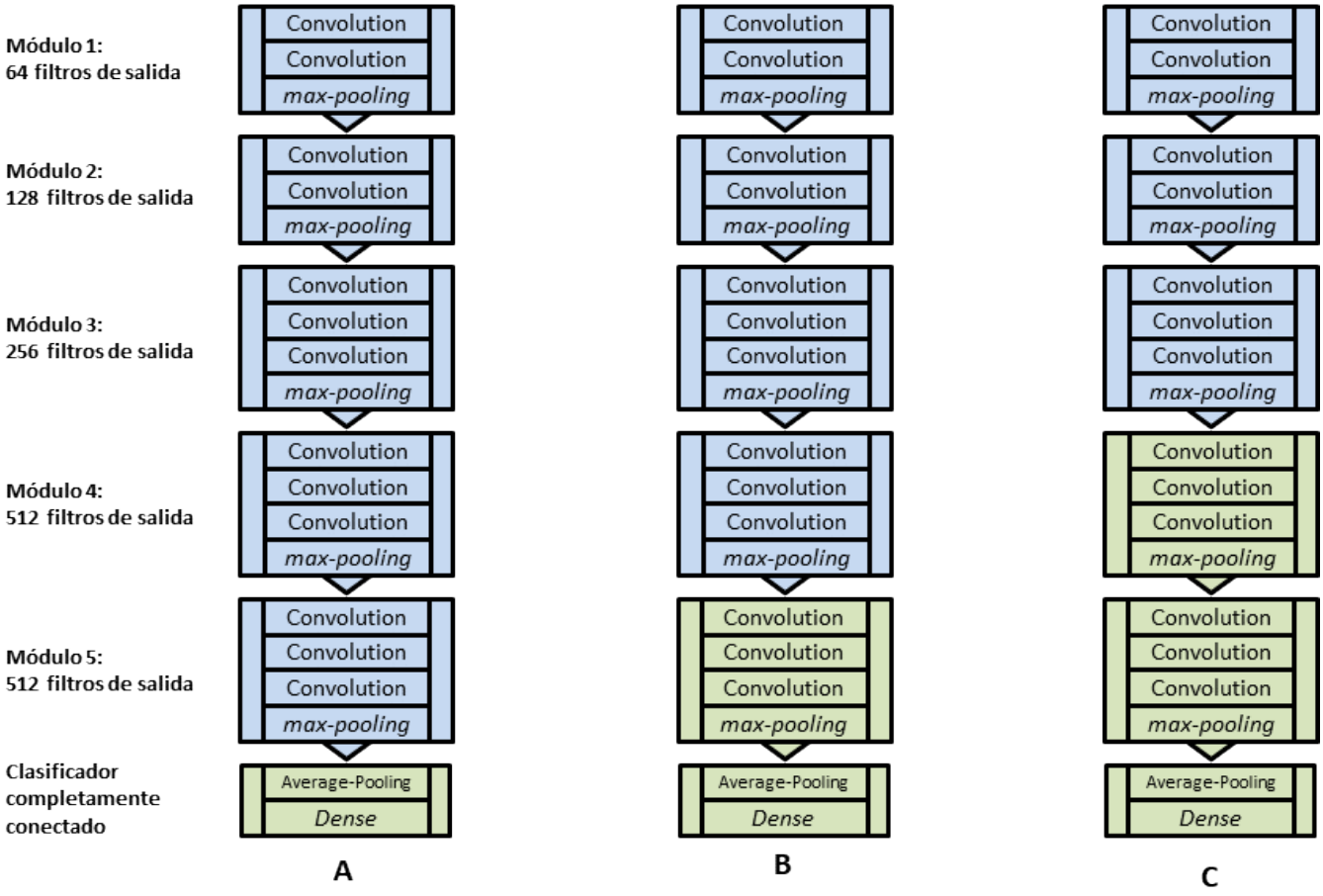


Figura 12. Re-entrenamiento de la arquitectura VGG-16. En azul las capas "congeladas" en verde las capas re-entrenadas. A) Entrenado de la capa completamente conectada. B) Re-entrenado del módulo 5 y de la capa completamente conectada. C) Re-entreando de los módulos 4 y 5 y de la capa completamente conectada.

Se puede observar en las gráficas de la Figura 13, como la arquitectura VGG-16 con los dos últimos módulos re-entrenados obtuvo el mejor resultado en la inferencia del conjunto de imágenes del test de validación. Aunque no hubo grandes diferencias entre el re-entrenado del último o dos últimos módulos del `VGG-16` (exactitud del 97,6% y 98,6% respectivamente en la inferencia de imágenes del test), sí que se observa como el procedimiento de *fine-tuning* es necesario para optimizar la CNN en comparación con sólo el re-entrenado del cuello de botella (exactitud del 95,7%).

En los resultados obtenidos por los modelos *fine-tuned* no se aprecia el *over-fitting* en los 50 epoch medidos, aunque sí se observa un estancamiento en la mejora de la exactitud. Por otra parte, la exactitud alcanzada en el test es significativamente mayor (97,6% y 98,6%) que en el entrenamiento (95% y 95,8%). Esto puede ser debido al bajo número de imágenes utilizadas para el test (70 por cada clase).

