Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil en Informática

Desarrollo de herramientas de imagenología para cuantificación de secuencias de imágenes biomédicas 3D

Proyecto para optar al título de
Ingeniero Civil en Informática

PROFESOR PATROCINANTE:
PABLO HUIJSE HEISE
DOCTOR EN INGENIERÍA ELÉCTRICA

PROFESOR CO-PATROCINANTE:
CLAUDIO ARAYA GARCIA
DOCTOR EN NEUROCIENCIAS

PROFESOR INFORMANTE:
NOMBRE DEL INFORMANTE
TÍTULOS Y GRADOS DEL INFORMANTE

LUIS FELIPE IGNACIO GUZMÁN BASTIDAS

VALDIVIA - CHILE
2020

AGRADECIMIENTOS

En primera instancia quisiera agradecer por el logro de ésta meta a toda mi familia, que con su apoyo, compresión y motivación hacia mi persona, me ayudaron a lograr finalizar éste viaje universitario, el cual es una de las etapas más importantes de mi vida. En especial quiero agradecer a mi madre, ya que sin ella nada de esto se podría haber logrado. Soy consciente que más de la mitad de este logro es de ella, ya que ella se esforzó tanto o más que yo, por lo que estaré toda la vida agradecido.

Desde el momento que ingrese a la Universidad se han conformado muchas amistades que perdurarán por toda la vida. A los cuales estoy profundamente agradecido por estar junto a mí en tiempos complicados en donde quería renunciar, y gracias a ellos y con palabras de aliento, pude salir adelante.

Deseo agradecer a la Universidad, en especial a todos los docentes tanto de la carrera como del bachillerato. También me gustaría agradecer a la selección de rugby, tanto a los profesores como a mis compañeros, ya que con ellos formé una familia, ya que semana tras semana nos esforzamos para ir a entrenar, independiente de los estudios, para de esta forma representar de la mejor manera posible a la universidad.

Además quiero agradecer a mi profesor Patrocinante, Dr. Pablo Huijse, quien me dio la posibilidad de realizar el proyecto de título y se dio el tiempo semana tras semana de reunirse conmigo para conversar y corregir el proyecto, esto independiente de la pandemia. También agradezco al Dr. Claudio Araya, mi Co-Patrocinante, por guiarme en un área la cual era totalmente desconocida para mi en el transcurso del trabajo realizado.

Finalmente agradezco a todas las personas que en algún momento estuvieron cerca de mí y me brindaron de alguna otra forma un grano de arena para lograr esta meta.

ÍNDICE

ÍNDICE DE TABLAS

TABLA PÁGINA

ÍNDICE DE FIGURAS

FIGURA PÁGINA

RESUMEN

La morfogénesis animal es un proceso altamente coordinado en el tiempo y el espacio mediante el cual un grupo de células es capaz de generar órganos tridimensionales con un tamaño y forma definido. En años recientes, el uso de la microscopía confocal ha permitido registrar en vivo este proceso a nivel celular. Estos estudios también demuestran que la conducta celular que guía estos procesos son rápidos y de tipo oscilatorios, presentando una serie de desafíos a la comunidad científica a fin de ser debidamente cuantificados. Normalmente, la caracterización de estos procesos descansa en la segmentación manual. El investigador inspecciona cuidadosamente estas secuencias de imágenes capturas con microscopio confocal y segmenta manualmente las células de cada imagen en busca de patrones. Sin embargo, este proceso es sumamente lento debido al gran volumen de información a segmentar y el alto nivel de ruido que normalmente presentan las imágenes.

En este proyecto se propone desarrollar e implementar las primeras etapas de una herramienta computacional para procesar estos tensores de imágenes de forma automática. La herramienta ha de cuantificar los procesos de deformación celular (área) a gran resolución temporal (segundos) responsables de la organogénesis a nivel celular, entregando información concreta y representativa al investigador. Estas primeras etapas consisten en la: (1) segmentación de las células, (2) seguimiento de las células a través de la secuencia de imágenes y (3) estimación de las oscilaciones en el área de cada célula a través del tiempo. Para segmentar las imágenes se consideran modelos basados en redes neuronales artificiales profundas, que corresponden al estado del arte en el procesamiento automático de imágenes. Debido a la escasez de datos etiquetados se considera utilizar un modelo de red neuronal basado en la arquitectura U-net cuyos parámetros fueron ajustados previamente con una base de datos abierta de características similares a las imágenes utilizadas en este proyecto. Para obtener una serie del tiempo del área de una célula en particular a lo largo de la secuencia se utiliza un método inspirado en los vecinos mas cercanos. Finalmente, se aplica la función de autocorrelación sobre la serie de tiempo obtenida en el paso anterior para estimar el período de oscilación del área de la célula.

Se utilizan dos datasets con imágenes reales provenientes de dos procesos de organogénesis a fin de validar los procesos antes mencionados. En primer lugar se compara un método referencial basado en técnicas clásicas (umbrales dinámicos) con el método propuesto basado en el modelo U-net para la tarea de segmentación semántica de las células. El modelo U-net produce máscaras menos ruidosas y alcanza un menor error de segmentación según la métrica de entropía cruzada. Para validar el procedimiento de extracción de series de tiempo y estimación de período se comparan los resultados obtenidos usando el sistema propuesto con los reportados en la literatura. Para el primer dataset se obtiene un período de oscilación promedio de 202.86 ± 31.04 segundos, lo cual

es consistente con la literatura. Para el segundo dataset se obtiene un período promedio de 4057.31 ± 1450.14 segundos, lo cual no concuerda con lo reportado previamente. Creemos que esto puede explicarse por la menor resolución temporal de este dataset. Finalmente se proporcionan lineamientos para mejorar y seguir desarrollando este prototipo tal que alcance el estado de producto.

ABSTRACT

Animal morphogenesis is a highly-coordinated temporal and spatial process by which a group of cells is able to generate three-dimensional organs with a defined size and shape. In recent years, the use of confocal microscopy has allowed scientists to record this process in real time at the cellular level. These studies also demonstrate that the cellular behavior that guides these processes is rapid and oscillatory, presenting a number of challenges to the scientific community in order to be properly quantified. Typically, the characterization of these processes relies on manual segmentation. The researcher carefully inspects the sequences of images captured using confocal microscopy and manually segments the cells in each image in search of patterns. However, this process is extremely slow due to the large volume of information to be segmented and the high level of noise in the images.

In this project we propose to develop and implement the first stages of a computational tool to process these image tensors automatically. The tool has to quantify the cell deformation processes (area) at high temporal resolution (seconds) responsible for organogenesis at the cellular level, and deliver concrete and representative information to the researcher. These first stages consist of: (1) segmentation of the cells, (2) tracking of the cells through the image sequence and (3) estimation of the oscillations of the cell's area over time. Models based on deep artificial neural networks, which correspond to the state of the art in automatic image processing, are considered for segmenting the images. Due to the scarcity of labeled data a pre-trained neural network model based on U-net architecture is considered. The parameters of this model were previously adjusted with an open database of similar characteristics to the images used in this project. To obtain a time series of the area of a particular cell, a method inspired by nearest neighbors is used. Finally, the autocorrelation function is applied on the recovered time series in order to estimate the oscillation period of the cell area.

Two datasets with real images from two organogenesis processes are used to validate the above mentioned processes. First, a referential method based on classical techniques (dynamic thresholding) is compared with the proposed method based on the U-net model for the semantic cell segmentation task. The U-net model produces less noisy masks and achieves a lower segmentation error according to the cross-entropy metric. To validate the time series extraction and period estimation procedure, the results obtained using the proposed system are compared with those reported in the literature. For the first dataset an average oscillation period of 202.86 ± 31.04 seconds is obtained, which is consistent with the literature. For the second dataset an average period of 4057.31 ± 1450.14 seconds is obtained, which is not in agreement with what was previously reported. We believe that this can be explained by the lower temporal resolution of this dataset. Finally, guidelines are provided to improve and further develop this prototype to reach product status.

1. INTRODUCCIÓN

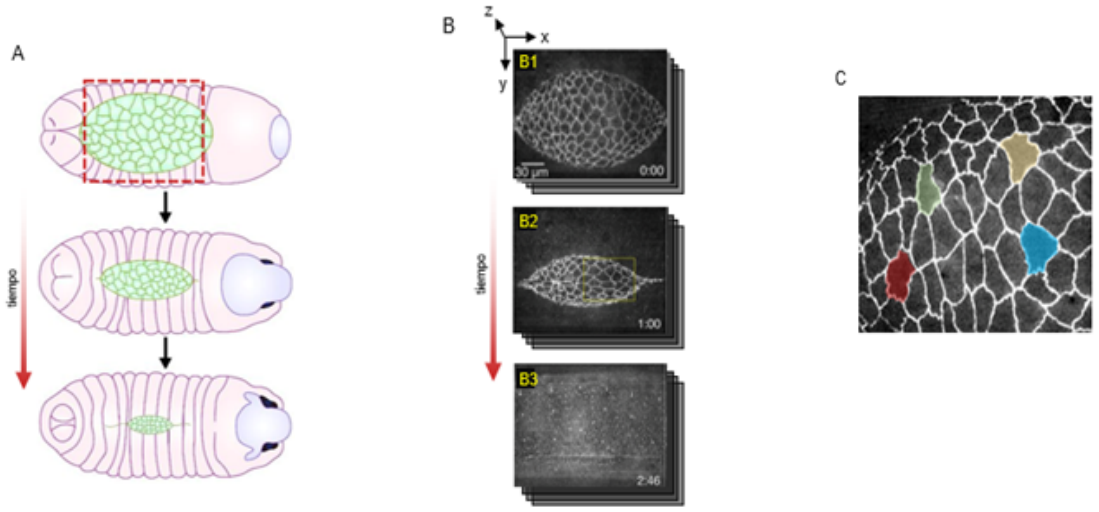
1.1. Motivación y oportunidad

La organogénesis es el proceso en el cual las células se coordinan para formar órganos dentro del embrión (Araya, Ward et al. 2015). Cómo los tejidos y órganos animales adquieren su forma y su tamaño definido durante la embriogénesis es una pregunta central en la Biología moderna. A su vez, entender cómo ocurre este proceso morfogenético (origen de la forma) es central en los estudios biomédicos a fin de comprender una multitud de defectos congénitos como el labio leporino y defectos en la formación del tubo neural, los cuales surgen como consecuencia de un morfogénesis embrionaria deficiente (Jiang et al. 2006). Dado que los tejidos animales están compuestos por muchas células, la comprensión de la morfogénesis tisular requiere entender las conductas celulares en forma individual y colectiva. El futuro de la biomedicina está en la capacidad de generar tejidos y órganos artificiales o compuestos de biomateriales (Sasai, 2013).

En años recientes, el desarrollo de la microscopía confocal ha permitido visualizar y registrar en forma digital y en vivo la dinámica celular colectiva responsable de la organogénesis animal a gran escala temporal y espacial. El uso de esta metodología de registro ha permitido reconsiderar una serie de paradigmas respecto a la forma por el cual las células llevan a cabo el proceso de morfogénesis. Tal vez, el paradigma más notable de estos avances ha sido el hecho de que el cambio en la forma celular en procesos de morfogénesis, dista mucho de ser un evento continuo, sino más bien es tipo pulsátil con ciclos rápidos de expansión y contracción (Solon et al. 2009), revelando aspectos biomecánicos que operan en la organogénesis, tales como tensiones, fuerzas, elasticidad y viscosidad.

Así, una etapa clave en el proceso de investigación científica de la organogénesis animal es poder cuantificar a gran precisión la dinámica celular a fin de poder desarrollar futuros modelamientos matemáticos y estadísticos que permitan captar y racionalizar la generación de tejidos animales. Un aspecto dinámico en particular de gran importancia es la denominada oscilación celular, que quiere decir cambios de área o perímetro de la célula en el tiempo, lo cual es muy importante de estudiar ya que aún se desconoce cómo se establece esta conducta celular durante el desarrollo embrionario y la influencia potencial que tendría en la organización mecánica de los tejidos en desarrollo (Araya, Ward et al. 2015). Hasta ahora, estos análisis se realizan en forma manual, donde el investigador debe realizar una segmentación manual sobre el objeto (célula) y en la secuencia de imágenes correspondientes.

Sin embargo, este proceso en muchos casos es muy lento, laborioso y muchas veces



(a) (b) (c)

Figura 1. A) Modelo de organogénesis en vivo en Drosophila melanogaster (cierre dorsal). Recuadro en rojo indica área de registro. B) Registro y adquisición de imágenes 3D en el tiempo y el espacio. X,Z, e Y indican ejes de registro. En blanco, se indican los límites (perímetros de las células). C) Reconstrucción de formas.

incompleto debido a la falta de herramientas de imagenología cuantitativas adaptadas a la necesidad del investigador. Se detecta entonces la oportunidad de desarrollar un algoritmo computacional para realizar segmentación, seguimiento y estimación de parámetros, como por ejemplo el perímetro y área celular, en una secuencia de imágenes biológicas de forma automática.

1.2. Propuesta e innovación

En este proyecto se busca desarrollar una herramienta para procesar automáticamente secuencias de imágenes que caracterizan un proceso de organogénesis y que fueron capturadas con un microscopio electrónico confocal. En la **??**, se muestra la metodología de adquisición de las imágenes de microscopía confocal durante un proceso de organogénesis en vivo. En ella se puede observar, el registro de los dataset en el tiempo y el espacio. Bajo esta aproximación, es posible registrar y estudiar la conducta celular (ejemplo: cambios temporales del área celular) mediante la tinción de sus membranas (líneas clara en series B-B3),a fin de reconstruir sus formas (C).

La herramienta propuesta recibe una secuencia de imágenes y retorna automáticamente una serie de tiempo con la evolución del área de cada célula en particular a lo largo de

la secuencia. Luego, con estas series es posible cuantificar la amplitud y frecuencia de sus oscilaciones. Mediante la utilización de herramientas automáticas de procesamiento de imágenes se libera al experto de analizar las imágenes de forma manual, se incrementa el volumen de imágenes analizadas y por ende se mejora la calidad del análisis estadístico de alto nivel que se realiza sobre los resultados de la segmentación. De esta forma se contribuye desde la informática a las ciencias biológicas, permitiendo avanzar en el entendimiento del proceso de organogénesis a nivel científico, y siendo una base de desarrollo interdisciplinar que puedan seguir siendo desarrollada en el futuro. Además la automatización de este proceso es clave en diversos proyectos, siendo posible escalarlo a toda la comunidad de investigadores.

Como parte de la herramienta se utilizan algoritmos para segmentación semántica basados en redes neuronales profundas (Goodfellow et al. 2016) que distinguen las membranas celular del resto de las estructuras. Luego se utiliza un método basado en el criterio de los vecinos más cercanos para identificar únicamente a cada célula a lo largo de la secuencia. La herramienta desarrollada queda disponible a la comunidad bajo una licencia de código abierto y libre de costo. Esto diferencia nuestra propuesta de soluciones existentes que en su mayoría son privadas y/o no abiertas a explorar los códigos por el cual fueron escritas. El principal desafío en este caso particular es el bajo número de imágenes etiquetadas para poder entrenar la red neuronal. Para solventar este situación se utilizan bases de datos abiertas de imágenes celulares etiquetadas que son similares a las imágenes presentadas.

