ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto fin de Carrera Grado

GITI

Segmentación de células en imágenes 3D con técnicas de machine learning

Realizado por
(ponente): Adrián López Carrillo

Dirigido por
María José Jiménez Rodríguez

Departamento
Matemática Aplicada I

Sevilla, 12 de Agosto de 2020 (v.0.1.0)

Resumen

En la actualidad existen microscopios que emplean técnicas ópticas para reconstruir imágenes 3D con una alta precisión. Gracias a esto los investigadores tienen a su alcance imágenes microscópicas de alta calidad y pueden sacar conclusiones de ellas. Estas imágenes suelen requerir un procesado digital con el objetivo de discernir los elementos importantes del resto.

En este proyecto se abordará el problema de la segmentación de células en imágenes 3D. Para ello se usarán imágenes cedidas por el Departamento de Biología Celular de la Facultad de Biología de la Universidad de Sevilla. En cada imágen hay decenas de células, todas en contacto con otras células sin espacio entre ellas.

El procesado digital de estas imágenes actualmente se hace de forma manual teniendo una duración de una a dos semanas, por lo que la automatización de este proceso conllevará un gran ahorro de tiempo.

Respecto a las técnicas usadas, este proyecto se centrará en el uso de redes neuronales para la segmentación de células. Se validará la efectividad del uso de redes neuronales y se estudiará qué tipo de red neuronal y arquitectura será mejor para esta tarea, teniendo en cuenta la exactitud de la segmentación y el coste computacional.

En el análisis de antecedenes se comprobará que la CNN (Convolutional Neural Network o Red Neuronal Convolucional) será con la que se obtienen mejores resultados en el reconocimiento de patrones en imágenes. También se verá que al diseñar una CNN con la arquitectura U-Net se obtienen buenos resultados.

Se probarán varios modelos de CNNs para la segmentación de células, esperándose el mejor resultado de la arquitectura U-Net.

Adicionalmente, se estudiará el uso de un preprocesado y postprocesado para aumentar la eficacia de la segmentación, así como un posterior mapeado a color de las células para ayudar a su visualización.

Esta herramienta estará disponible para el usuario final por medio de una imágen docker, para facilitar su distribución y que perdure el paso del tiempo.

Índice general

Índice de cuadros

Índice de figuras

CAPÍTULO 1

Definición de objetivos

Este proyecto tiene como objetivo segmentar correctamente las células mostradas en imágenes 3D mediante la aplicación de técnicas de machine learning. Posteriormente se colorearán las células de distintos colores para facilitar su visualización.

Este proceso podrá ser reproducido por un usuario ejecutando los archivos .ipynb con Jupyter Notebook, con la cual el usuario podrá entrenar el modelo con un nuevo conjunto de imágenes o inferir la segmentación de una o más imágenes.

Se intentará que esta herramienta sea accesible al mayor nº de usuarios posibles, para ello se optimizará todo lo posible el proceso de entrenamiento e inferencia y se estudiarán varios modelos con distinto grado de complejidad. Con esto se reducirá el hardware necesario.

CAPÍTULO 2

Análisis de antecedentes y aportación realizada

En este capítulo se hará una breve introducción a las redes neuronales, se justificará el uso del tipo de red neuronal CNN para tratamiento de imágenes así como el de la arquitectura CNN U-Net para la segmentación de células.

2.1– Red Neuronal Artificial

En esta sección se hará una breve introducción a las redes neuronales artificiales, compuesta por neuronas artificiales. Después se describirá qué es una neurona artificial y se definirán 3 tipos: perceptrón, sigmoide y unidad lineal rectificada (ReLU). Por último se hablará sobre el entrenamiento.

2.1.1. Introducción a Redes Neuronales Artificiales

Desde la antigüedad la humanidad ha sentido interés en la posibilidad de emular la inteligencia de forma artificial. Con los avances en neurociencia hemos sido capaces de entender cómo funcionan las neuronas, la unidad básica en el funcionamiento de los cerebros. Siendo el cerebro un ejemplo funcional de un sistema inteligente, es natural que haya interés en replicar su funcionamiento.

Un hito importante se produjo gracias al desarrollo de la teoría del aprendizaje biológico, introducida por Warren McCulloch y Walter Pitts en 1943 (McCulloch y Pitts, 1943), popularizando lo que fue llamado como *cibernética* (Goodfellow, Bengio, y Courville, 2016, p13). Gracias a esto surgió el ADALINE (elemento lineal adaptativo) (Widrow y Hoff, 2015), que es un caso concreto del algoritmo descenso por gradiente estocástico (**SGD**), algoritmo que con pequeñas modificaciones es usado en la actualidad en el proceso de aprendizaje(Goodfellow y cols., 2016, p14). Fue también gracias al estudio de McCulloch y Pitts que en 1958 Frank Rosenblatt introdujo por primera vez el **perceptrón**, un modelo general de neurona artificial (Rosenblatt, 1958), que fue perfeccionado por Minsky Y Papert en 1969 (Minsky y Papert, 1969).