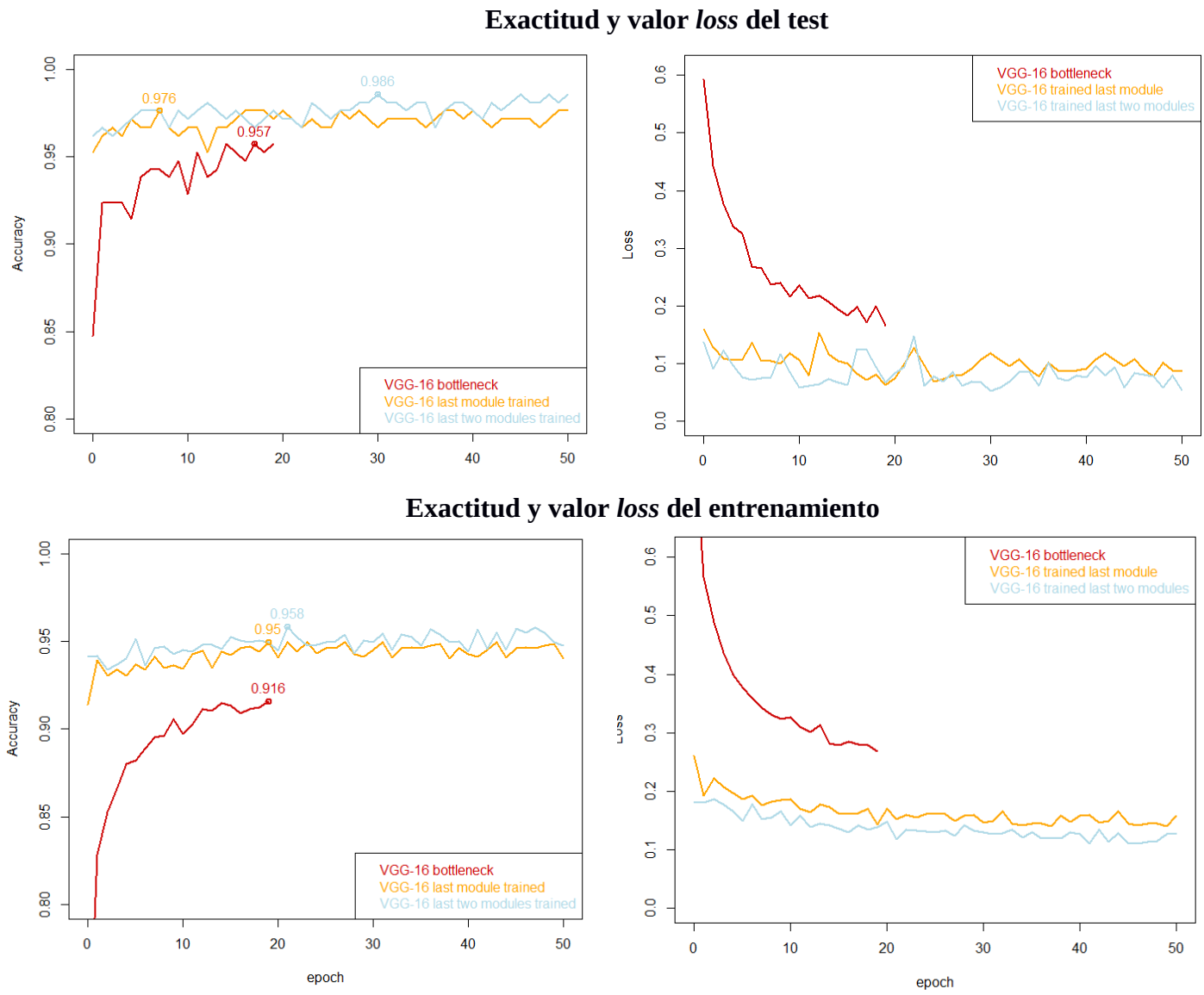


Figura 13. Gráficas de exactitud y valor de loss en los diferentes epoch en el entrenamiento y en el test.

4.4 Generación de parches

La aplicación de un método de escalado de imágenes es necesaria para que la herramienta desarrollada pueda ser utilizada en diferentes conjuntos de imágenes de microscopía con diferente tipo de enfoque y resolución.

Dado que se va a utilizar una misma CNN pre-entrenada independientemente del tamaño del núcleo de la célula, las imágenes a analizar deben presentar las mismas características básicas que las imágenes utilizadas en el entrenamiento. Para el entrenamiento se utilizaron parches con imágenes de núcleos celulares con una resolución de 192 x 192 pixeles y con un ratio medio de área del parche frente al área del núcleo segmentado de 13.75 (cada parche estaría cubierto con 13.75 núcleos).

Para que las imágenes a analizar por primera vez presenten las mismas características básicas que las de entrenamiento se deben normalizar las primeras, para que se cumplan las características básicas. Para llevar a cabo la normalización de los parches del test, primeramente se halla el área de las regiones segmentadas y se obtiene el tamaño de idóneo del parche. Posteriormente se obtienen los parches de todas las regiones segmentadas con el tamaño de parche calculado (n x n) y se escalan hasta una resolución de 192 x 192.

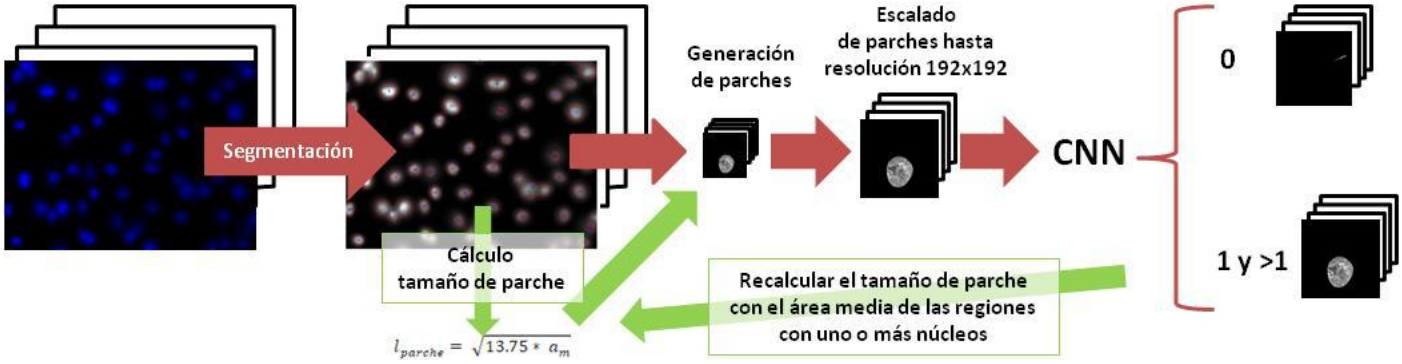


Figura 14. Método de escalado de parches con imágenes con núcleos celulares. a_m: área media de las regiones segmentadas, l_{parche}: longitud en pixeles de los lados del parche calculado.

Uno de los problemas que se pueden presentar es que en el caso del análisis de imágenes de microscopía con mucho ruido, se generaran una gran cantidad de regiones candidatas que realmente pertenecen al ruido de fondo. Para resolver dicho problema se llevó a cabo una modificación del primer método de escalado. En resumen, el método -esquematizado en la Figura 14- segmenta la imagen por un método límite, halla el área de las regiones segmentadas y calcula el tamaño del parche idóneo. Posteriormente se generan los parches con las regiones candidatas y se escalan hasta una resolución de $192\,x\,192$ para ser clasificadas por la red neuronal convolucional (hasta aquí el método es idéntico al caso anterior). Aquellas regiones que se clasifiquen como 0 (es decir, que no contengan núcleos celulares) se eliminan y se vuelve a recalcular la media de las áreas de todas las regiones que presenten núcleos celulares. Finalmente, se recalcula el nuevo tamaño de parche definitivo.

El script creado para tal fin se puede encontrar como una definición escrito en Python llamada **`patch_tune`** del archivo `SegNu_def.py` (que contiene las definiciones necesarias para el software `SegNu`).

4.5 Segmentación recursiva de agregados nucleares

Una vez clasificados los parches con núcleos celulares por la CNN, las regiones que hayan sido clasificados como agregados nucleares se segmentarán para volver a ser clasificados por la CNN. Éste método recursivo permite separar agrupaciones celulares que los métodos simples de segmentación no son capaces de separar, dado que la CNN guía en que regiones se debe realizar la segmentación.