Dentro de este documento se dan a conocer las principales herramientas utilizadas por los investigadores hoy en día, para luego dar paso a las redes neuronales artificiales y el fundamento de su utilización en el procesamiento de imágenes y en particular en la segmentación celular. Luego se presenta el diseño metodológico y el detalle de las etapas conducente a la implementación de la herramienta de cuantificación, sus resultados así como los mejoras que podrían implementarse a futuro.

1.3. Objetivos

A continuación se describen los distintos objetivos formulados para el trabajo del proyecto de título.

1.3.1. General

Automatizar y disminuir el tiempo de segmentación y análisis de la deformación celular en imágenes de microsocopía confocal a alta resolución mediante una herramienta basada en redes convolucionales.

1.3.2. Específicos

1. Investigar las alternativas existentes para realizar segmentación celular, identificando sus fortalezas y debilidades.
2. Revisar el estado del arte en segmentación y tracking en imágenes con redes convolucionales.
3. Entrenar un modelo con redes convolucionales para hacer segmentación automática y estimación de área/perímetro de células a partir de una secuencia temporal de imágenes provenientes de microscopia confocal.
4. Desarrollar un algoritmo para realizar tracking automatizado de área celular en el tiempo. En base a esto cuantificar la oscilación celular de un proceso de organogénesis en vivo.
5. Construir una plataforma interactiva donde el investigador pueda utilizar los modelos y algoritmos de manera óptima y eficiente.

2. MARCO TEÓRICO

2.1. Herramientas estándar y sus limitaciones

Una de las herramientas de más amplio uso para el análisis de imágenes biológicas es ImageJ[1] (Abràmoff et al. 2004), el cual es un programa para el procesamiento de imágenes digitales con foco en el ámbito científico. El programa es multiplataforma (existen versiones para Windows, Linux y Mac OS) y la mayoría de sus módulos son de código abierto. El programa permite realizar operaciones sobre imágenes que van desde lo trivial y sencillo, como por ejemplo ajustar el contraste y transformar una imagen a escala de grises hasta lo más complejo y sofisticado, como la aplicación de distintos filtros de procesamiento prediseñados y ampliamente usados del procesamiento clásico de imágenes (Patin, 2003). Algunos de los filtros más utilizados de esta herramienta, son el de realce de contraste, filtro de suavizado como lo pueden ser el de media, mediana o Gaussiano, entre otros. Esta herramienta es utilizada por ser de fácil acceso y contar con un gran número de filtros muy útiles para el procesamiento digital de imágenes.

ImageJ tiene numerosas aplicaciones ya sea tanto a nivel profesional como a nivel de investigación, gracias a que ofrece una gran potencialidad para el procesamiento de imágenes, siendo una herramienta de libre distribución. Además cuenta con una licencia de dominio publico y fue desarrollada en lenguaje de programación Java. La limitación que conlleva el utilizar este software, es que la gran mayoría los procesos se tienen que hacer de manera manual, por lo que no es eficiente ni rápido. Sin embargo es posible, a través de su sistema de *plug-ins*, incorporar funcionalidades automáticas y más avanzadas tales como las que revisamos a continuación.

Existe un precedente de un proyecto de título de ingeniero civil en informática de la Universidad Austral de Chile donde se desarrolló un *plug-in* de ImageJ para la segmentación de núcleos celulares en tejidos profundos a partir de imágenes captadas con microscopio confocal de embriones de peces cebra (Subiabre, 2013). El objetivo del proyecto fue identificar y cuantificar los núcleos celulares presentes en las imágenes biológicas curvas y de alta densidad celular (Araya, Tawk et al. 2014) mediante un algoritmo de segmentación de carácter adaptativo. Un ejemplo de las imágenes analizadas se puede apreciar en la Figura 2.

Para desarrollar dicho *plug-in* se utilizó el lenguaje de programación Java con el ambiente de desarrollo integrado Eclipse. El proceso de segmentación de dichas imágenes consta de los siguientes pasos:

1. Preprocesamiento: Se considera un suavizado de los píxeles de la imagen mediante

[1] https://imagej.nih.gov/ij/

la convolución con un filtro gaussiano. Cada pixel resultante es un promedio entre su valor y el de sus vecinos más cercanos ponderados según una función gaussiana bidimensional. Se usa la rutina *smooth* incluida en ImageJ.

2. Segmentación por umbral: Es uno de los métodos más simples de segmentación. El método divide los píxeles de la imagen según su valor de intensidad o nivel de gris en dos conjuntos, los pertenecientes a objetos de interés y los pertenecientes al fondo, dando como resultado una imagen binaria. Típicamente, el valor umbral es único para todos los píxeles de la imagen y se escoge en base a la distribución de intensidades o histograma de los valores de intensidad. Esto se conoce como segmentación con umbral fijo. Dentro del proyecto de tesis se propone un umbral de segmentación dinámico que luego se compara con el umbral fijo basado en histograma. La umbralización dinámica o adaptativa no se basa en un único umbral (nivel de gris) para binarizar la imagen, sino que utiliza un umbral particular para cada píxel de la imagen cuyo valor depende de la vecindad de dicho pixel, permitiendo binarizar correctamente imágenes cuya condiciones de iluminación no se mantienen constantes (Subiabre, 2013). Aun así, es necesario proporcionar ciertos parámetros al operador para lograr un resultado aceptable, y estos parámetros están fundamentalmente relacionados con el tamaño del objeto relativo al fondo.

3. Reducción de ruido impulsivo: Se aplica un filtro mediana usando la rutina *despeckle* incluida en ImageJ, bajo el submenú *noise*. Este filtro retiene la mediana de los píxeles en cada vecindad de 3x3 presente en la imagen. El objetivo del filtro mediana es eliminar el ruido de tipo sal y pimienta, el cual se caracteriza por presentar valores extremos en la imagen.

Luego de estos pasos se realiza un proceso de cuantificación manual por parte del experto para la validación. Si bien esta metodología sienta un precedente importante se reconoce que no es directamente aplicable al presente problema puesto que:

1. Se enfoca en la detección de núcleos en lugar de membranas celulares (útiles para estudiar cambios en la forma celular).

2. Las imágenes son de naturaleza distinta y en particular presentan una mejor definición tal como se aprecia en la Figura 2.

3. El proceso de cuantificación celular final y selección de corte se realizan manualmente.

4. Las imágenes corresponden a un cuadro en particular en lugar de una secuencia por lo que esta orientado en obtener información estacionaria (cuantificación de objetos ó núcleos) y no dinámica (deformidad celular en el tiempo).

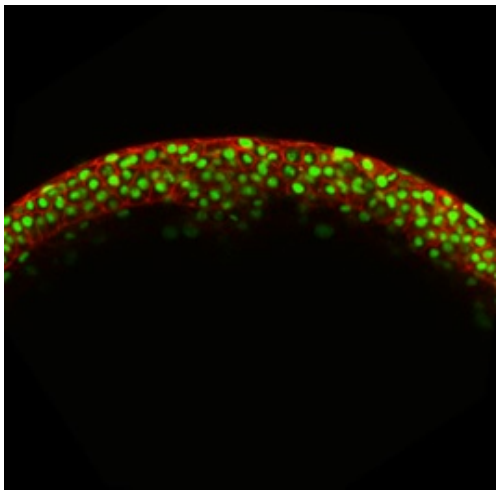


Figura 2. Imagen de embrión pez cebra exhibiendo núcleos celulares (verde) y paredes celulares (rojo).

Cabe destacar que los filtros utilizados por Subiabre, 2013 no se adaptan en base a datos como los que se proponen en este trabajo.

2.2. Redes neuronales profundas para procesamiento de imágenes

En estos últimos años se ha visto un importante avance en lo que respecta al campo de la visión computacional, es decir, modelos para clasificación, localización y segmentación de entidades en imágenes o video. El estado del arte está representado por las redes neuronales convolucionales (Convolutional Neural Networks, CNN) (Goodfellow et al. 2016; LeCun et al. 1989). Estos modelos son aproximadores de funciones paramétricos que aprenden filtros bidimensionales directamente desde las imágenes. Una estructura clásica de una CNN se puede apreciar en la Figura 3. La Figura muestra los tres tipos de capas que aparecen típicamente en este tipo de arquitecturas, las cuales se describen en detalle a continuación:

1. Capas convolucionales: En esta capa los parámetros o neuronas se organizan como filtros, típicamente bidimensionales, que se convolucionan con la entrada para producir un mapa de características. Seguido del mapa de características suele aplicarse una función de activación no lineal tales como $\tanh(x)$ o $\text{ReLU}(x) = \text{mín}(0, x)$. La operación de convolución en sí corresponde a multiplicar punto a punto el filtro con un segmento o parche particular de la imagen para luego sumar el resultado, luego el filtro se desplaza una cierta cantidad de píxeles y repitiendo el proceso. El proceso termina una vez que se ha recorrido toda la entrada. Los parámetros entrenables de esta capa los valores de los filtros y los hiperparámetros

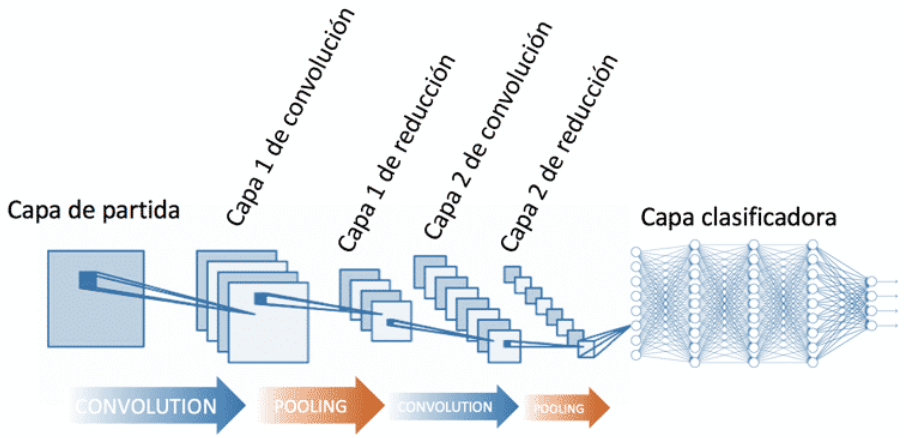


Figura 3. Esquema de la estructura general de una red neuronal convolucional

que el usuario debe preseleccionar son (1) la cantidad de filtros de convolución, (2) el tamaño de los filtros de convolución y (3) el paso o *stride* de los filtros.

2. Capas de pooling: Se utilizan para reducir la dimensionalidad de los mapas de características resultantes de las capas convolucionales. La capa de pooling submuestrea la imagen usando un operador. Los operadores más típicos son el máximo y el promedio. Por ejemplo, en max-pooling, se recorre la imagen rescatando solo el valor máximo de cada vecindad de píxeles. Esto no sólo hace más eficiente a las capas subsecuentes sino que además favorece a que los resultados del modelo sean invariantes a la posición de las entidades en la imagen. Notar que esta capa no tiene parámetros entrenables. Los hiperparámetros que el usuario debe preseleccionar son el tamaño, el paso y el tipo del operador.

3. Capa completamente conectada: En esta capa, como su nombre lo dice cada neurona esta conectada a todas las neuronas de la capa anterior y cada conexión tiene su peso propio. Este es un patrón de conexión de propósito totalmente general y no hace suposiciones sobre las características de los datos. La última capa de la red neuronal es la que se encarga de clasificar y tendrá tantas neuronas como el número de clases a predecir.

La arquitectura general de una CNN consiste de una primera parte compuesta por una concatenación de bloques de convolución, activación y pooling. La cantidad de bloques, es decir la profundidad del modelo, es un decisión de diseño que el usuario debe tomar en base al problema particular que se busca resolver. Esta parte de la red neuronal se encarga de extraer características relevantes a partir de las imágenes. La segunda parte, compuesta por capas completamente conectadas, se encargan de realizar la clasificación

de las entidades en base a las características extraídas en la parte anterior.

Las redes neuronales convolucionales se entrenan mediante un proceso de optimización supervisado donde se minimiza una función de costo que representa el error entre la salida de la red neuronal y una etiqueta asignada por un experto que indica clase o categoría (Goodfellow et al. 2016). Típicamente se utilizan optimizadores de primer basados en gradiente descendente estocástico para realizar la optimización. Las librerías actuales para entrenar redes neuronales (Paszke et al. 2019) realizan el cómputo de gradientes y la actualización de los parámetros empleando Unidades de Procesamiento Gráfico (Graphical Processing Units, GPU) acelerando considerablemente los cálculos y permitiendo entrenamientos con grandes conjuntos de datos.

Como en otros métodos supervisados es fundamental que el conjunto de datos utilizado para entrenar sea representativo del problema que se busca resolver. La eficacia del modelo depende tanto de la arquitectura como del conjunto de datos empleado para su entrenamiento. En estos modelos es común que se de un problema de sobreajuste el cual se produce cuando el modelo se ajusta excesivamente bien a los datos de entrenamiento perdiendo capacidad de generalizar a datos fuera del mismo. Hay que recordar que el objetivo de los modelos de aprendizaje automático es el de obtener patrones de los datos de entrenamiento disponibles de cara a predecir o inferir correctamente sobre datos nuevos. Es decir, el concepto clave es el de entrenar y obtener patrones generales que sean extrapolables a nuevos datos (Bishop, 2006). Algo similar ocurre en el aprendizaje de los seres humanos, el sobreajuste se produciría cuando aprendemos las cosas de memoria, sin entender el concepto. Para evitar el sobreajuste de los modelos se utilizan estrategias de validación cruzada (Bishop, 2006). El criterio más utilizado es la validación simple el cual consiste en repartir aleatoriamente las observaciones disponibles en dos grupos. El primer grupo se utiliza para ajustar los parámetros del modelo mientras que el segundo se utiliza para validar que el modelo no se haya sobreajustado.

Una alternativa para aprender modelos con bases de datos reducidas capaces de generalizar adecuadamente sin sobreajuste consiste en usar metodologías de transferencia de aprendizaje o *transfer learning* (Yosinski et al. 2014). Esto corresponde en tomar un modelo que fue previamente entrenado en un dataset de gran volumen, denominado el dominio fuente, y transferirlo a un dataset de interés posiblemente más pequeño, denominado el dominio objetivo. El proceso de transferencia en el caso de las redes neuronales consiste en entrenar iterativamente el modelo con los datos del dominio objetivo pero partiendo de la mejor solución obtenida en el dominio fuente, en lugar de una solución inicial aleatoria como es lo usual. Este entrenamiento o ajuste-fino puede ocurrir en todas las capas del modelo o en un subconjunto de capas, en particular si el dataset objetivo es pequeño se entrenan sólo las capas finales. Esta metodología se sustenta en la observación formalizada por Zeiler y Fergus, 2014 de que las capas iniciales de la

red aprenden filtros que extraen características comunes a todas las clases del problema, como serían por ejemplo bordes en distintas orientaciones y gradientes de color. En cambio las capas finales son las que tienen más relación a las clases y por ende son las más dependientes al dataset utilizado. En ejemplo clásico de esta metodología es el entrenamiento de modelos específicos de visión computacional basados en modelos pre-entrenados con ImageNet[2], un dataset de imágenes naturales que contiene millones de ejemplos y cientos de clases etiquetadas. En importante detalle a considerar es que la efectividad de *transfer learning* es directamente proporcional a la similitud que exista entre los ejemplos de los dominios fuente y objetivo.