El perceptrón es un modelo lineal que, dado un conjunto de elementos con dos categorías distintas como entrada, puede clasificar cada elemento en una de esas dos categorías. Un ejemplo sencillo son las puertas lógicas, siendo la entrada un conjunto de dos elementos con las categorías 0 o 1 y la salida sería un 0 o un 1.

Minsky y Papert encontraron un problema en los modelos lineales y lo demostraron con la función XOR, siendo imposible para un modelo lineal compuesto de una neurona artificial aprender esta función. Esto causó un declive en el interés sobre este campo.

En la década de 1980 resurgió el interés gracias en parte al conexionismo, cuya idea central es que un gran número de unidades de computación simples pueden tener un comportamiento inteligente al estar conectadas entre sí (Goodfellow y cols., 2016, p16). Durante esta etapa se hicieron importantes contribuciones como la representación distribuida (G. E. Hinton, 1986), donde se habla sobre representar las entradas de un sistema en base a sus características, reconocidas por patrones de actividad en redes neuronales. Otra gran contribución fue la populariación del algoritmo de propagación hacia atrás, **backpropagation** (Rumelhart, Hinton, y Williams, 1986) para entrenar redes neuronales artificales y actualizar sus pesos, siendo este algoritmo el más usado en la actualidad.

En este punto de la historia los algoritmos más importantes involucrados en las redes neuronales artificiales usadas en la actualidad habían sido descubiertos, pero no se estaban obteniendo resultados tan buenos como los esperados. Desde un principio lo que se había estado buscando era replicar de forma artificial el funcionamiento del cerebro, siendo el cerebro un sistema de computación genérico, capaz de aprender todo tipo de conocimiento distinto sin la necesidad de cambiar su arquitectura o su método para aprender. Era imposible conocer el algoritmo de aprendizaje usado en el cerebro ya que para ello haría falta monitorizar una gran cantidad de neuronas con gran precisión, lo cual es imposible incluso en la actualidad.

En 2006 se produjo un hito importante que comenzó la etapa del **aprendizaje profundo**, cuando Geoffrey Hinton demostró que era posible entrenar de forma eficiente una red neuronal profunda con un gran número de capas ocultas (G. Hinton, Osindero, y Teh, 2006).

2.1.2. Neurona artifical

En la figura 2.1 se puede ver una neurona artifical genérica con la que pueden ser descritos los distintos tipos que se verán en esta sección.

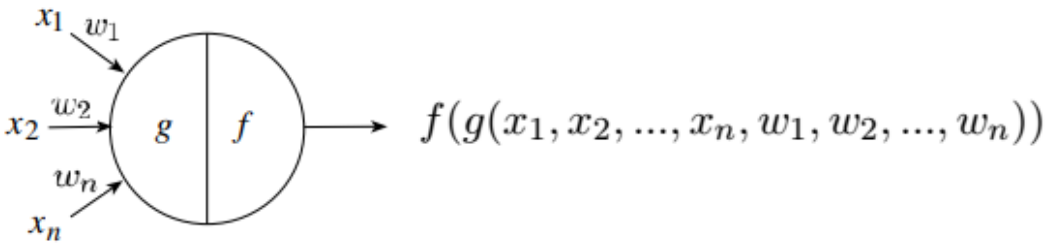

Figura 2.1: Neurona artificial genérica

Siendo:

- $(x_1, x_2, ..., x_n)$ el vector de entrada.
- $(w_1, w_2, ..., w_n)$ el vector de pesos.
- g la función de integración, encargada de reducir el vector de entrada a un único valor.
- f la función de activación, encargada de producir la salida de este elemento.

Se puede simplificar la representación al asumir que siempre se usará $\sum x_i w_i$ como función de integración. Además toda neurona artificial tendrá una entrada y un peso por defecto, independientemente del vector de entrada, esto hará referencia al *bias*. Será común ver una representación como 2.2 en la que f indicará la función de activación.