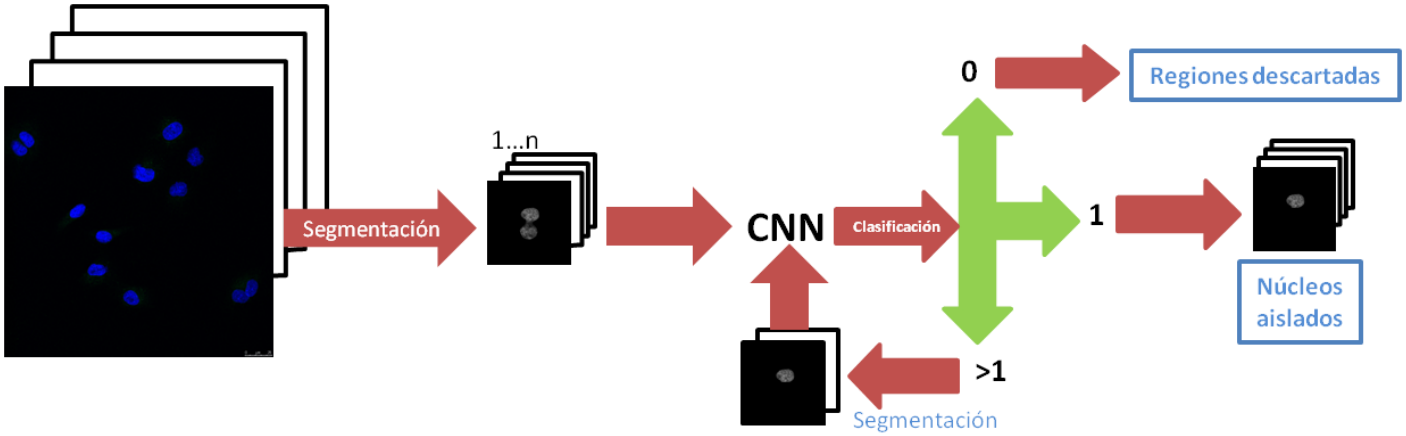


Figura 15. Modelo creado para clasificar y segmentar imágenes con núcleos celulares.

Se utilizaron los siguientes métodos de segmentación: método de inundación[54] (*watershed*) con un valor de sigma de 6, método de inundación con un valor de sigma de 12, método de agrupación *K-means++*[19], método de agrupación por aglomeración con enlace de Ward[52] (AC) y método de agrupación mixto gaussiano *(Gaussian)*.

En total, se analizaron siete imágenes con diferentes tipos de agrupaciones nucleares.

Mientras que con el método de segmentación por inundación se pueden obtener más de dos segmentaciones, con los métodos de agrupación al fijarse el número de *cluster* en dos: siempre obtendremos 2 clases.

Por otra parte, se puede observar que el método de *watershed* produce diferentes segmentaciones dependiendo del parámetro sigma o distancia entre elementos. Por ello es importante conocer el radio de los núcleos celulares, que se determina en la fase del optimizado del tamaño del parche, por el cual se clasifican inicialmente las áreas clasificadas como núcleos unicelulares y se promedia su radio.

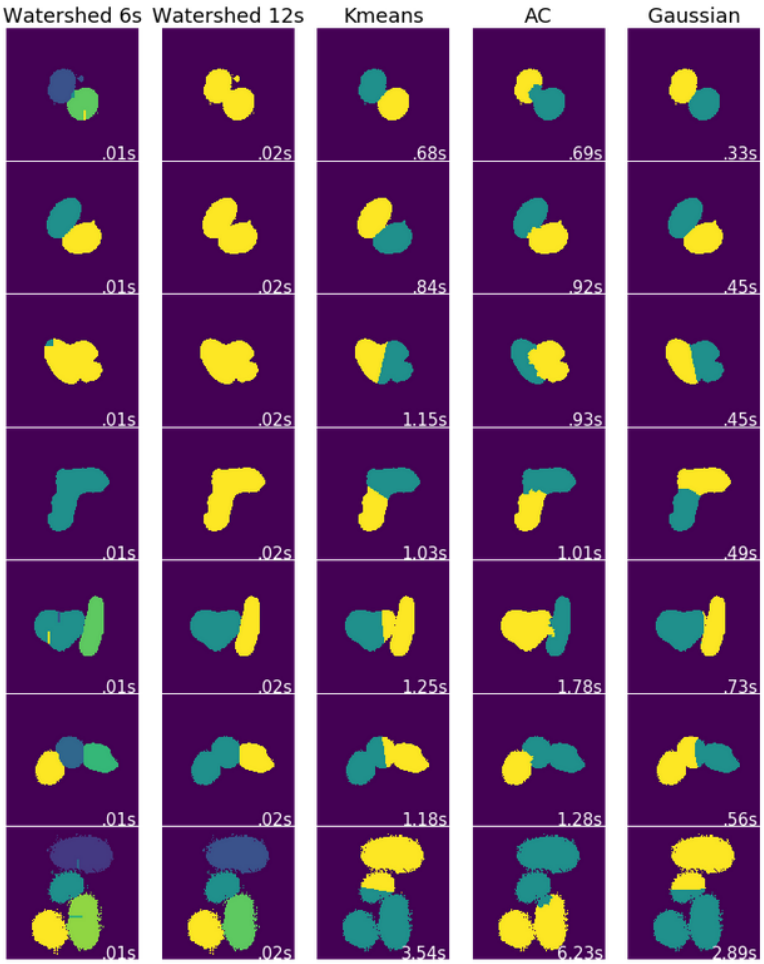


Figura 16: Comparación de diferentes métodos de segmentación en agrupaciones celulares.

4.6 Estructura del software:

El software `SegNu` fue escrito en `Python 3.0` y su uso sólo esta soportado para Ubuntu. Puede ser descargado del repositorio de `GitHub`: https://github.com/davco6/Trabajo-fin-de-Master. Además, se creó un cuaderno en `Jupyter` (`SegCNN-v3.ipynb`) que describe paso por paso el funcionamiento de `SegNu`. Para el correcto funcionamiento del software es necesario cargar el archivo de pesos de la CNN.

Para utilizar el software `SegNu` es necesario que en el sistema estén instalados los módulos externos descritos en la Tabla 4, se recomienda la instalación de los módulos vía `Conda` (https://conda.io/docs/).

Módulo	Pre-instalado con Conda	Descripción
NumPy	**Sí**	**Módulo fundamental para algebra lineal**
Scikit-learn	Sí	Utiliza herramientas para el análisis de datos, en especial se utilizan los métodos de agrupado
Scikit-image	**Sí**	**Colección de algoritmos para el procesado de imágenes**
Keras	No	Conjunto de funciones y procedimientos para la implementación de redes neuronales
TensorFlow	**No**	**Librería ampliamente utilizada en la programación en machine learning**

Tabla 4. Módulos externos utilizados en SegNu

El acceso al software SegNu se hace mediante vía de comandos, utilizando los argumentos descritos en la Tabla 5.