2.3. Redes Neuronales profundas para segmentación semántica

El problema de segmentación de imágenes pertenece también al paradigma supervisado, sin embargo en lugar de asignarle una categoría a la imagen completa lo que se busca es clasificar píxel a píxel, asignando cada uno a una cierta categoría o entidad presente en la imagen (Sultana et al. 2020). La asignación de píxeles a una categoría se conoce como segmentación semántica. En este caso los píxeles se asignan sin considerar a la entidad particular a la que pertenecen. Por otro la asignación de píxeles a entidades individuales se conoce como segmentación por instancia. Este tipo de segmentación es en realidad dos procesos en uno, primero se deben identificar las entidades con un modelo localizador que retorne un *bounding-boxes* por instancia. Luego se obtiene una máscara que segmenta el contorno de la entidad previamente localizada. Debido a esto la segmentación por instancia es en general más costosa que la segmentación semántica. La Figura 4 muestra la diferencia entre los resultados de los métodos de segmentación semántica y por instancia. En este ejemplo particular la segmentación semántica clasifica cada píxel ya sea como persona o como fondo, mientas que la segmentación por instancia identifica los píxeles de cada persona.

Dos arquitecturas de redes profundas ampliamente utilizadas en segmentación semántica son la Fully-convolutional network (FCN) (Long et al. 2015) y la U-net (Ronneberger et al. 2015). La FCN es una red neuronal que ocupa sólo capas de convolución, pooling y convolución traspuesta (Dumoulin y Visin, 2016). Después de cada convolución se aplica una función de activación no lineal Rectified Linear Unit (ReLU), para luego submuestrear mediante max pooling. La FCN está formada por seis bloques de convolución, activación y submuestreo, concatenados. El tamaño de los feature maps disminuye progresivamente a la vez que la cantidad de los filtros aumenta. La última capa es una convolución traspuesta con paso largo que aumenta el tamaño del feature map hasta hacerlo equivalente al tamaño de la imagen original. Para entrenar la FCN se requiere una etiqueta del mismo tamaño de la imagen original donde cada píxel tiene asignado una entidad. La FCN fue diseñada originalmente para segmentar imágenes naturales pero también ha sido aplicada

[2]https://www.image-net.org/

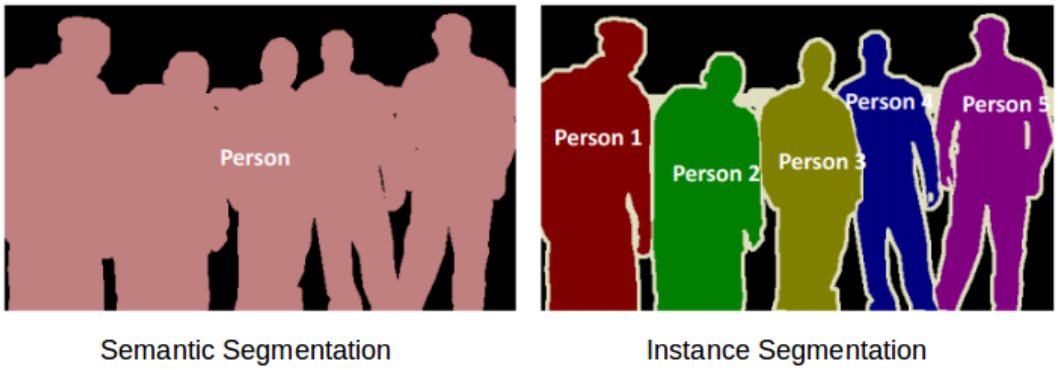


Figura 4. Segmentación semántica (izquierda) y segmentación por instancia (derecha). El primero entrega una máscara por clase o categoría mientras que el segundo entrega una máscara por cada entidad detectada.

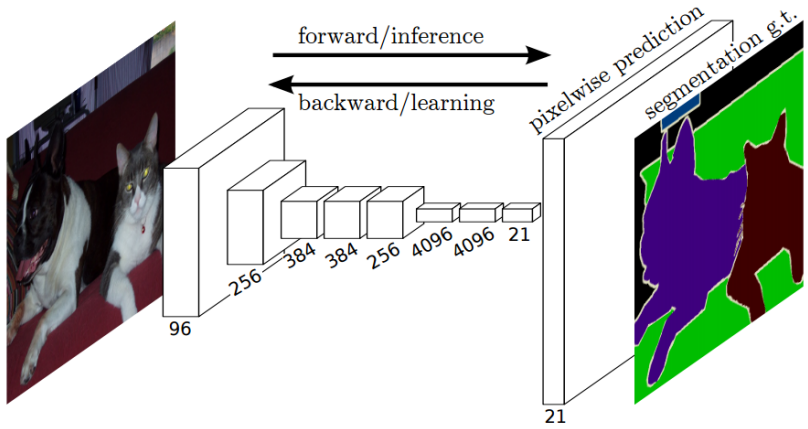


Figura 5. Arquitectura original de la Fully Convolutional Network para segmentación semántica

a imágenes de microscopio (Xie et al. 2018). La estructura original de la FCN se puede apreciar en la Figura 5.

La U-net (Ronneberger et al. 2015) funciona de forma similar, pero fue creada específicamente para segmentación de imágenes biomédicas, siendo el modelo ganador del desafío de segmentación de imágenes de microscopio electrónico de transmisión (MET) organizado por la IEEE International Symposium on Biomedical Imaging (ISBI) en 2012 Arganda-Carreras et al. 2015. El desafío proporcionaba un dataset con imágenes de 30 secciones de cordón nervioso central de larvas de primer estadio de drosophilas a resolución 4x4x50 nanómetros por píxel con máscaras de segmentación binaria marcadas por neuroanatomístas expertos. Las imágenes son representativas de un proceso real, presentando ruido y pequeños errores de alineamiento. Esta dataset se describe en mayor

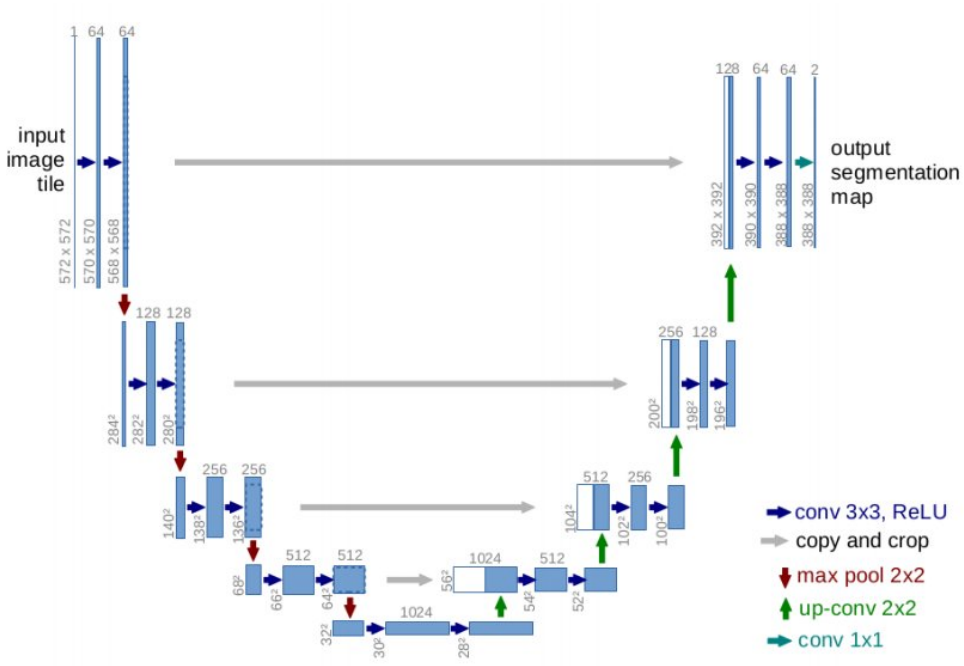


Figura 6. Arquitectura original de la U-net para segmentación semántica

detalle en la Sección 3.2.2.

La arquitectura de la U-net se puede apreciar en la Figura 6. A pesar de realizar la misma tarea que la FCN (segmentación semántica) su arquitectura revela dos diferencias importantes. La primera es que el aumento de dimensionalidad posterior al cuello de botella (bloque inferior en la figura) no es abrupto sino progresivo. Cada capa que reduce dimensión (flechas rojas en la figura) tiene una versión espejada que aumenta dimensión (flechas verdes en la figura), hasta recuperar el tamaño de la imagen original. En general se llama codificador o *encoder* al flujo que reduce o contrae la dimensión (capas de la izquierda en la figura), y decodificador o *decoder* al flujo que aumenta o expande la dimensión (capas de la derecha en la figura). La segunda diferencia es que los mapas de características del encoder se copian y concatenan a la salida de las capas del decoder (flechas grises en la figura). Esta transferencia de información entre los flujos de contracción y expansión de dimensionalidad mejora la resolución final de la imagen segmentada (Ronneberger et al. 2015).

2.4. Algoritmo Watershed para segmentación de instancia

Los modelos de redes neuronales anteriormente descritos pertenecen al paradigma de segmentación semántica, es decir que asignan una clasificación para cada píxel irrespecto de la entidad a la que pertenecen. En este caso particular dicha clasificación es binaria

y la salida del modelo corresponde a la probabilidad de pertenecer a una membrana o pared celular. Una vez se tiene esta máscara se debe reconocer e identificar cada una de las células de la imagen. Para lograr esto se propone utilizar un algoritmo de segmentación por instancia que reciba la máscara de segmentación semántica como entrada. El algoritmo seleccionado en este trabajo se conoce como transformada watershed[3] (Roerdink y Meijster, 2000) y fue originalmente propuesto por Digabel y Lantuéjoul, 1978.

La transformada watershed pertenece a la familia de los operadores morfológicos (Chen et al. 2017) y ha sido ampliamente utilizada para procesar imágenes biológicas y médicas (Benson et al. 2015). El algoritmo general le da una interpretación topográfica a los valores de los píxeles, es decir que la imagen se considera como una mapa de relieves. El objetivo del algoritmo es encontrar las cuencas, es decir, los conjuntos de píxeles cuya dirección de máximo gradiente negativo converge a un mismo mínimo local. Los píxeles que separan estas cuencas corresponde a las "montañas" o lineas divisorias.

A continuación se revisa el algoritmo clásico para imágenes en escala de grises (Kornilov y Safonov, 2018; Vincent y Soille, 1991) conocido como método de inundación. El primer paso consiste en encontrar los mínimos locales de la imagen y asignarle a cada uno un marcador único o etiqueta. La selección de los marcadores se hace típicamente en base a la imagen de gradientes o primeras diferencias entre píxeles. Cabe destacar que este es un paso bastante sensible del algoritmo y que una incorrecta selección de los marcadores puede provocar sobresegmentación (Kornilov y Safonov, 2018).

El segundo paso es simular el proceso de "inundación" a partir de los marcadores seleccionados. Para esto se utilizan H colas de prioridad donde H son los niveles de intensidad de la imagen, siendo 256 un valor típico. Los píxeles con intensidad h ingresan a la cola h con un nivel de prioridad según la magnitud de su gradiente, es decir que los píxeles con mayor gradiente se extraen primero de la cola. En primer lugar se ingresan los marcadores, es decir los mínimos locales. Luego por cada píxel extraído de la cola se revisan sus vecinos más cercanos, si el píxel extraído tenía una marca esta se traspasa a sus vecinos que luego ingresan a su cola correspondiente. El algoritmo termina cuando todas las colas están vacías. La Figura 7 muestra un esquemático de este proceso.

2.5. Estado del arte en segmentación celular

La U-net ha sido una arquitectura ampliamente utilizada en problemas de segmentación de imágenes biológicas y médicas. En específico, desde el 2015 a la fecha, la utilización de la U-net a visto un aumento exponencial, según muestra la Figura 8. Esta figura corresponde a la cantidad de artículos escritos sobre segmentación celular usando U-net según la plataforma *Google Scholar*. Se seleccionan los artículos más relevantes a este proyecto

[3]Cuya traducción literal es cuenca hidrográfica

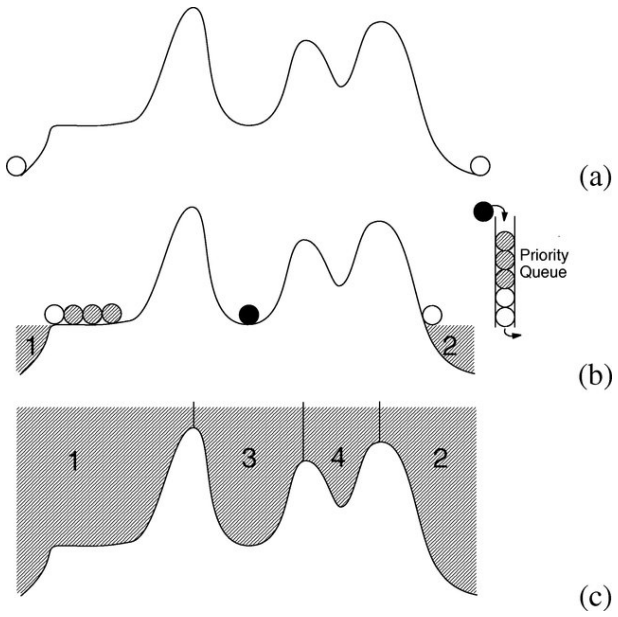


Figura 7. Ilustración del proceso de inundación descrito en (Vincent y Soille, 1991) para una fila/columna de una imagen. La subfigura (a) muestra el primer nivel de intensidad el cual presenta dos mínimos locales. La subfigura (b) muestra otro nivel de intensidad que tiene un mínimo local. La subfigura (c) muestra el resultado de la propagación de las marcas únicas de los mínimos locales.

para ser revisados a continuación.

Paulauskaite-Taraseviciene et al. 2019 comparó un modelo basado en U-Net y otro modelo basado en Mask R-CNN(He et al. 2017) con el objetivo de detectar y cuantificar células en imágenes de microscopía de fluorescencia. El objetivo era identificar cual modelo realiza una segmentación más robusta ante problemas tales como el exceso de iluminación, la presencia de células sobrepuestas y la segmentación de células ocluidas o parcialmente visibles. El dataset utilizado contiene 300 imágenes utilizadas en el desafío Kaggle Data Science Bowl 2018 (KDSB18)[4]. Las imágenes están marcadas con un cuadro al rededor de cada célula y están afectas a distintos tipos de ruido. Los resultados en términos de *accuraccy* fueron de $96{,}8 \pm 0{,}59$ % para la U-net y un $94{,}5 \pm 2{,}08$ % para la Mask R-CNN. La mask R-CNN no solo presenta un desempeño peor sino también mayor inestabilidad entre distintos entrenamiento a pesar de ser un modelo más moderno que la U-net.