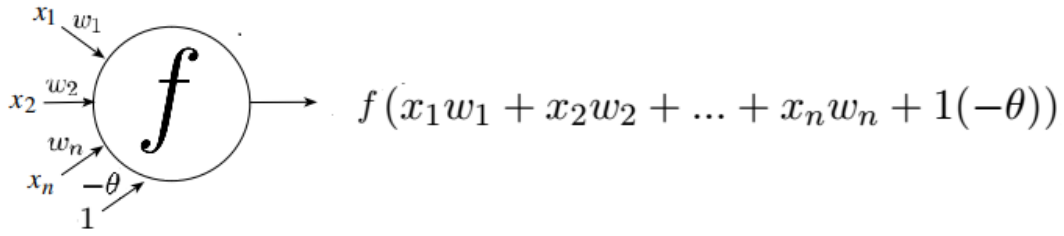

Figura 2.2: Representación de una Neurona Artificial con $\sum x_i w_i$ como función de integración

- Al vector de entradas se le añade un elemento de valor constante 1, siendo ahora de tamaño $n + 1$.
- Al vector de pesos se le añade un elemento de valor inicial $-\theta$, siendo ahora de tamaño $n + 1$. A este valor se le llamará *bias*.
- f indicará la función de activación de la neurona artificial.

Sigmoide

La función sigmoide como función de activación es una función no lineal usada principalmente en redes neuronales prealimentadas (feedforward neurals networks), que son las que usaremos en este proyecto. Es una función real, acotada y diferenciable (a diferencia de la usada en el perceptrón). Su definición es la siguiente relación (Nwankpa, Ijomah, Gachagan, y Marshall, 2018):

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

Unidad Lineal Rectificada (ReLU)

La unidad lineal rectificada (ReLU) fue propuesta como función de activación en 2010 por Nair y Hinton (Nair y Hinton, 2010) y desde entonces ha sido la más usada en aplicaciones de aprendizaje profundo (deep learning, DL). Si se compara con la función de activación Sigmoide, ofrece un mejor rendimiento y es más generalista (Nwankpa y cols., 2018).

$$f(x) = max(0, x) = \begin{cases} x_i & \text{si } x_i \geq 0 \\ 0 & \text{si } x_i < 0 \end{cases} \tag{2.2}$$

2.2– Red Neuronal Convolucional

2.3– Arquitecturas usadas en segmentación

2.4– Software desarrollado

2.5– Aportación Realizada

CAPÍTULO 3

Análisis de requisitos, diseño e implementación

3.1– Datos de entrada

Los datos de entrada son 20 ficheros .mat, cada uno contiene la siguiente :

- **imageSequence**: Imagen inicial con la forma $(57, 1024, 1024)$
- **imageSequenceInterpolated**: Imagen incial interpolada, ahora con la forma $(244, 1024, 1024)$.
- **labelledImage3D**: Resultado de etiquetar correctamente la imagen inicial interpolada $(244, 1024, 1024)$.
- **centroidOfRois**: Coordenadas del centroide de cada célula.
- **usedZScale**: Valor usado en la interpolación.

A continuación se analizarán las imágenes **imageSequenceInterpolated** y **labelledImage3D**.

imageSequenceInterpolated

El tipo usado para almacenar el valor de cada vóxel es *f8* que, según la documentación de numpy es el equivalente a un número de 64-bit float.

Se han comprobado los valores máximo y mínimo de las 20 imágenes y estos son los resultados:

35.0 4095.0	32.0 4095.0	33.0 4095.0	0.0 65535.0	31.0 4069.0	24.0 4095.0	19.0 4095.0	0.0 65535.0	0.0 65535.0	24.0 4095.0
22.0 4095.0	22.0 4095.0	23.0 4095.0	30.0 4095.0	31.0 4095.0	30.0 4095.0	25.0 4095.0	30.0 4095.0	23.0 4095.0	19.0 4095.0

Cuadro 3.1: Valores mínimo y máximo de los píxeles de las imágenes de entrada.

Se ha analizado el histograma de una imagen para entender mejor la distribución de valores. Podría ser que se pudiera simplificar el nº de valores distintos que tiene la imagen para aumentar el rendimiento. La imagen analizada tiene 32 de valor mínimo y 4095 de valor máximo.

En la figura 3.1 se puede ver un histograma en el que en el eje x representa el valor de un píxel y el eje y representa el nº de píxeles con ese valor. En esta figura se puede comprobar que los valores están concentrados en el rango $[0, 500]$, pero da la impresión de que no hay ningún elemento en el resto de valores. Para obtener una mejor visualización del resto en la figura 3.2 se puede ver otro histograma con el eje y limitado a 4000, donde se aprecia los distintos valores que un píxel puede tomar.