Argumento(s)	Valor por defecto	Descripción
-h **--help**	-	**Muestra los argumentos permitidos**
-i **--ipath**	-	Carpeta donde se encuentran las imágenes a analizar
-o **--opath**	Input_path	**Carpeta donde se guardarán los resultados del análisis**
-t **--segmentation**	watershed	Tipo de segmentación utilizada: watershed: segmentación watershed clustering_AC: segmentación por agregados aglomerativos clustering_gauss: segmentación por agregados Gaussianos mixtos only_limit_threshold: segmentación límite sin usar CNN watersehd_no_cnn: segmentación watershed sin usar CNN
-s **--save_images**	False	**Si "True", se guardaran las imágenes originales marcadas en rojo las regiones identificadas como núcleos.**

Tabla 5. Argumentos de SegNu

Una vez analizadas las imágenes de microscopía por SegNu, se creará una carpeta con los resultados. En dicha carpeta se creará un archivo llamado "Results.txt" el cual muestra el número de núcleos detectados en cada imagen analizada. Además, se creara una carpeta llamada "Instances_images" donde se guardaran las "imágenes de instancias", es decir, éstas son las imágenes con la misma resolución que la imagen original pero con etiquetas en las regiones detectadas como núcleos celulares. Además también se pueden obtener (mediante la opción –save_images) las imágenes originales marcados en rojo las núcleos celulares detectados.

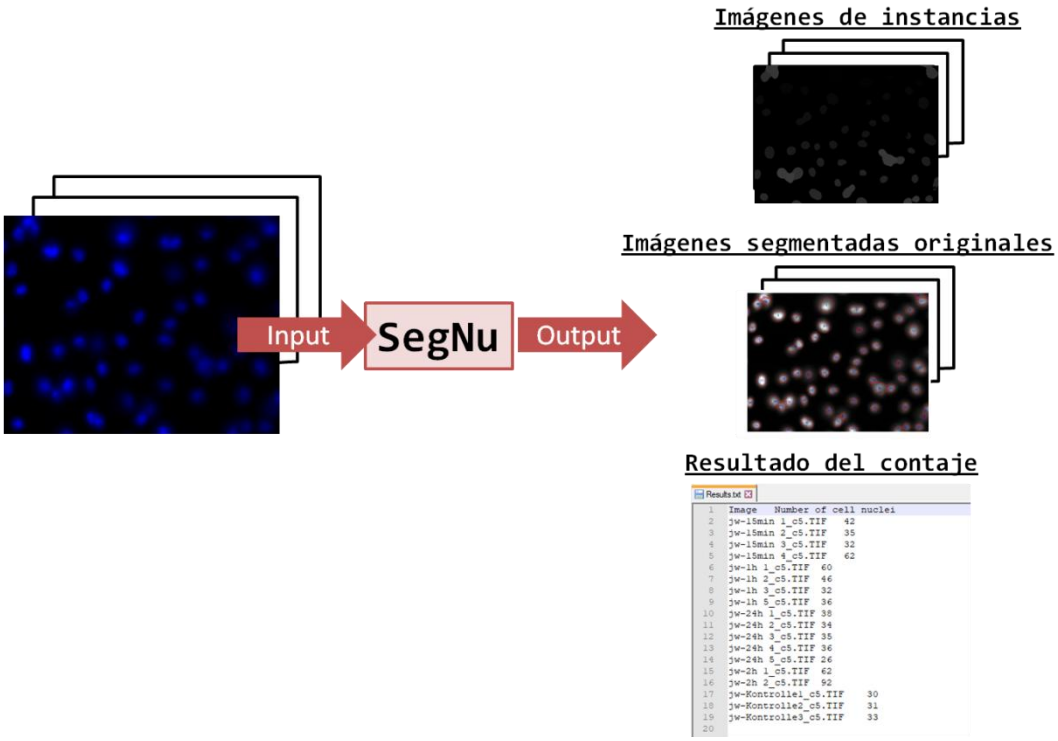


Figura 17. Input y output generado por SegNu tras el análisis de imágenes de microscopía.

5. Resultados

La evaluación de la eficacia en la detección y segmentación de núcleos celulares del software `SegNu` fue realizada en dos conjuntos de imágenes de microscopía. La primera base de datos está compuesta por 16 imágenes de microscopía no utilizadas previamente. La segunda base de datos son un conjunto de imágenes de referencia del *dataset* BBBC020 (que pertenecen a imágenes públicas de la Broad Bioimage benchmark Collection) creadas por Ljosa *et al*[25]. Ésta base de datos además incorpora imágenes con todos los núcleos segmentados reales de cada imagen original.

Los métodos de segmentación analizados fueron los siguientes: método de Otsu[53] sin un ulterior filtrado (**Otsu**), método de Otsu seguido de un método de *watershed*[54] (**WS**), método de clasificación por CNN con segmentación por el método de agrupación aglomerativa con enlace de Ward[52] (**AC-CNN**), método de clasificación por CNN seguido de una segmentación por el método Gaussiano mixto (**Gauss-CNN**) y método de clasificación por CNN seguido de una segmentación *watershed* (**WS-CNN**).

Análisis de imágenes obtenidas por microscopía confocal de células C6:

Se analizaron 16 imágenes obtenidas por un microscopio confocal de preparaciones de cultivos celulares de la línea celular C6, tal y como se describe en el apartado de métodos. Dichas imágenes fueron adquiridas de la misma manera que las imágenes utilizadas para el entrenamiento de la CNN, pero no forman parte de las imágenes utilizadas para el entrenamiento o el test de la CNN. Además, de cada imagen se realizó un recuento manual para calcular la eficacia en la detección de núcleos del modelo. Para ello fue calculado el error relativo del contaje de núcleos (**CRE**), éste calcula la diferencia entre el número de núcleos detectados -por uno de los métodos- con respecto al número de núcleos reales, en el caso que ocurra sobre-segmentación el valor será positivo, y si existe infra-segmentación el valor será negativo. Es decir, el CRE es un parámetro que calcula la probabilidad estadística por el cual un determinado núcleo celular no se detecte (en el caso de que sea negativo) o que sea sobre-segmentado (si es positivo su valor).