Una modificación de la arquitectura U-net fue recientemente propuesta por Punn y Agarwal, 2020 con el fin de mejorar su desempeño en la segmentación de núcleos celulares. El entrenamiento y las pruebas realizadas consideran el dataset KDSB18 previamente descrito. La modificación principal consiste en reemplazar las capas

[4]https://www.kaggle.com/c/data-science-bowl-2018

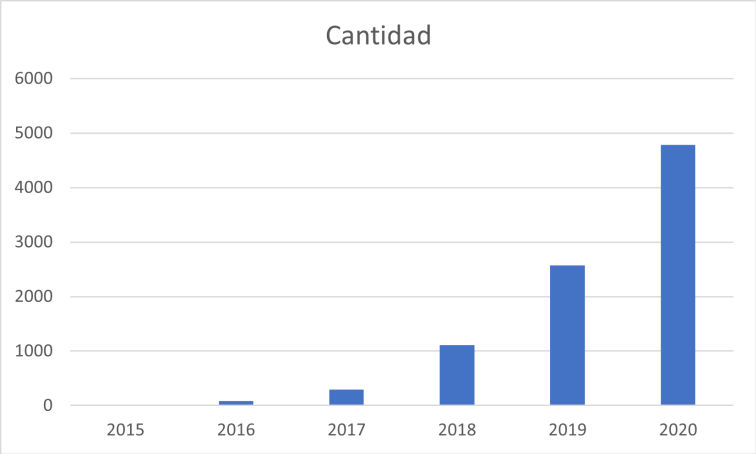


Figura 8. Cantidad de artículos donde se utiliza la arquitectura U-net para segmentación celular

convolucionales de la U-net tradicional con bloques de tipo *inception*. Este bloque fue propuesto por primera vez como parte del modelo GoogLenet, la entrada ganadora del ImageNet Large-Scale Visual Recognition Challenge 2014 (Szegedy et al. 2015). Un desafío al momento de usar capas convolucionales es seleccionar adecuadamente el tamaño de los filtro de para cada tarea. El bloque *inception* resuelve este problema por medio de la aplicación de múltiples operadores en paralelo. Para este caso particular se utilizaron tres convoluciones con tamaños 1x1, 3x3, 5x5, respectivamente, y un operador de pooling híbrido. Cabe aclarar que si bien el operador de 1x1 no tiene vecindad espacial es totalmente capaz de combinar los distintos canales de la imagen. Los mapas de características que se obtienen a la salida de los cuatro operadores del bloque *inception* se concatenan antes de entrar en el bloque siguiente, actuando como un filtro multi-resolución. El modelo puede escoger dinámicamente cual o cuales de estos filtros ocupar en cada nivel durante al entrenamiento por lo que puede considerarse una búsqueda de arquitectura óptima basada en los datos.

El operador de pooling híbrido es también otro diferenciador con respecto a la arquitectura U-net tradicional y consiste de la aplicación consecutiva de un operador de pooling espectral y el ampliamente utilizado MaxPooling. El pooling espectral recibe su nombre debido a que la operación de submuestreo se aplica en el dominio de la frecuencia por medio de la transformada discreta de Hartley (Zhang y Ma, 2020). En la palabras de los autores “esto permite preservar mejor las estructuras espaciales de alta frecuencia” con respecto a los operadores típicos de pooling que actúan directamente sobre la imagen (Punn y Agarwal, 2020). Para comparar los distintos modelos se utilizó la métrica de accuracy e Intersection over Union (IoU) (Tychsen-Smith y Petersson, 2018) sobre las imágenes del conjunto de validación. El modelo U-net clásico obtiene un accuracy de 0.9719 y un IoU de 0.5981, mientras que el U-net con bloques inception y pooling híbrido

alcanza un accuracy de 0.9745 y un IoU de 0.6523. Cabe resaltar que si bien los bloques *inception* pueden dar una ligera ventaja en términos de resultados, en general se requerirá de una mayor cantidad de datos debido a la introducción de parámetros entrenables adicionales, lo cual puede resultar poco práctico cuando se tienen bases de datos acotadas.

Otro modificación interesante a la arquitectura U-net es la presentada por Ke et al. 2019, donde se proponen y comparan dos rediseños del bloque codificador de la U-net para mejorar su desempeño en la detección de células cervicales a partir de biopsias. El objetivo último es el diseño de un sistema para detectar y diagnosticar displasia o cáncer. El dataset utilizado contiene 100 imágenes de entrenamiento y 40 imágenes de prueba, de aproximadamente 900x900 píxeles obtenidas de 6 hospitales en un lapso de 18 meses etiquetadas por patólogos expertos. El primer bloque codificador propuesto está basado en la arquitectura VGG16 (Simonyan y Zisserman, 2014) con 12 capas convolucionales. Debido a la cantidad de parámetros de esta arquitectura los autores proponen utilizar un esquema de ajuste fino con *transfer-learning*, donde sólo se entrenan los parámetros de las capas más cercanas a la salida. Esto es notable pues en este caso el dataset objetivo es muy distinto a ImageNet, el dataset original de imágenes naturales usado para entrenar VGG. El segundo bloque codificador Hybrid Standard and Dilated Convolution (HSDC), que consiste en reemplazar las capas convolucionales del encoder original de la U-net con bloques convoluciones dilatadas en paralelo y luego combinan el resultado. En términos de resultados la U-net con encoder VGG16 alcanza un *accuracy* promedio de 0.89, mientras que la U-net con bloques HSDC obtiene 0.879, aunque esta diferencia podría no ser significativa dado el pequeño tamaño del dataset de prueba. Los autores tampoco contrastan con una arquitectura U-net convencional, por lo que tampoco es posible notar las ventajas de cambiar el bloque codificador por las arquitecturas propuestas.

3. METODOLOGÍA

Para la gestión del proyecto se utiliza como metodología general el método de Kanban, ya que este facilita la visualización, identificación e inclusión de nuevas tareas. Además, los retrasos se aprecian al instante respecto a la planificación, por lo que es sencillo reconocer cuando hay que acelerar una actividad. Este método se aplica en las siguientes tres etapas:

1. Investigación: Se investiga el estado del arte, tanto en el ámbito del procesamiento de imágenes con foco en segmentación celular, como también con respecto a la utilización de los algoritmos de segmentación y tracking en imágenes digitales. A partir de la investigación se logra dilucidar, aquello que podía servir para el cumplimiento de los objetivos propuestos en particular que modelo y que arquitectura de red neuronal se iban a utilizar, qué métricas son más apropiadas para evaluar, como validar, etc.

2. Implementación y Desarrollo: Se utiliza un modelo iterativo incremental con reuniones semanales con el profesor patrocinante, además de algunas con el profesor co-patrocinante en la medida que fuera necesario. En todas las iteraciones se repite un proceso de trabajo similar mostrando los avances del desarrollo, para proporcionar un resultado completo sobre el producto final. Además en cada iteración se evoluciona el producto a partir de los resultados completados en las sesiones anteriores, añadiendo nuevos objetivos o mejorando los actuales.

3. Mediciones (Validación): Se validan los avances con respecto al estado del arte viendo cuales son los mejores pasos a seguir para lograr el mejor resultado posible. Además con el profesor co-patrocinante se valida tanto la usabilidad del producto como también el resultado esperado.

3.1. Datos

En particular se consideran dos datasets, ambos resultados de un proceso de captura de imágenes utilizando técnicas de microscopía confocal. Nos referiremos a estos como dataset A el cual fue captado de embriones de pez cebra en la visualización de la organogénesis del sistema nervioso central (neurulación) por el Dr. Claudio Araya del Instituto de Ciencias Marinas y Limnológicas de la Universidad Austral de Chile. El dataset B captado en Drosophila Melanogaster durante el cierre dorsal en el proceso de gastrulación (Solon et al. 2009).

El dataset A tiene extensión CZI[5], un formato propietario de la empresa Zeiss que es estándar en microscopía y ampliamente utilizado en investigaciones biomédicas. Este

[5]https://www.zeiss.com/microscopy/int/products/microscope-software/zen/czi.html

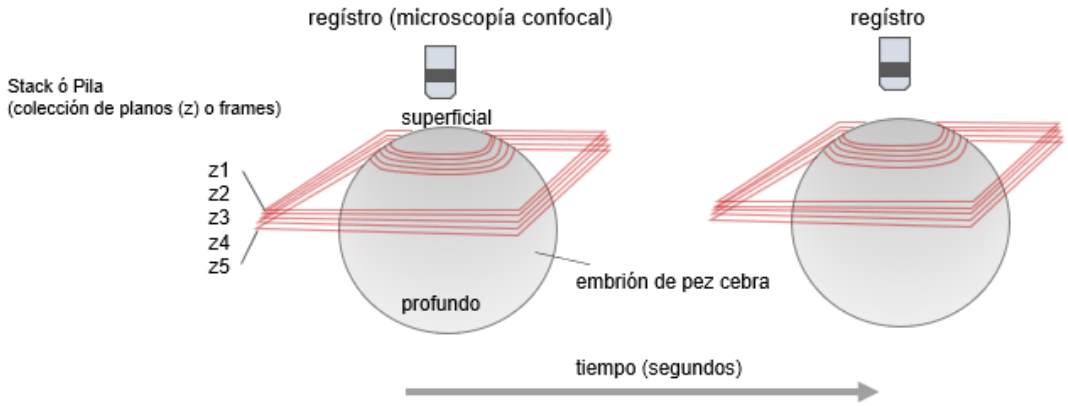


Figura 9. Representación del tensor de imágenes de cuatro dimensiones asociado al Dataset A.

formato es un contenedor capaz de guardar archivos de datos de imagen de microscopio en forma de pilas (stack) de imágenes, series de lapso de tiempo y mosaicos.

Para tener acceso a los píxeles del contenedor desde Python se utiliza la librería CziFile versión 2019.7.2[6]. Los datos importados se presentan como valores numéricos organizados en un tensor de cuatro dimensiones, tres dimensiones espaciales y una dimensión temporal, es decir una pila de imágenes de un solo canal que evoluciona en el tiempo. Por cada archivo se tienen 17 capas de imágenes en el eje Z (altura) y 60 frames resultando en un total de 1020 imágenes por archivo. Además el archivo cuenta con una unidad de medida de 0.2380157 micras por píxel de la imagen y un tiempo de muestreo o tiempo entre frames de 113.65 segundos. La Figura 9 muestra una representación esquemática del dataset A.

El dataset B tiene extensión .TIFF, el cual es un formato de almacenamiento que evita la perdida de información (resolución) y admite una profundidad de color de 64 bits.Es un formato de archivos de gráficos por trama ampliamente utilizado y desarrollado por Aldus y Microsoft en el año 1986 y es actualmente propiedad de Adobe Systems. Este formato puede contener secuencias de imágenes al igual que el formato CZI. Para importar el dataset B se utiliza la librería PIL[7], resultando en un tensor de 149 frames con imágenes de un canal. A diferencia del dataset A este no cuenta con eje Z. Por ultimo el archivo cuenta con una unidad de medida de 0.2374839 micras por píxel de la imagen y un tiempo de muestreo o tiempo entre frames de 9.96 segundos. La primera imagen de esta secuencia se aprecia en la Figura 10. Un resumen con las características más importantes de ambos dataset se muestra en la tabla 1.

[6]https://pypi.org/project/czifile/
[7]https://pypi.org/project/Pillow/

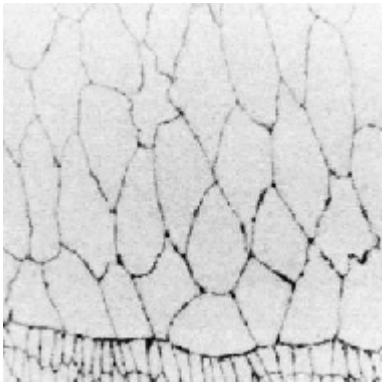


Figura 10. Primer cuadro del Dataset B.

	Dataset A	Dataset B
Formato	CZI	TIFF
Número de cuadros	60	149
Profundidad	17	1
Tamaño de píxel [μm]	0.2380157	0.2374839
Tiempo de muestreo [s]	113.65	20.0

Tabla 1. Datasets de imágenes de microscopio evaluados en este trabajo

3.2. Sistema

En esta sección se describe el sistema propuesto para detección de patrones a partir de secuencias de imágenes de microscopio. El sistema se divide en etapas encadenadas denominadas (1) pre-procesamiento, (2) segmentación semántica, (3) identificación y (4) tracking, las cuales se describen en detalle a continuación. La Figura 11 muestra un diagrama de bloques del sistema que se utilizara para llevar a cabo el proyecto.

Los códigos que soportan este trabajo fueron desarrollados utilizando el lenguaje Python versión 3.6.10 y el ambiente de cómputo científico Jupyter versión 6.0.3, este ejecutado por medio de Anaconda navigator versión 1.9.12 que es una interfaz gráfica de usuario para administrar las diferentes versiones de los paquetes mediante el sistema de gestión de paquetes Conda. Esta decisión se fundamenta en la amplia disponibilidad de librerías que existen y facilitan este proceso. Las librerías utilizadas para procesar las imágenes en este trabajo son OpenCV versión 3.3.1[8] (Bradski, 2000), NumPy version 1.18.1 y Pytorch versión 1.5.0[9] (Paszke et al. 2019). Para la visualización de resultados se ocupa Matplotlib

[8]https://docs.opencv.org/3.3.0/

[9]https://pytorch.org/docs/1.5.0/

versión 2.2.2.

3.2.1. Pre-procesamiento

Las imágenes de los dataset A y B se importan como tensor utilizando los librerías CziFile y PIL, respectivamente, como fue descrito en la Sección 3.1. Debido a que la importación es un proceso lento y con el objetivo de facilitar los experimentos posteriores se almacenan en disco las imágenes en formato PNG sin pérdida en escala de grises como se aprecia en la Figura 12.

Previo a la segmentación semántica y luego de cargar la imagen desde disco se aplican dos procedimientos con el objetivo de disminuir el nivel de ruido y mejorar el contraste, respectivamente. A continuación se describen estos procedimientos en detalle.