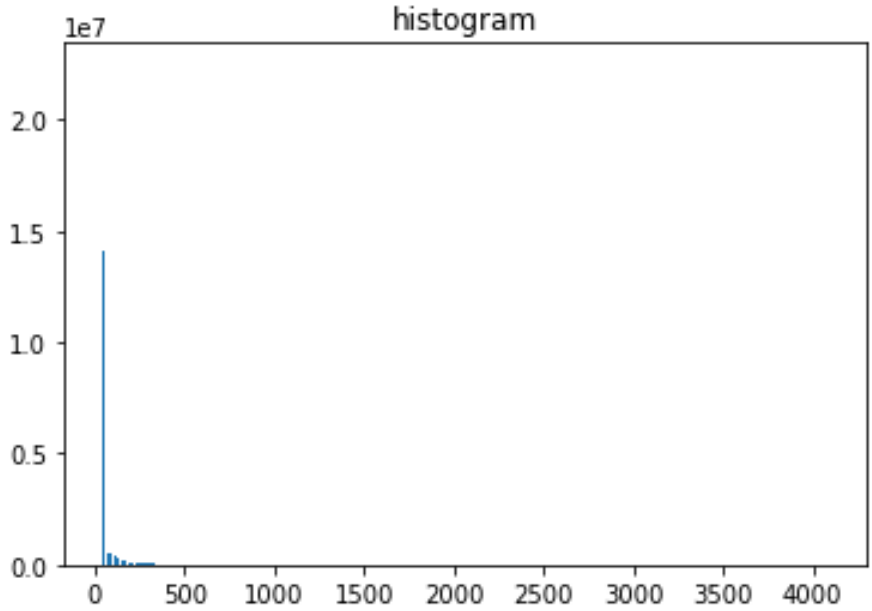

Figura 3.1: Histograma de todos los valores

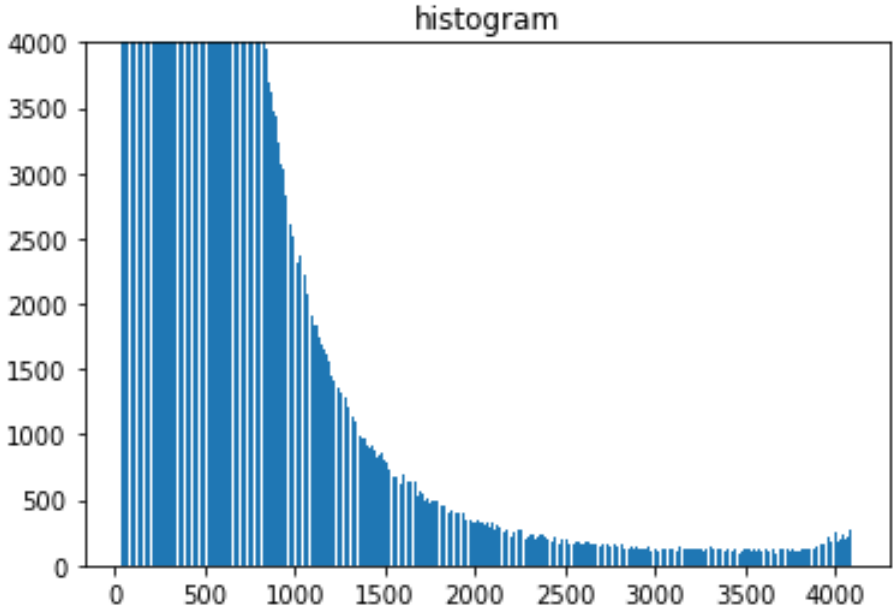

Figura 3.2: Histograma de todos los valores con el eje y limitado a 4000

3.2– Diseño e implementación

Se tienen 20 imágenes de células
Cada imagen es una matriz .
De una imagen se ha obtenido un valor mínimo de 33 y máximo de 4095.

Referencias

Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning*. MIT Press. Descargado de https://www.deeplearningbook.org/ (http://www.deeplearningbook.org)

Hinton, G., Osindero, S., y Teh, Y.-W. (2006, 08). A fast learning algorithm for deep belief nets. *Neural computation*, *18*, 1527-54. doi: 10.1162/neco.2006.18.7.1527

Hinton, G. E. (1986). Learning distributed representations of concepts. En *Proceedings of the eighth annual conference of the cognitive science society.*

McCulloch, W. S., y Pitts, W. (1943, 01 de Dec). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133. Descargado de https://doi.org/10.1007/BF02478259 doi: 10.1007/BF02478259

Minsky, M., y Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. Cambridge, MA, USA: MIT Press.

Nair, V., y Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. En *Proceedings of the 27th international conference on international conference on machine learning* (p. 807–814). Madison, WI, USA: Omnipress.

Nwankpa, C., Ijomah, W., Gachagan, A., y Marshall, S. (2018). *Activation functions: Comparison of trends in practice and research for deep learning.*

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65–386.

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986, 01 de Oct). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536. Descargado de https://doi.org/10.1038/323533a0 doi: 10.1038/323533a0

Widrow, B., y Hoff, T. (2015). Adaptive linear neuron..