Los resultados obtenidos del cálculo del CRE de cada método de segmentación se han resumido en la Tabla 6. Así, se observa como el método de **Otsu** tiende a infra-segmentar (los intervalos de confianza al 95% son inferiores al 0%) mientras que el método de **WS** clásico tiende a sobre-segmentar los núcleos (los intervalos de confianza al 95% son superiores al 0%). Mientras que los métodos que utilizan CNN obtienen resultados menos extremos. Aunque, por una parte el método de **WS-CNN** infra-segmenta los núcleos y por otra parte los métodos de segmentación por aglomerados (**AC-CNN** y **Gauss-CNN**) ligeramente sobre-segmenta los núcleos en un 3% de las veces (aunque los valores de confianza al 95% sí contienen al 0%)

También se ha calculado el tiempo de procesado por imagen de cada método de segmentación. Éste hace referencia al tiempo transcurrido desde que se inicia el procesamiento de la imagen hasta que se generan los archivos de salida, dicho procesamiento se ha realizado con el equipo descrito en métodos pero sin ser utilizada la GPU. Se puede observar como los métodos clásicos de segmentación son apreciablemente más rápidos que los métodos que utilizan la CNN debido al coste

computacional por la inferencia de la CNN. A pesar de ello, los tiempos de procesado por imagen son aceptables y pueden ser realizados en equipos de sobremesa que no contengan GPU incorporada.

	Otsu	WS	AC-CNN	Gauss-CNN	WS-CNN
CRE	-7% (-10%; -3%)	+10% (+3%; +16%)	+3% (-2%; +8%)	+3% (-3%; +9%)	-4% (-7%; -2%)
Tiempo de procesado por imagen	0,57s	4,28s	18,64s	16,80s	18,25s

Tabla 6. Resultados obtenidos por los diferentes métodos de detección-segmentación en el análisis de imágenes de núcleos de células C6. Entre paréntesis los intervalos de confianza al 95% de la distribución *t-student* calculada.

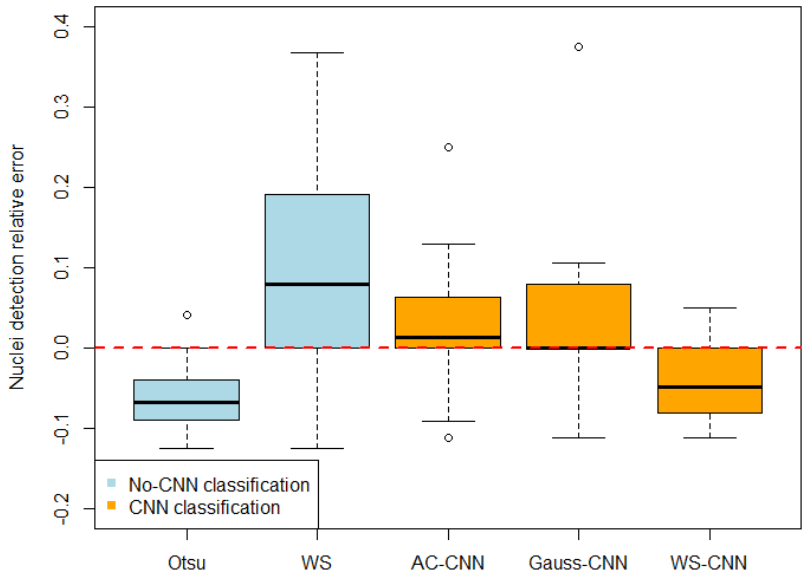


Figura 18. Boxplot de los CRE de los métodos de segmentación-detección de núcleos.

Base de datos de referencia BBC020:

Con el fin de comprobar la bondad de la robustez de los modelos diseñados se analizaron las imágenes de la base de datos de referencia BBC020. Tal y como se describe en la publicación de Ljosa *et al*[25], las imágenes fueron obtenidas de cultivos celulares de macrófagos derivados de la médula ósea de ratones C57BL/6. Los núcleos celulares fueron marcados con DAPI. La segmentación de células en estas imágenes presenta un reto ya que las células presentan un aspecto irregular y algunos macrófagos se tocan y superponen entre sí.

El análisis fue realizado con 17 imágenes del *dataset* BBBC020 (aquellas que contenían imágenes con regiones etiquetadas) tomadas con un objetivo de 20x y procesadas digitalmente obteniendo una resolución de 1388 x 1040 pixeles. Por otra parte, las imágenes que fueron utilizadas en el entrenamiento del modelo desarrollado

en este proyecto se tomaron con un objetivo de 60x y tienen una resolución de 1024 x 1024 pixeles. Por tanto, el estudio de este *dataset* nos informará sobre eficacia del escalado utilizada por el método.

Por otra parte, se utilizaron dos aproximaciones para el análisis estadístico de la bondad del ajuste en la detección y segmentación. Para calcular la eficacia en la detección de núcleos celulares se calculó el error relativo del contaje de núcleos (**CRE**). Observando un comportamiento similar al análisis de imágenes de células C6, es decir, los métodos que mejor resultado dieron fueron los que contienen métodos de clasificación por CNN y métodos de segmentación aglomerativa (**AC-CNN** y **Gauss-CNN**), siendo la probabilidad media de infrasegmentar un núcleo celular menor del 1%. Se puede observar gráficamente en el *boxplot* de la Figura 19 la distribución de los errores relativos en el contaje de núcleos celulares de cada método utilizado. De dichos resultados se puede extraer que es efectivo escalar la imagen para acomodarla a la misma resolución utilizada en el entrenamiento.

Para analizar la exactitud de la segmentación, las funciones más utilizadas son: exactitud por pixel (**PA**), exactitud media por pixel (**MPA**) e intersección media sobre unión (**MIoU**). Los resultados obtenidos se resumen en la Tabla 7.

El cálculo de la función **MIoU** es considerada como la medida estándar para evaluar métodos de segmentación. Se puede observar como los métodos que mejor resultado obtienen son los métodos aglomertivos con clasificación por CNN (**AC-CNN** y **Gauss-CNN**) con una exactitud del 0,73 frente a la exactitud del 0,68 obtenido por el método de segmentación límite de Otsu.

Los resultados del análisis de la bondad del ajuste en la segmentación son significativamente más bajos que los obtenidos del error medio del contaje de núcleos. Esto puede ser debido a que el método de detección y clasificación es muy eficiente en comparación con el método de segmentación. Esto puede ser debido a que los métodos de segmentación utilizados sólo tienen en consideración la superficie ocupada de los núcleos y no el contenido, ya que la segmentación ocurre por el segmento más corto y no por ello el segmento real que divide en dos al aglomerado nuclear. Una solución al problema podría ser la segmentación de instancias de los parches que contienen aglomerados nucleares, pero dicha solución está más allá del alcance del proyecto presentado. Aún así, los modelos presentados que engloban la segmentación aglomerativa y clasificación mediante CNN son efectivos y computacionalmente rápidos en la detección y segmentación de núcleos celulares en imágenes de microscopía.