- Filtro bilateral: Es un filtro no lineal diseñado para atenuar el ruido manteniendo la mayor nitidez posible de los bordes presentes en la imagen (Elad, 2002). El filtro modifica la intensidad de cada píxel p de la imagen reemplazándola por un promedio ponderada de los valores de intensidad de los píxeles cercanos en una vecindad Q según la siguiente fórmula

$$I^f[p] = \frac{1}{W_p} \sum_{q \in Q} G_{\sigma_s}(\|p - q\|) \cdot G_{\sigma_r}(\|I[p] - I[q]\|) \cdot I[q], \tag{1}$$

  donde $G_\sigma(\cdot)$ es una kernel o filtro Gaussiano. Podemos ver que ponderación de $I[q]$ considera dos términos, el primero es función de la diferencia en posición de los píxeles $\|p - q\|$ mientras que el segundo es función de la diferencia en valor de los píxeles $\|I[p] - I[q]\|$. La primera suaviza el ruido (pasa bajo) mientras que la segunda preserva los bordes o cambios abruptos (pasa alto). El balance entre ambas se configura con los parámetros σ_s y σ_r, es decir los anchos de los kernel. Finalmente W_p es un factor de normalización. Una desventaja de este filtro es que es lento en términos de tiempo de cómputo en comparación con la mayoría de los filtros típicamente usados (lineales).

  Para implementar el filtro se utiliza la función `bilateralFilter` de la librería OpenCV con parámetros:

  1. `d`: Define el diámetro de cada vecindario de píxeles. Fuera de este diámetro la ponderación es cero. Se usa un valor de 5.
  2. `sigmaSpace`: Corresponde a σ_s en la Ecuación (1), es decir el ancho del kernel que compara las coordenadas o posiciones de los píxeles. Cuanto mayor sea su valor, mayor será la ponderación de los píxeles vecinos y mayor será el suavizado del filtro. Se usa el valor por defecto de 75.

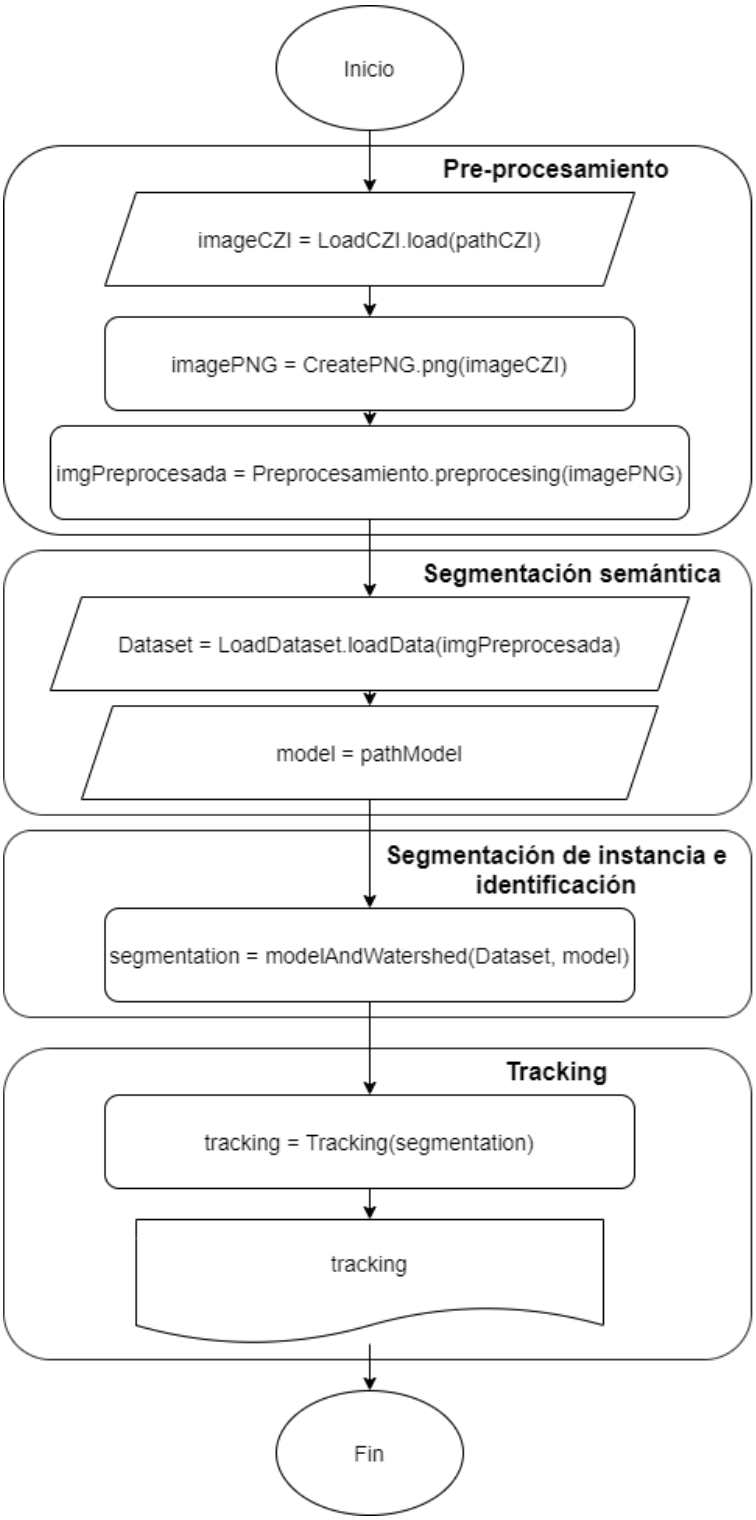


Figura 11. Diagrama de bloques del sistema propuesto

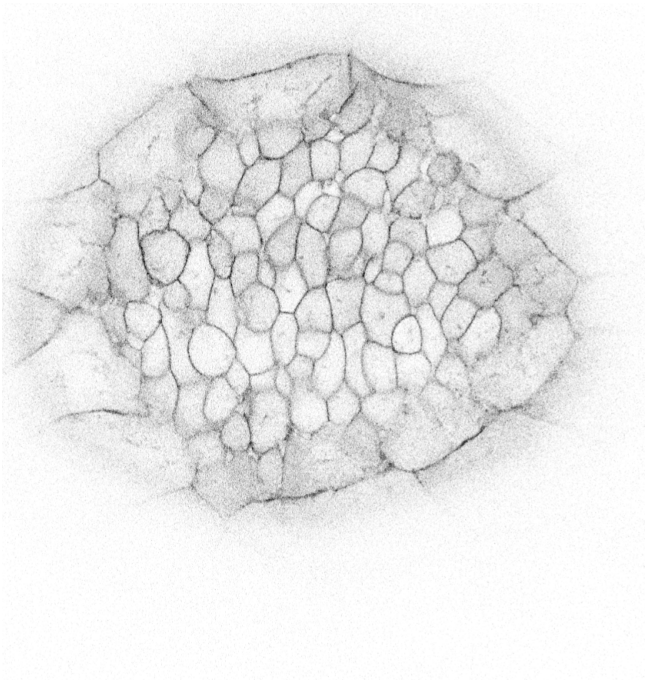
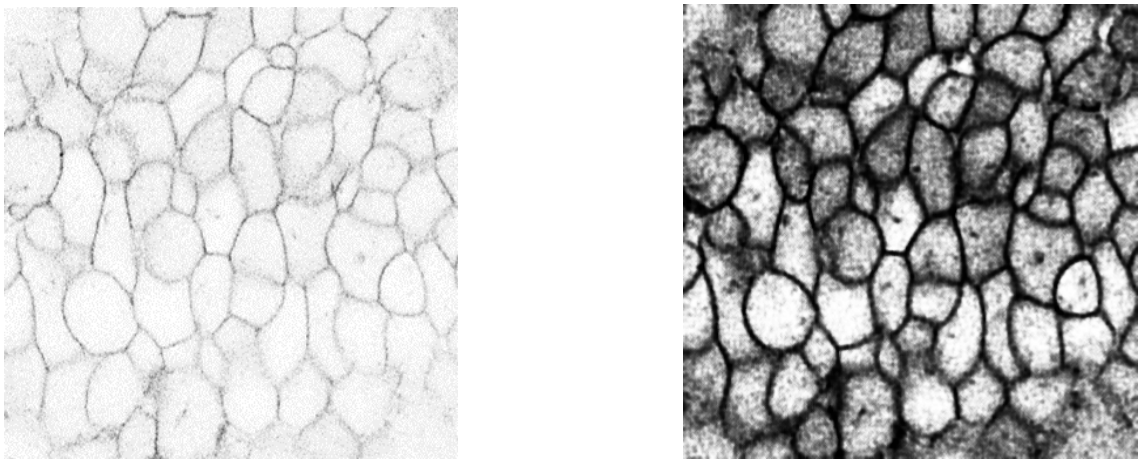


Figura 12. Primer cuadro del tensor de imágenes importado con la librería CziFile a partir del dataset A. La secuencia de imágenes se guarda en disco en formato PNG sin pérdidas.

3. `sigmaColor`: Corresponde a σ_r en la Ecuación (1), es decir el ancho del kernel que compara intensidades de color. Cuanto mayor sea el valor, los colores más alejados en valor comenzarán a mezclarse. Se usa el valor por defecto de 75.

- Ecualización de histograma: Primero se calcula el histograma de la imagen en escala de grises, luego se normaliza de modo que la suma de los intervalos del mismo sea 255, después se calcula la integral de este que finalmente se utiliza para trasformar la imagen usando una tabla de búsqueda. El resultado es que mejora el contraste de una zona de la imagen de forma local. Para implementar el filtro se utiliza la función `equalizeHist` de la librería OpenCV que recibe como parámetro la imagen original en escala de grises.

El objetivo de estos filtros estáticos es que las imágenes sean más similares a las del conjunto de datos utilizado para entrenar el modelo de segmentación selecciona (ver sección siguiente). La Figura 13 muestra una imagen (a) que es recorte de la Figura 5 mostrada anteriormente, además de otra figura (b) que muestra como se ve la imagen luego de aplicar los procedimientos antes descritos.



(a) (b)

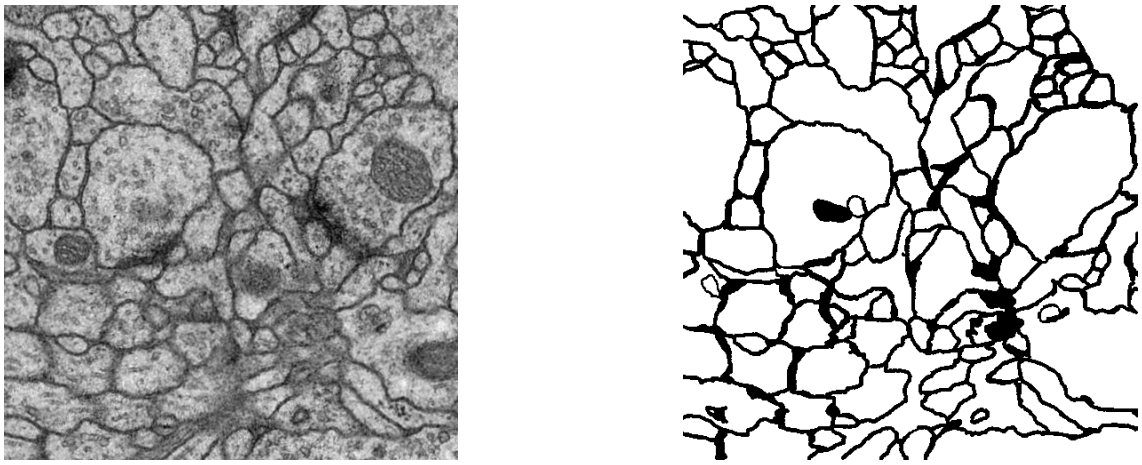
Figura 13. Recorte en escala de grises de la primera imagen de la secuencia (a) y resultado de aplicar la rutina de preprocesamiento (b).

3.2.2. Segmentación semántica

Para generar las máscaras binarias de segmentación de bordes celulares se utiliza un modelo de red neuronal basado en la arquitectura U-net cuyos parámetros fueron ajustados previamente con la base de datos del desafío de segmentación de imágenes del IEEE ISBI del año 2012 (Arganda-Carreras et al. 2015). Este conjunto consiste de 30 secciones del cordón nervioso ventral de larvas de primer estadio de Drosophilas. Cada imagen del dataset cuenta con una máscara de segmentación binaria, es decir una etiqueta binaria por cada píxel. La etiqueta representa si el píxel en cuestión corresponde a una membrana celular o no. La Figura 14 muestra una de las imágenes del conjunto de datos junto a su máscara. Este dataset fue escogido ya que sus imágenes son las más semejantes que se lograron encontrar en la literatura respecto a las que se evalúan en este trabajo.

La implementación del modelo fue desarrollada con la librería PyTorch[10] (Paszke et al. 2019), un paquete de Python 3 diseñado para realizar cálculos numéricos haciendo uso de la manipulación de tensores que además proporciona funciones de alto nivel para programar arquitecturas de redes neuronales profundas. Una ventaja de PyTorch es que los tensores pueden traspasarse de forma transparente a Unidades de Procesamiento Gráfico (GPU) acelerando notablemente su procesamiento. Esta librería es usada tanto como una alternativa de mayor rendimiento para la librería de cómputo numérico NumPy como para realizar cálculos en GPU en la investigación y desarrollo de redes neuronales artificiales. Al contrario que otros paquetes de características similares como Tensorflow, PyTorch

[10]https://pytorch.org/



(a) (b)

Figura 14. Ejemplo de imagen (a) del conjunto de datos del desafío de segmentación de ISBN 2012 con su respectiva etiqueta binaria (b).

trabaja con grafos dinámicos en vez de estáticos. Esto significa que en tiempo de ejecución se pueden ir modificando las funciones y el cálculo del gradiente variará con ellas.

El modelo utilizado tiene la misma arquitectura que la U-Net original pero con ligeras modificaciones para disminuir la cantidad de parámetros entrenables del modelo, disminuyendo así la posibilidad de sobreajustarse a un conjunto de datos pequeños como es el del desafío ISBN 2012. Al igual que la U-net original la entrada del modelo corresponde a imágenes mono-canal de 572x572 píxeles y la máscara de segmentación o salida del modelo es de 388x388 píxeles. La red se aplica sobre imágenes de 388x388 que deben ser extendidas artificialmente al tamaño 572x572 mediante `padding`. Con respecto a la U-Net original, el modelo modificado reduce la cantidad de filtros convolucionales a la mitad para todas las capas, manteniendo todos los demás parámetros intactos. El esquema del modelo U-Net modificado se muestra en la Figura 15. Los códigos asociados a esta implementación se distribuyen libremente bajo licencia MIT[11]

Los parámetros del modelo luego de haber sido entrenado en el dataset del desafío ISBN 2012 fueron puestos a libre disposición por los autores y se pueden descargar desde dropbox[12]. El modelo escogido es el entrenado con RMSprop(Zou et al. 2019), ya que fue el que alcanzó la máxima exactitud (accuracy) en validación. Para crear una instancia del modelo en PyTorch con los parámetros pre-entrenados sólo es necesario descargar el archivo mencionado y cargar el modelo en el ambiente de desarrollo como muestra el

[11]https://github.com/ugent-korea/pytorch-unet-segmentation

[12]https://www.dropbox.com/s/cdwltzhbs3tiiwb/model_epoch_440.pwf?dl=0

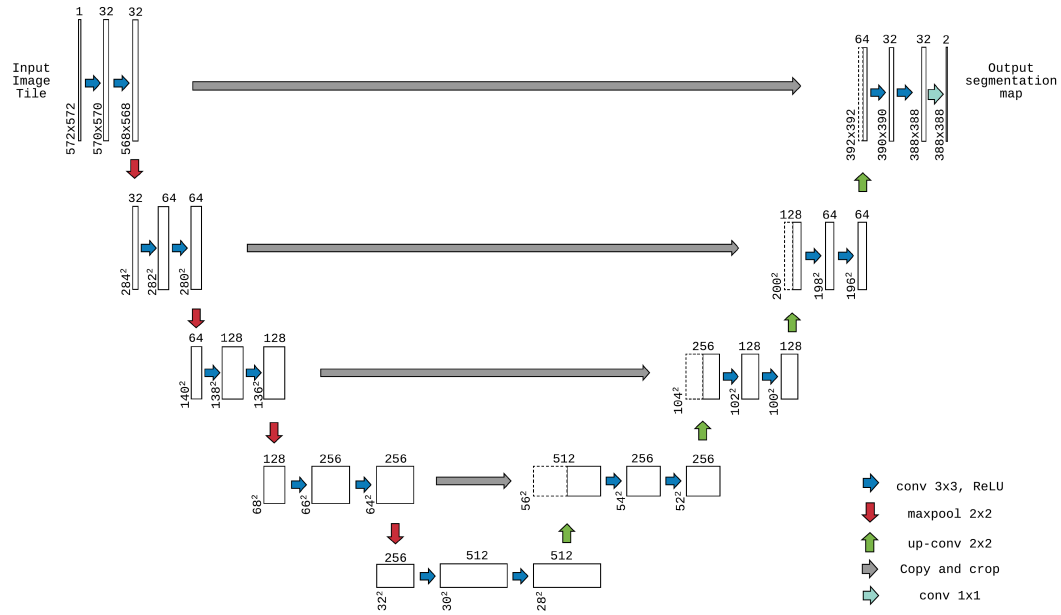


Figura 15. Arquitectura del modelo U-Net modificado. Este modelo reduce la cantidad de filtros a la mitad con respecto a la versión original.

Cuadro 1.

```
import torch.nn as nn
path='pytorch-unet-segmentation-master/modelos/model_epoch_440.pwf'
model=torch.load(path, map_location=torch.device('cpu')).module.cpu()
```

Cuadro 1. Carga del modelo pre-entrenado con datos del desafío ISBN 2021

El conjunto de datos del desafío ISBN 2012 contiene pocos ejemplos. Debido a esto se optó por realizar una aumentación sintética sobre las imágenes con las siguientes estrategias:

1. Gaussian noise: Se le aplica ruido Gaussiano a las imágenes. Este tipo de ruido agrega más ruido a los medios tonos y menos ruido a las sombras y las regiones resaltadas de la imagen.

2. Uniform noise: Se aplica ruido uniforme, como sugiere el nombre, el uso de esta opción agrega ruido aleatorio de igual intensidad en toda la imagen. El efecto es notablemente más sutil que el ruido Gaussiano.

3. Brightness: Se modifica el brillo de las imágenes.

4. Flip: Se traspone y espeja la imagen.

5. Elastic deformation: Se deforma la imagen, de esta manera las deformaciones realistas se pueden simular de manera eficiente.

Finalmente la imagen se recorta y luego se aplica un *padding* simétrico mediante reflejos verticales y horizontales para hacerla compatible con las dimensión de entrada esperada por la red neuronal. Para evaluar este modelo en otras imágenes es necesario replicar este preprocesamiento

Para entregar los datos al modelo es necesario implementar un objeto DataSet y DataLoader de PyTorch. El objeto Dataset recibe una ruta a la carpeta con las imágenes en formato PNG y además una serie de transformaciones de la librería torchvision que se aplicarán sobre ellas al momento de evaluarlas por la red neuronal:

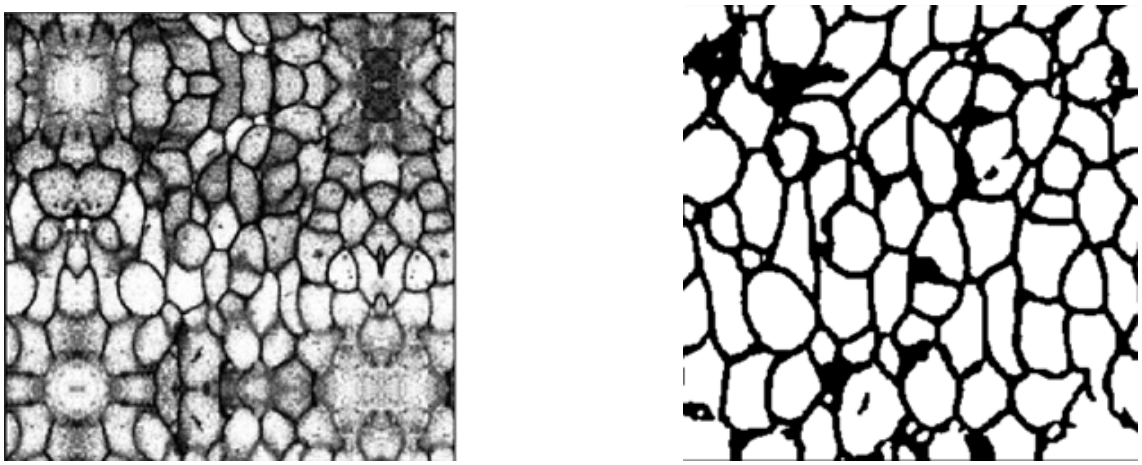
1. `Grayscale`: Convierte imágenes a color a escala de grises.
2. `Pad`: Extiende la imagen de 388x388 píxeles en todas direcciones resultando en una imagen de 572x572. Se utiliza el modo reflejo el cual rellena espejando los píxeles.
3. `ToTensor`: Convierte una imagen en formato PIL o numpy array a formato Tensor de PyTorch.

Con estas transformaciones las imágenes quedan como se muestra en la Figura 16 (a). Por otro lado, el Dataloader representa un iterable sobre un conjunto de datos y recibe tres argumentos:

1. Dataset: el cual recibe como valor el dataset cargado anteriormente.
2. Batch_size: que define la cantidad de muestras que se propagarán a través de la red, en este caso se le da valor igual a 1.
3. Shuffle: el cual recibe el valor false, ya que las imágenes tienen que cargarse en orden, ya que son una secuencia de imágenes o "vídeo".

El resultado o salida de la U-net es una imagen donde cada píxel tiene asociado un valor entre 0 y 1 que representa la probabilidad de pertenecer a un borde celular. Esta imagen es equivalente en tamaño a la imagen original previo al proceso de *padding* quedando una imagen como se aprecia en la Figura 16 (b).

Luego de segmentar la imagen utilizando el modelo, se realiza una etapa de post-procesamiento sobre la máscara para mejorar su calidad. Se aplican las siguientes operaciones en orden:



(a) (b)

Figura 16. Imagen preprocesada y extendida con padding simétrico (a). Esta es la imagen que entra a la red neuronal. En (b) se muestra el resultado de segmentación semántica asociado a esta imagen.

1. Binarización: Se busca binarizar la imagen para poder identificar entre el fondo y la membrara o pared celular, esto se hace por medio de la función argmax() de torch. Esta función recibe un tensor de la imagen con píxeles entre 0 (fondo) y 1 (membrana), además de la dimensión a la cual se quiere reducir dicho tensor (en este caso dimensión 1). La función retorna un tensor con 0 o 1 dependiendo de si la probabilidad del píxel devuelto por la red es menor que 0.5 o no.

2. Transformación de arreglo: Se busca transformar el la salida del paso anterior a un formato compatible con la librería OpenCV. Para esto se utiliza la función detach().numpy() la cual recibe el tensor y lo transforma a un arreglo numpy de la misma dimensión pero multiplicándolo por 255. De esta forma los píxeles quedan como 0 (fondo) o 255 (membrana). Luego se utiliza la función `threshold` de CV2 para transformar la imagen a formato de openCV.

3. Dilatación: Mediante la dilatación se busca el enfatizar o aumentar la región blanca de imagen (primer plano), esto para poder conectar dichas regiones desprendidas entre sí y identificar de forma robusta los píxeles que corresponden al fondo. Se ocupa la función `dilate` del módulo CV2 que recibe como argumento la imagen y un kernel de unos de tamaño 3x3 que corresponde a la vecindad donde se aplica la operación morfológica.

3.2.3. Segmentación de instancia e identificación

La etapa anterior retorna una imagen segmentada semánticamente donde cada píxel pertenece a una de dos clases: fondo o membrana (pared celular). El objetivo de la etapa de identificación es asignar un marcador único a cada célula presente en la imagen a partir de la binarización de la etapa anterior. Para esto se utiliza la transformada *watershed*.

Se utiliza la implementación de la transformada *watershed* de la librería OpenCV. Para marcar las células es necesario realizar 4 pasos:

1. Transformada de distancia: En este paso se busca medir la distancia de cada píxel al valor 0 (limite) mas cercano. Esto estima que tan distante está el primer plano del fondo. Para hacer la transformación de distancia se utiliza la función `distanceTransform` de OpenCV. Esta función recibe como argumentos una imagen binarizada, un entero que indica la métrica distancia utilizar (en este caso la distancia euclidiana) y por ultimo el tamaño de la búsqueda.

2. Separación de fondo: Ahora que tenemos una idea de que tan alejado esta el primer plano del fondo, se busca decidir que píxeles de la imagen corresponden efectivamente al primer plano. Para esto se utiliza la función de `threshold` y `subtract` de OpenCV. Primero se aplica un valor de umbral para decidir qué parte seguramente podría ser el primer plano. El valor umbral es la distancia máxima (calculada a partir del paso anterior) multiplicada por un hiperparámetro que llamaremos "multiplicadorDistanciaMaxima", el cual debe sintonizarse manualmente. Cuanto mayor sea su valor mayor será el valor del umbral y, por lo tanto, obtendremos menos área de cierto primer plano. Mediante experimentación empírica con diferentes valores del hiperparámetro se decidió utilizar el valor de 0,1. A partir de los pasos anteriores, determinamos ciertas regiones de primer plano y de fondo. El resto se clasificará como regiones desconocidas.

3. Detección de mínimos locales: Previo a la segmentación de instancia con la transformada watershed es necesario encontrar un conjunto de puntos que actuarán como base de las cuencas. En este caso dichos puntos deberían ser muy cercanos al centro de cada célula. Para identificar los centros de los células se utiliza la función `connectedComponents` de openCV, la cual retorna un identificador único por cada marcador.

4. Segmentación de instancia: Para realizar la segmentación de instancia, la cual nos permitirá expandir el marcador a todos los píxeles de cada célula se utiliza la transformada watershed, la cual está implementada por la función `watershed` de OpenCV. Esta función recibe como argumentos la imagen binaria y los marcadores obtenidos en el paso anterior.

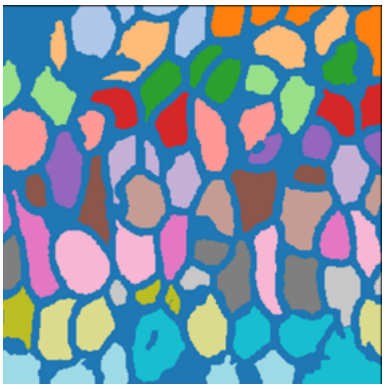


Figura 17. Resultado del algoritmo watershed a partir de la segmentación semántica entregada por la red neuronal

La Figura 17 muestra el resultado de aplicar el procedimiento descrito a la imagen binaria y post-procesada.

3.2.4. Tracking

La etapa anterior retorna un conjunto de células únicamente identificadas para cada una de las imágenes de la secuencia. Sin embargo no hay garantía de que una célula mantendrá la misma etiqueta a lo largo de la secuencia. Para poder obtener una serie del tiempo del área de una célula en particular a lo largo de la secuencia es necesario saber su posición en cada una de las imágenes.

La última etapa de esta metodología se encarga de hacer seguimiento o *tracking* a cada una de las células marcadas por el paso anterior. Para esto se utiliza un método inspirado en los vecinos mas cercanos, el cual se realiza en seis pasos:

1. El usuario escoge una célula en particular y se obtienen sus coordenadas centrales.
2. Se obtienen las coordenadas centrales de todas las células de la siguiente imagen.
3. Se calculan todas las distancias desde la célula seleccionada y se busca la célula mas cercana.
4. Se guarda el área y la posición de dicha célula.
5. Se actualiza la posición central de la célula inicial.
6. Se repite hasta que se hayan evaluado todas las imágenes de la secuencia.

Cabe resaltar ciertas validaciones realizadas para asegurar el correcto funcionamiento del algoritmo

1. Verificación de borde: Cuando el usuario escoge una célula, se verifica si esta se encuentra en el borde de la imagen o si existe otra con su mismo marcador. En este caso se le da un mensaje al usuario para que pueda escoger otra. Esto se hace ya que si la célula se encuentra en un borde no hay suficiente información para estimar de forma confiable el área de la misma.

2. Verificación de área: Luego de encontrar la célula más cercana en el cuadro siguiente, se compara su área A_{i+1} con el área de la célula en el cuadro actual A_i. Si $A_{i+1} \notin [d^{-1}A_{i+1}, dA_{i+1}]$, se asume que el vecino más cercano no corresponde a la célula en cuestión y por ende se conserva la posición y área actual. En general esta situación se da por errores en el frame o por una división física de la célula en cuestión. Se utiliza un valor conservador $d = 1{,}5$ para definir el rango aceptable de área. Este valor se valida empíricamente estudiando las discontinuidades de las series de tiempo de área.

3. Verificación de distancia: El algoritmo considera una entrada de usuario que corresponde a la máxima distancia en píxeles a la cual el centro de una célula puede desplazarse entre cuadros consecutivos. Para el dataset A se encuentra que un valor de 35 pixeles es suficiente, pero para el dataset B se utiliza un valor de 800. Si el algoritmo detecta que la célula más cercana esta demasiado lejos, se asume que la célula no se pudo segmentar con watershed posiblemente por que su borde no era lo suficientemente notorio. Si esto ocurre el valor de área para las siguiente iteraciones queda con valor nulo. A diferencia del paso anterior esto no da posibilidad a que el algoritmo se recupere.

Luego de hacer este procedimiento, se obtiene como resultado una serie de tiempo del área de la célula seleccionada. La Figura 18 muestra una visualización de ejemplo de una serie de tiempo en particular. Este tipo de visualización es de gran utilidad para los expertos pues permite observar las posibles oscilaciones exhibidas por la célula.

Notar que el procedimiento de segmentación de instancia retorna una máscara que indica los píxeles asociados a cada célula. Para una mejor interpretación de los resultados es necesario transformar el área de la célula en píxeles a unidades físicas. Para esto se extrae de los metadatos el tamaño de píxel en micrómetros (ver Tabla 1). El área de la célula se calcula como la cantidad de píxeles pertenecientes a la misma multiplicado por el cuadrado del tamaño de píxel. Esto se hace para cada cuadro. Finalmente se multiplica el índice secuencial de cada cuadro por el período de muestreo para transformar el eje horizontal de las series de tiempo a segundos.