	Otsu	WS	AC-CNN	Gauss-CNN	WS-CNN
CRE	-8% (-13%,-3%)	+5% (-2%; +11%)	-0.4% (-3%, +2%)	0.0% (-3%, +3%)	-6% (-9%, -2%)
PA	0,97 (0,96; 0,98)	0,97 (0,96; 0,98)	0,98 (0,97; 0,98)	0,98 (0,97; 0,98)	0,97 (0,96; 0,99)
MPA	0,84 (0,79; 0,89)	0,70 (0,65; 0,75)	0,89 (0,86; 0,92)	0,89 (0,85; 0,92)	0,88 (0,84; 0,91)
MIoU	0,68 (0,62; 0,74)	0,58 (0,53; 0,63)	0,73 (0,69; 0,78)	0,73 (0,69; 0,77)	0,72 (0,68; 0,77)

Tabla 7. Resultados obtenidos por los diferentes métodos de detección-segmentación en el análisis de imágenes de referencia BBC020. Entre paréntesis los intervalos de confianza al 95% de la distribución *t-student* calculada.

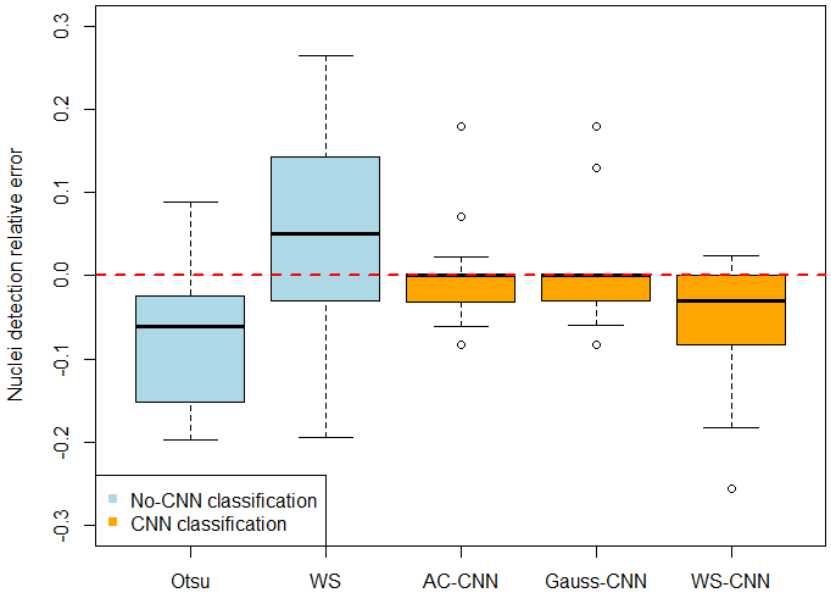


Figura 19. Boxplot de los CRE de los métodos de segmentación-detección de núcleos.

6. Conclusiones

- Se ha desarrollado un modelo de detección y segmentación de núcleos celulares eficaz, flexible al escalado y computacionalmente rápido. Para ello, se han utilizado conjuntamente un método de segmentación límite, una clasificación de los núcleos por redes neuronales artificiales y una segmentación recursiva que segmenta agrupaciones nucleares.

- El entrenamiento de la red neuronal artificial se realizó utilizando imágenes de tres regiones diferentes: regiones mono-nucleares, regiones poli-nucleares y artefactos no-nucleares. Debido al bajo número de regiones con agregados nucleares (en comparación con los núcleos aislados), se generaron imágenes sintéticas con agregados nucleares.

- Para la clasificación de núcleos celulares, se utilizó una red neuronal con arquitectura VGG-16 previamente pre-entrenada. El mejor resultado se obtuvo tras entrenar los dos últimos módulos de la red VGG-16 con una exactitud en la inferencia de las imágenes del test del 98,6%.

- La comprobación de la bondad en la robustez de los modelos desarrollados se realizó mediante el análisis de la base de datos de referencia BBC020. El análisis dio como resultado que los métodos de segmentación aglomerativa ayudados por redes neuronales artificiales (AC-CNN y Gauss-CNN) obtuvieron un resultado significativamente mejor que los métodos clásicos de segmentación (método de segmentación límite de Otsu y de *watershed*) tanto en la detección de los núcleos celulares como en la eficacia en la segmentación.

- Se desarrolló un software escrito en Python que analiza imágenes de núcleos celulares, dando como resultado el contaje del número de núcleos detectados y las imágenes de los elementos segmentados, con la misma resolución que la imagen original.

- En líneas generales, la detección de núcleos celulares fue satisfactoria pero no así la segmentación de aglomerados nucleares. Por tanto, hay margen para el optimizado de la herramienta diseñada. Una forma de mejora sería la implementación de un algoritmo de segmentación de instancias guiado mediante redes neuronales convolucionales.

- Los siguientes pasos a realizar para continuar con el proyecto son: primeramente darle una mayor exposición pública mediante una publicación científica, y por otra parte facilitar el uso del software a los usuarios mediante el diseño de una página web con interfaz sencilla y con el software SegNu embebido.

7. Anexos

Glosario

AC-CNN: método de segmentación por agrupación algomerativa con clasificación con redes neuronales artificiales
CNN: Red neuronal convolucional
ECN: Error relativo del contaje de núcleos
FCN: Red neuronal completa
Gauss-CNN: método de segmentación por agrupación gaussiana mixta con clasificación con redes neuronales artificiales
FISH: Fluorescencia por hibridación *in situ*
MIoU: Intersección media sobre unión
MPA: Exactitud media por pixel
PA: Exactitud por pixel
VGG: Visual Geometry Group
WS: método de *watershed*