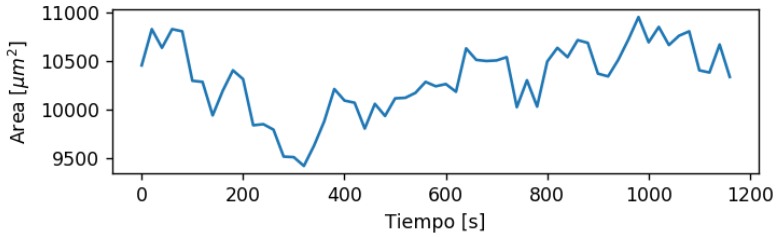


Figura 18. Área de la célula en el tiempo

3.3. Experimentos

Para medir y validar el proyecto se realizan 2 experimentos, los cuales son:

1. **Comparación de red neuronal con metodología clásica para segmentación semántica:** Con este experimento se busca validar la utilización de la red neuronal por sobre técnicas clásicas para hacer segmentación semántica. Se utiliza el primer cuadro del dataset A, el cual es etiquetado manualmente usando el programa "Gimp". Luego se obtiene la máscara binaria usando la red neuronal y un pipeline de procesamiento de imágenes clásico basado en umbrales dinámicos. La máscara retornada por ambos métodos se compara con la etiqueta en base a la entropía cruzada binaria, que se utiliza para medir el error de una reconstrucción. Esta es calculada con la formula:

$$L = -\sum_{n=1}^{N} y_n \cdot \log(x_n) + (1 - y_n) \cdot \log(1 - x_n),$$

   donde $x_n \in [0, 1]$ e $y_n \in \{0, 1\}$ corresponden a la predicción y etiqueta del píxel n, respectivamente. Para calcular la entropía cruzada binaria se utiliza la función `BCELoss` de la librería torch.nn, la cual recibe como argumentos la mascara calculada por la segmentación clásica o red neuronal y la etiqueta creada por el experto. Esta función es robusta a los casos límite $x_n = 0$ e $x_n = 1$.

2. **Estimación y validación del período de las secuencias de área celular:** En este experimento se muestra gráficamente como cambia el área de una célula en especifico a través del tiempo. Para este experimento se utilizan ambos dataset.

   Para medir la periodicidad de la serie de tiempo de área celular se utiliza la función de autocorrelación discreta

$$R[m] = \sum_{n \in \mathbb{Z}} x[n]x[n - m]$$

donde x es la serie de tiempo y m es el retardo. La auto correlación es equivalente a la correlación de la serie con una versión desplazada o retardada m unidades de tiempo de la misma. Si la autocorrelación es alta para un cierto retardo m^*, significa que la serie presenta un patrón que se repite cada m^* unidades de tiempo. El máximo de la autocorrelación siempre ocurre en $m = 0$. En este trabajo reportaremos el retardo asociado al segundo máximo de la autocorrelación como el candidato para el período fundamental de la serie. Como validación adicional la periodicidad calculada en ambos dataset se comparará con los resultados obtenidos originalmente por Solon et al. 2009 en su investigación.

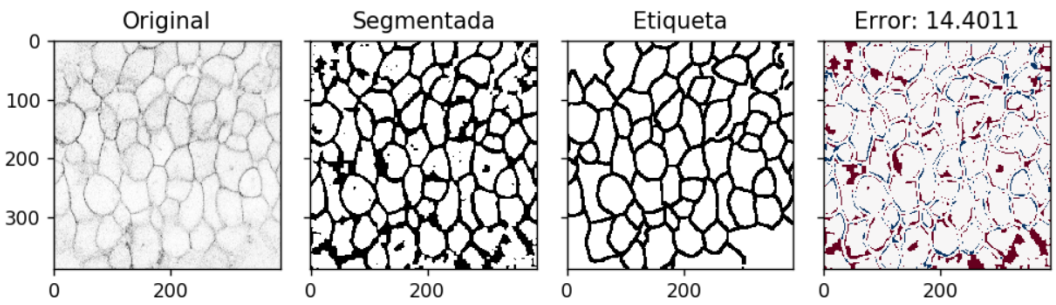


Figura 19. Segmentación clásica

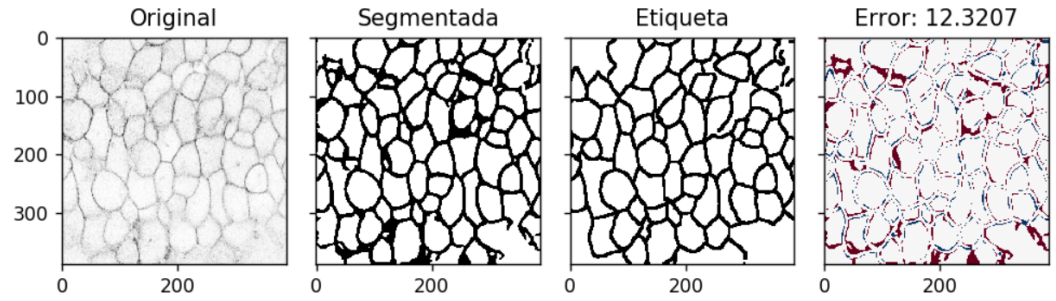


Figura 20. Segmentación U-net

4. RESULTADOS Y DISCUSIONES

4.1. Comparación de red neuronal con metodología clásica para segmentación semántica

Los resultados de esta comparación se pueden apreciar tanto visualmente como en el error calculado por la entropía cruzada binaria. En la figura 19 y en la figura 20, se aprecia como la red neuronal hace un mejor trabajo a la hora de generar la mascara binaria, teniendo menor ruido y mayor similitud a la etiqueta. Esto se ve en el error, ya que en el caso de la mascara generada por la segmentación clásica es de 14.4001, mientras que la generada por la red neuronal es de 12.3007.

Cabe destacar que gracias a este experimento se pudo calibrar el preprocesamiento de la red neuronal, la cual nos sirve luego para realizar la segmentación por medio de watershed y esto a su vez mejor el tracking por medio de los vecinos mas cercanos.

4.2. Estimación del período de las secuencias de área celular

En este experimento se evaluaron ambos dataset, primero de una forma cualitativa y luego cuantitativa. Para el análisis cualitativo se escogen tres células de cada dataset para

visualizar la segmentación cuadro a cuadro, la evolución del área celular y la función de autocorrelación. El análisis cuantitativo consiste en estimar la distribución del período para todas las células de la muestra con el objetivo de compararlas con los resultados de la literatura. Se estudia primero el dataset B y luego el A.

4.2.1. Dataset B

Los resultados de la segmentación de las tres células escogidas, se pueden apreciar en el la Figura 23, en donde se aprecia en cada fila un paso diferente del proceso. La primera fila muestra la imagen original, mientras que en la segunda fila se aprecia la imagen segmentada semánticamente usando la red neuronal U-net. Se aprecia que en general los bordes están bien identificados exceptuando algunas las células del centro de la imagen. Por último en la tercera fila se muestra la imagen segmentada por instancia resultante de aplicar la transformada watershed. Los errores del paso anterior claramente se arrastran a la segmentación de instancia. Cada columna de la imagen representa un cuadro distinto de la secuencia. Con esto se puede apreciar visualmente como varia la forma y tamaño de las células. Se destacan con números rojos tres células seleccionadas para estudiar sus series de tiempo de área celular a lo largo de la secuencia completa.

La figura 22, muestra las series de tiempo de área y su correspondiente función de autocorrelación para las tres células seleccionadas. La linea punteada roja marca el segundo máximo de la función de autocorrelación, es decir el retardo que se selecciona como la periodicidad estimada de la célula. En el título del gráfico de autocorrelación se muestra el valor de dicho retardo en segundos. En los tres casos seleccionados se puede ver que el período es cercano en los 200 segundos. En el caso de las células 1 y 3 se aprecian claramente los máximos locales en la función de autocorrelación. Por el contrario, en el caso de la célula 2, el período no es distinguible fácilmente a simple vista y la función de autocorrelación no tiene máximos locales tan claros. Cabe notar que el área en este caso particular tiene una tendencia descendiente constante, es decir que la serie no es estacionaria en su media, lo cual afecta la estimación de la autocorrelación.

Para un análisis cuantitativo más completo se repite el procedimiento anterior para un conjunto de siete células donde la segmentación fue efectiva, reportándose los períodos detectados a partir de la función de autocorrelación en la Tabla 2. Se puede apreciar que el periodo promedio es muy cercano a los 200 segundos y que la dispersión es baja. Adicionalmente podemos notar que no hay una relación clara entre la periodicidad de la oscilación y el tamaño de la célula (área media).

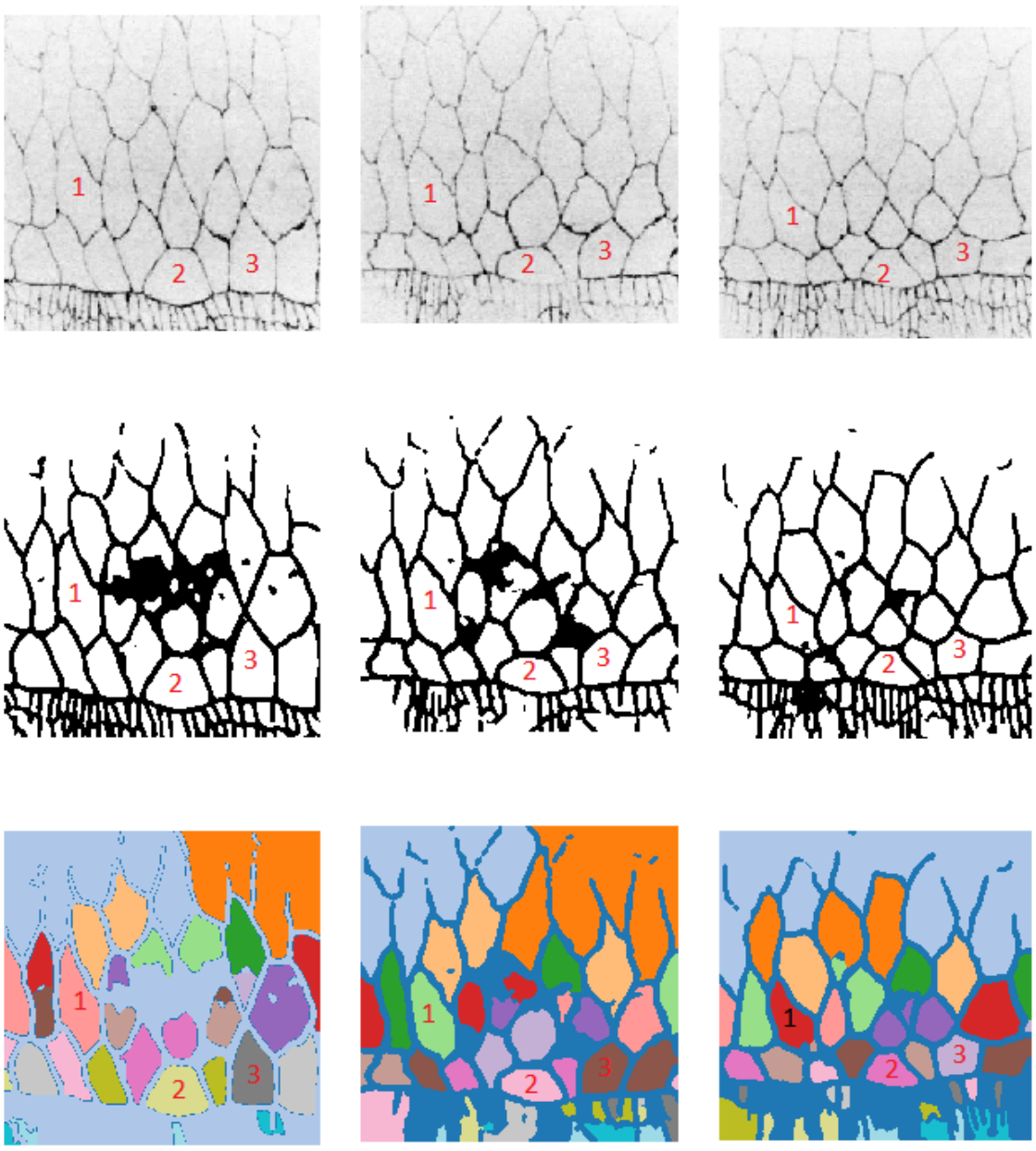
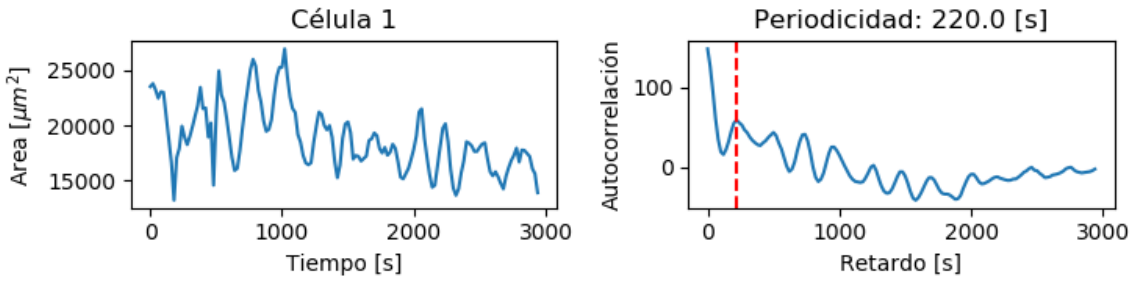
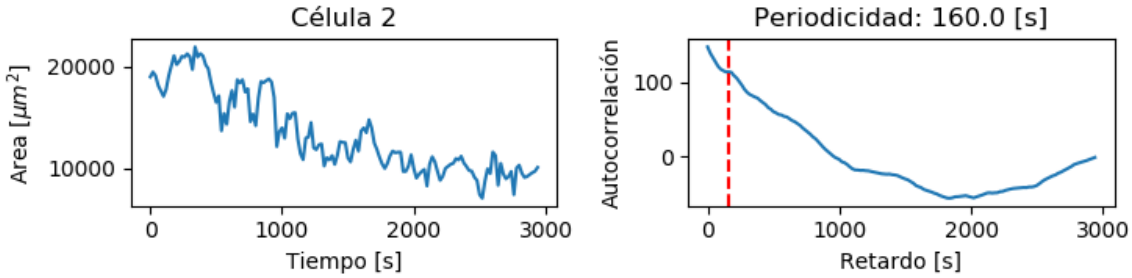


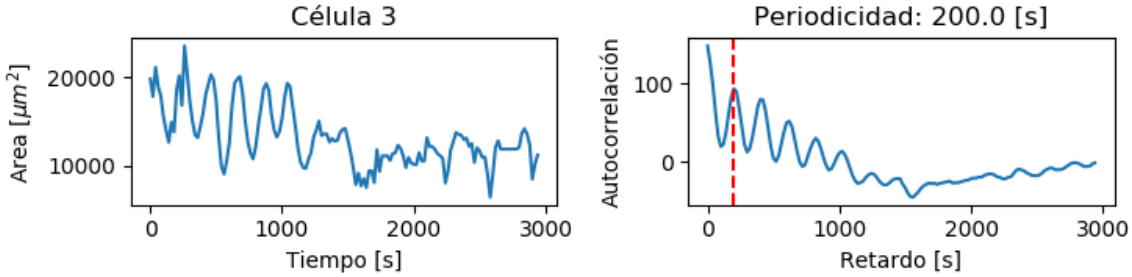
Figura 21. Cuadros 1, 76 y 149 del dataset B. La primera fila es la imagen original. La segunda fila corresponde a la segmentación semántica con la red neuronal. La tercera fila es la segmentación de instancia con la transformada watershed.