Pipeline 2. Entrenamiento de la CNN con arquitectura VGG-16

```
"""
Modificación del script desarrollado por Abner Ayala-Acevedo
https://www.kaggle.com/abnera/transfer-learning-keras-xception-cnn

Estructura del directorio de trabajo:
          1000 imágenes de restos celulares situados en la carpeta cell_debris
          1000 imágenes de imágenes con un solo núcleo
          1000 imágenes de imágenes con conglomerados de núcleos


data/
    train/
        cell_debris/
            001.jpg
            002.jpg
            ...
        I/
            001.jpg
            002.jpg
            ...
        II/
            001.jpg
            002.jpg
            ...
    validation/
        cell_debris/
            001.jpg
            002.jpg
            ...
        I/
            001.jpg
            002.jpg
            ...
        II/
            001.jpg
            002.jpg
            ...

"""
import numpy as np
import sys
import os
from keras.layers import *
from keras.optimizers import *
from keras.applications import *
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, EarlyStopping, CSVLogger
from keras import backend as k



top_model_weights_path = 'bottleneck_fc_model.h5'
# Dimensiones de la imagen
img_width, img_height = 192, 192

nb_classes = 3  # Número de clases
based_model_last_block_layer_number = 15  #Bloqueo del entrenamiento de las capas. Para el modelo VGG16 en e
l que se entrenan sólo el último modulo el valor es 15.

batch_size = 32
nb_epoch = 100


def train(train_data_dir, validation_data_dir, model_path):
    # pre-entrenamiento del modelo VGG16
    base_model = VGG16(input_shape=(img_width, img_height, 3), weights='imagenet', include_top=False)

    # Bloqueo del las capas de más arriba
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    predictions = Dense(nb_classes, activation='softmax')(x)
    model = Model(base_model.input, predictions)
    print(model.summary())

    #Se congelan todas la capas del modelo
    for layer in base_model.layers:
        layer.trainable = False

    #Entrenamiento de la base de datos de imágenes
    train_datagen = ImageDataGenerator(rescale=1. / 255,
                                       width_shift_range=0.2,
                                       height_shift_range=0.2,
                                       rotation_range=30,
                                       shear_range=0.2,
                                       zoom_range=0.5,
```

```
                                        cval=transformation_ratio,
                                        horizontal_flip=True,
                                        fill_mode="nearest",
                                        vertical_flip=True)

    validation_datagen = ImageDataGenerator(rescale=1. / 255)

    os.makedirs(os.path.join(os.path.abspath(train_data_dir), '../preview'), exist_ok=True)
    train_generator = train_datagen.flow_from_directory(train_data_dir,
                                                        target_size=(img_width, img_height),
                                                        batch_size=batch_size,
                                                        class_mode='categorical')

    validation_generator = validation_datagen.flow_from_directory(validation_data_dir,
                                                                  target_size=(img_width, img_height),
                                                                  batch_size=batch_size,
                                                                  class_mode='categorical')

    model.compile(optimizer='nadam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    top_weights_path = os.path.join(os.path.abspath(model_path), 'top_model_weights.h5')


    callbacks_list = [
        ModelCheckpoint(top_weights_path, monitor='val_acc', verbose=1, save_best_only=True),
        EarlyStopping(monitor='val_acc', patience=10, verbose=0),
        CSVLogger('bottleneck_vgg16.log')
    ]

    # Entenamiento del cuello de botella
    model.fit_generator(train_generator,
                        steps_per_epoch=3000/batch_size,
                        nb_epoch=nb_epoch / 5,
                        validation_data=validation_generator,
                        nb_val_samples=210/batch_size,
                        callbacks=callbacks_list)


    print("\nStarting to Fine Tune Model\n")


    # Cargamos los "mejores" pesos obtenidos mediante el entrenamiento del cuello de botella
    model.load_weights(top_weights_path)

    # Para el entrenamiento del último módulo de la arquitectura VGG16 congelamos todas las capas convolucio
nales y de pooling menos las del último módulo.
    for layer in model.layers[:based_model_last_block_layer_number]:
        layer.trainable = False
    for layer in model.layers[based_model_last_block_layer_number:]:
        layer.trainable = True

    # Compilamos el modelo con una menor velocidad de aprendizaje
    model.compile(optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])


    final_weights_path = os.path.join(os.path.abspath(model_path), 'model_weights.h5')
    callbacks_list = [
        ModelCheckpoint(final_weights_path, monitor='val_acc', verbose=1, save_best_only=True),
        EarlyStopping(monitor='val_loss', patience=20, verbose=0),
        CSVLogger('finetuning.log')
    ]


    model.fit_generator(train_generator,
                        steps_per_epoch=3000/batch_size,
                        nb_epoch=nb_epoch,
                        validation_data=validation_generator,
                        nb_val_samples=210/batch_size,
                        callbacks=callbacks_list)


    model_json = model.to_json()
    with open(os.path.join(os.path.abspath(model_path), 'model.json'), 'w') as json_file:
        json_file.write(model_json)


if __name__ == '__main__':
    if not len(sys.argv) == 3:
        print('Los argumentos deben estructurarse como:\npython code/train.py <data_dir/> <model_dir/>')
        print('Ejemplo: python code/train.py data/cells/ model/cells/')
        sys.exit(2)
    else:
        data_dir = os.path.abspath(sys.argv[1])
        train_dir = os.path.join(os.path.abspath(data_dir), 'train')
        validation_dir = os.path.join(os.path.abspath(data_dir), 'validation')
        model_dir = os.path.abspath(sys.argv[2])

```

```
176.         os.makedirs(os.path.join(os.path.abspath(data_dir), 'preview'), exist_ok=True)
177.         os.makedirs(model_dir, exist_ok=True)
178. 
179.     train(train_dir, validation_dir, model_dir)
180.     k.clear_session()
```

8. Bibliografía

1 Imagen de https://www.proteinatlas.org.
2 Vanel N, Vierling V, Kreshak J, Gambarotti M, Cochi S, Tranfaglia C, Vanel D. Clinical Sarcoma Resarch 2011,1:9.
3 Kaern M, Elston TC, Blake WJ, Collins JJ (2005) Stochasticity in gene expression: From theories to phenotypes. Nature Reviews Genetics 6: 451–464.
4 García Rojo M, Punys V, Slodkowska J, Schrader T, Daniel C, Blobel B. Digital pathology in europe: coordinating patient care and research efforts. Stud. Health Technol. Inform. 2009; 150:997–1001. [PubMed: 19745463]
5 Robust Nucleus/Cell Detection and Segmentation in Digital Pathology and Microscopy Images: A Comprehensive Review. Xing F et al. Rev Biomed Eng. 2016; 9:234-263
6 A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. Maurer CRJ, Ahmad M, Swamy M. Trans. Pattern Anal. Mach. Intell. 2003. 25(2):265-270.
7 Morphological Image Analysis: principles and Applications. Soille P. Springer-Verlag. 1999.
8 Overlapping cell nuclei segmentation using a spatially adaptive active physical model. Plissiti M, Nikou C. IEEE Trans. Image Process. 2012, 21(11): 4568-4580.
9 Automated tool for the detection of cell nuclei in digital microscopic images: application to retinal images. Byun J, Verardo MR, Sumengen B, Lewis G, Manjunath BS, Fisher SK. Mol. Vis. Aug. 2006 12:949-960.
10 Automated nucleus and cytoplasm segmentation of overlapping cervical cells. Lu Z, Carneiro G, Bradley AP. Int. Conf. Med. Image Comput Comput Assist Intervent. 2013; 8149:452-460.
11 Segmentation of haematopoeitic cells in bone marrow using circle detection and splitting techniques. Ramesh N, Salama M, Tasdizen T. Int Symp Biomed Imag. 2012: 206-209.
12 Marker-controlled watershed segmentation of nuclei in H&E stained breast cáncer biopsy images. IEEE. Veta M, Huisman A, Viergever M, Van Diest PJ, Pluim JPW. Int. Symp Biomed Imag. 2011:618-621.
13 Automatic myonuclear detection in isolated single muscle fibers using robust ellipse fitting and sparse representation. Su H, Cing F, Lee JD, Peterson CA, Yang L. Trans Comput Biology Bioinfo. 2014; 11(4):714-726.
14 Automatic cell detection in bright-field microscope images using SIFT, random forest and hierarchical clustering. Mualla F, Scholl S, Sommerfeldt B, Maier A, Hornegger J. Transs Med Imaging. 2013; 32(12): 2274-2286.
15 Intravital leukocyte detection using the gradient inverse coefficient of variation. Dong G, ray N, Action ST. Trans med Imaging. 2005; 24(7): 910-924.
16 Breast cancer histopathology image analysis: a review. Trans. Biomed. Eng. 2011; 35(7-8):515-530.
17 Digital image processing. Gonzalez RC, Woods RE. Person Education Inc, Upper Saddle River, NJ, USA: 2008.
18 Cell segmentation: 50 years down the road. Meijering E. Sig Proc magaz. 2012; 29(5): 140-145.
19 Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques. Kothari S, Chaudry Q, Wang MD. Int Symp Biomed Imag. 2009:795-798.
20 Towards automated cellular image segmentation for RNAi genome-wide screening. Zhou X, LiuKY, Bradley P, Perrimon N, Wonsg STC. Int Conf Med Image Comput Comput Assist Intervent. 2005; 3749:885-892.
21 Unsupervised segmentation of overlapped nuclei using Bayesian classification. Jung C, Kim C, Chae SW, Oh S. Trans Biomed Eng. 2010; 57(12): 2825-2832.
22 An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. Wu Z, Leahy R. Trans Pattern Anal Mach Intell. 1993; 12(11): 1101-1113.
23 Partitioning histopathological images: an integrated framework for supervised color-texture segmentation and cell splitting. Kong H, Gurcan M, Belkacem-Boussaid K. Trans Med Imaging. 2011; 30(9):1661-1677.
24 Charisma: an integrated approach to automatic H&E-stained skeletal muscle cell segmentation using supervised learning and novel robust clump splitting. Janssens T, Antanas L, Derde S, Vanhorebeek I, den Berghe GV, Grandas FG. Med Image Anal. 2013; 17(8):1206-1219.
25 Annotated high-throughput microscopy image sets for validation. Ljosa V, Sokolnicki KL, Carpenter AE. 2012. Nature methods 9(7):637.
26 A logical calculus of ideas immanent in nervous activity. McCulloch W, Walter P. Bulletin of Mathematical Biophysics. 1943, 5(4):115-133.

27 The organization of behaviour. Hebb, D. 1949. New York; Wiley.
28 Intelligent Machinery, A Heretical Theory. Turing AM. Elsevier Science Publishers, 1992.
29 Beyond regression: new tools for prediction and analysis in the behavioural sciences. Werbos PJ. 1975.
30 A framework for designing the architectures of deep convolutional neural networks. Albelwi S, Mahmood A. Entropy 2017, 19(6), 242.
31 Receptive fields of single neurones in the cat's striate cortex. Hubel DH, Wiesel TN. J. Physiol. 1959, 148(3):574-591.
32 Recent Advances in Convolutional neural Networks. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang L, Wang G, Cai J, Chen T. 2017.
33 Efficient backpropagation. LeCun YA, Bottou L, Orr GB, Müller KR. Neural Networks: Tricks of the Trade. 2012; 9-48.
34 Rectified linear units improve restricted Boltzmann machines. Nair V, Hinton E. Proceedings of the International Conference on Machine Learning. 2010; 807-814.
35 End to end text recognition with convolutional neural networks. Wang T, Wu DJ, Coates A. Proceedings of the International Conference on Pattern Recognition. 2012, 3304-3308.
36 A theoretical analysis of feature pooling in visual recognition. Boureau Y, Ponce J, LeCun Y. Proceedings of the International Conference on Machine Learning. 2010, 111-118.
37 Gradient-based learning applied to document recognition. LeCun Y, Bottou L, Bengio Y, Haffner P. Proceedings of the IEEE. 1998.
38 Very deep convolutional networks for large-scale image recognition. Simonyan K, Zisserman A. Proceedings of the International Conference on Learning Representations. 2015.
39 Imagenet large scale visual recognition challenge. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, ma S, Huang Z, Karpathy A, Khosla A, Bernstein M. International Journal of Conflict and Violence. 2015, 115(3); 211-252.
40 Dimensionality of data with neural networks. Hinton GE, Salakhutdinov RR Science. 2006 (313); 504-507. A fast learning algorithm for deep belief nets. Hinton GE, Osindero S, Teh YW. Toronto University. 2006.
41 Unsupervised learning of invariant feature hierarchies with applications to object recognition. Ranzato MA, huang FJ, Boureau YL, LeCun Y. 2006.
42 High performance convolutional neural networks for document processing. Chellapilla K, Puri S, Simard P. Inventeurs du monde numérique. 2006.
43 Face detection using GPU-based convolutional neural networks. Nasse F, Thurau C, Fink GA. International Conference on computer analysis of images and patterns. 2009
44 Convolutional networks can learn to generate affinity graphs for image segmentation. Turage SC, Murray JF, Jain V, Toth F, Helmstaedter M, Briggman K, Denk W, Seung HS. Neural Computation. 2010; 22(2):511-538.
45 ImageNet classification with deep convolutional neural networks. Krizhevsky A, Sutskever I, Hinto GE. 2012.
46 Improving neural networks by preventing co-adaptation of feature detectors. Hinton GE, Srivastava N, Krizhevsky, Sutskever I and Salakhutdinov RR. 2012. University of Toronto.
47 Very deep convolutional networks for large-scale image recognition. Simonyan K, Zisserman A. Computer Vision and Pattern Recognition. 2014.
48 Going deeper with convolutions. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Computer Vision Foundation. 2015.
49 A review on Deep Learning Techniques Applied to Semantic Segmentation. Garcia-Garcia A. orts-Escolano S., Oprea SO, Villena-Martinez V, García-Rodriguez J. arXiv:1704.06857
50 Fully convolutional networks for semantic segmentation. Long J, Shelhamer E, Darrell T. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 3431-3440.
51 Segnet: a deep convolutional encoder-decoder architecture for image segmentation. Badrinarayanan V, Kendall A, Cipolla R. arXiv preprint arXiv:1511.00561, 2015.
52 Hierarchical Grouping to Optimeze an Objective Function. Ward JH. Journal of the American Statistical Association. 1963: 58, 236-244.
53 A threshold selection method from gray-level histograms. Otsu N. IEEE Trans Sys Man Cyber. 1979: 9(1):62-66.
54 Watershed in Digital Space: An Efficient Algorithm Based on Immersion Simulations. Vincent L, Soelle P. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1991: 13(6):583-593.