(a)



(b)



(c)

Figura 22. Series de área en el tiempo y autocorrelación de las células seleccionadas del Dataset B

Célula	Media [μm^2]	Desviación [μm^2]	Período [s]
1	18794.94	29.84	220.0
2	13359.80	54.34	160.0
3	13226.95	34.36	200.0
4	19146.56	21.71	240.0
5	8245.47	32.64	200.0
6	17341.73	24.42	240.0
7	8996.83	28.89	160.0
Promedio	14158.91	32.32	202.86 ± 31.04

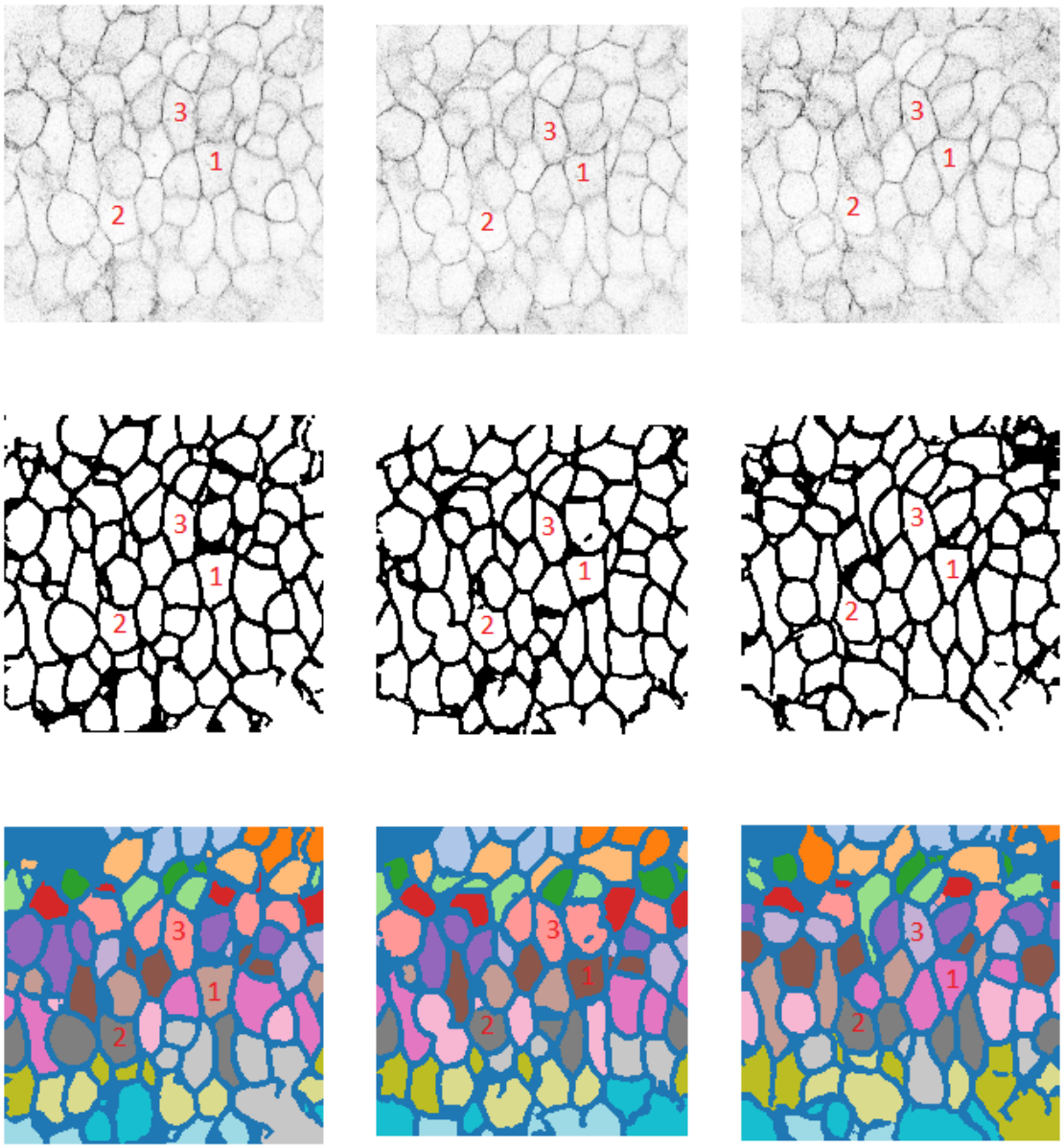
Tabla 2. Media, desviación estándar y período del área celular de siete células del dataset B donde el proceso de segmentación fue efectivo

4.2.2. Dataset A

Los resultados de la segmentación de las tres células escogidas en este dataset, se pueden apreciar en el la Figura 23. La primera fila muestra la imagen original, mientras que en la segunda fila se muestra la imagen segmentada semánticamente usando la red neuronal U-net, en donde se aprecia que de manera general el resultado de la segmentación es adecuado exceptuando algunas células de los bordes que son demasiado difusas. Por último en la tercera fila se muestra la imagen segmentada por instancia resultante de aplicar la transformada watershed. Al igual que en el dataset anterior se muestra en cada fila un paso distinto del proceso de segmentación y en cada columna un cuadro distinto de la secuencia, pudiendo así visualizar como varía la forma y tamaño de las células en el tiempo. Se destacan con números rojos tres células seleccionadas para estudiar sus series de tiempo de área celular en mayor detalle.

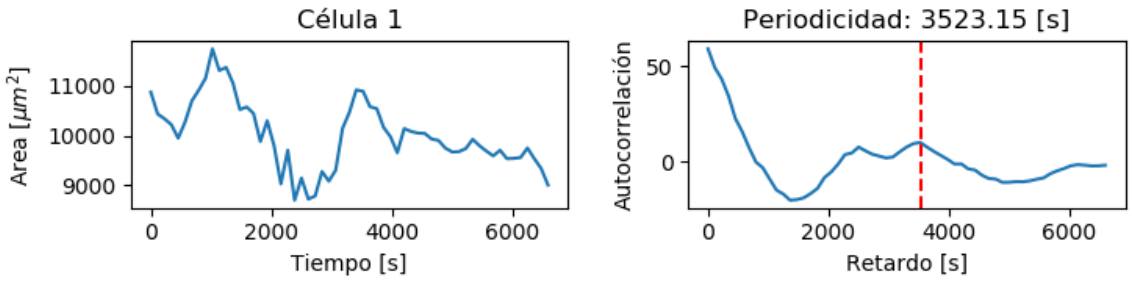
Las gráficas de las series de tiempo de área celular para las células seleccionadas, juntos con su función de autocorrelación se muestran en la Figura 24. A diferencia del dataset anterior, se puede observar que las periodicidades estimadas son mucho mas altas. Se observa también que las series de área celular no presentan un patrón oscilatorio claro mostrándose en general más ruidosas con respecto a las del dataset anterior.

Para un análisis cuantitativo más completo se repite el procedimiento anterior para un conjunto de diez células donde la segmentación fue efectiva y se reportan los períodos detectados a partir de la autocorrelación en la Tabla 3. El promedio del periodo estimado es de 4057.31 ± 1450.14 segundos, es decir considerablemente más alto y más disperso que el dataset anterior. Nuevamente no se aprecia una relación clara entre la periodicidad de la oscilación y el área de la célula.

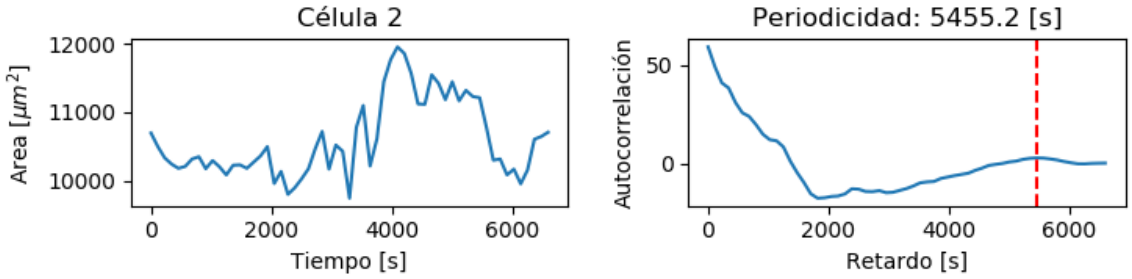


(a) (b) (c)

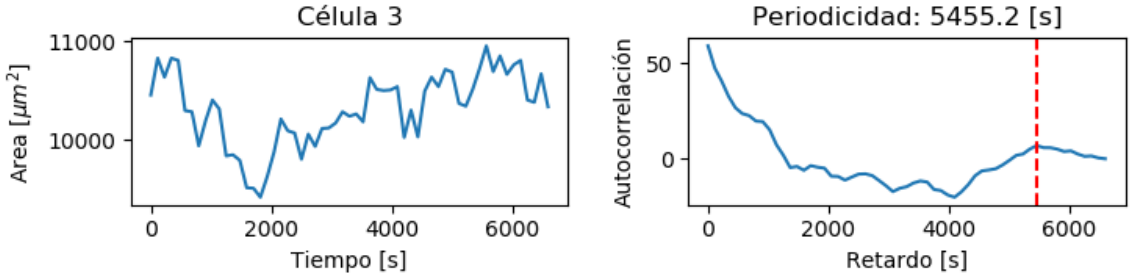
Figura 23. Frame 1, 31 y 60 de dataset A. La primera fila es la imagen original. La segunda fila corresponde a la segmentación semántica con la red neuronal. La tercera fila es la segmentación de instancia con la transformada watershed.



(a)



(b)



(c)

Figura 24. Series de área en el tiempo y autocorrelación de las células seleccionadas del Dataset A

Célula	Media [μm^2]	Desviación [μm^2]	Período [s]
1	10015.41	15.22	3523.15
2	10595.74	16.60	5455.2
3	10317.19	16.29	5455.2
4	8159.44	19.69	3523.15
5	10133.51	13.01	5114.25
6	11330.58	15.55	2045.7
7	7908.35	24.52	3977.75
8	8098.37	17.06	5909.8
9	13014.09	14.1	4318.7
10	8841.0	17.59	1250.15
Promedio	9841.37	16.96	4057.31 ± 1450.14

Tabla 3. Media, desviación estándar y período del área celular de diez células del dataset A donde el proceso de segmentación fue efectivo

4.2.3. Comparación con resultados previos

A modo de validación se contrastan los resultados obtenidos en cada dataset con los resultados presentados por Solon et al. 2009. En dicho trabajo se encontró una periodicidad media para las series de oscilación celular de 230 segundos con un rango de variación de 76 segundos[13]. Según esto podemos decir que la periodicidad media de 202 ± 31 segundos obtenida con el método propuesto sobre el dataset B es consistente con la literatura.

Por otro lado la periodicidad media de 4057,31 ± 1450,14 obtenida en el dataset A resulta ser considerablemente mayor a la reportada en la literatura y la obtenida en el dataset B. Adicionalmente las periodicidades detectadas son menos consistentes entre sí, lo cual se refleja por su mayor dispersión. Para descartar errores se inspeccionaron visualmente los resultados de segmentación de las células seleccionadas, pero no se encontraron anormalidades que expliquen esta situación.
Dado que el resultado de segmentación es correcto sospechamos que la resolución de la secuencia podría no ser suficiente para detectar sin ambigüedad las oscilaciones en el rango que se espera. Cabe destacar que las secuencias del dataset A tienen 60 cuadros y una período de muestreo de 113.65 segundos. La secuencia del dataset B en cambio tiene 149 cuadros y un período de muestreo de 20 s. Una hipótesis es que la frecuencia de muestreo del dataset A es demasiado baja para detectar sin ambigüedad las oscilaciones, que como vimos antes se encuentran en el rango de los 200 segundos.

[13] Figura 1 (D) en (Solon et al. 2009)

5. CONCLUSIONES Y TRABAJO FUTURO

5.1. Conclusiones

En este trabajo se implementó una programa para la estimación de periodicidad en series de tiempo en el área celular a partir de secuencias de imágenes registradas mediante la técnica de microscopía confocal. El elemento principal del programa es una red neuronal artificial de arquitectura U-net que realiza segmentación semántica, creando una máscara que identifica las paredes o membranas celulares. Sobre este resultado se utilizan técnicas de procesamiento de imágenes clásico y estadística para realizar segmentación de instancia y construir las secuencias de área celular.

Para probar el método se utilizan ambos datasets. En primer lugar el método propuesto se compara con un programa equivalente donde la segmentación semántica es realizada con técnicas clásicas (umbrales dinámicos), resultando en una máscara de segmentación con menor error. Luego las periodicidades estimadas se contrastan con resultados reportados en la literatura a modo de validación externa. Para el dataset B se obtienen resultados consistentes con la literatura validando así el método. Para el dataset A no se logra identificar las periodicidades en el rango esperado, posiblemente debido a la menor frecuencia de muestre de este dataset.

Consideramos que el método propuesto podría proveer un apoyo significativo a los expertos que actualmente realizan este tipo de procesos de segmentación de forma manual. El método realiza el flujo completo de procesamiento, es decir toma como entrada la secuencia de imágenes y retorna la serie de tiempo de área celular. Para los datasets considerados y usando hardware de sobremesa, el tiempo de procesamiento no supera los 10 minutos. En contraste, la segmentación manual de un dataset completo es un trabajo arduo que puede tomar semanas o meses. Una herramienta automática libera de esta pesada tarea al experto permitiéndole enfocarse en el análisis y abre la posibilidad de analizar datasets de mucho mayor volumen.

5.2. Trabajo futuro

El programa en su estado actual corresponde a un prototipo funcional. Un aspecto sumamente relevante para convertir el prototipo en un producto es la experiencia de usuario. El programa actualmente es una serie de rutinas en Python y cuadernillos Jupyter sin una interfaz de usuario unificada. Se propone como trabajo futuro el construir una interfaz gráfica con el objetivo de mejorar la experiencia de usuario y facilitar el acceso a científicos que no manejen lenguajes de programación. Para logra construir dicha capa front-end lo primero que se tendría que realizar sería una toma de requerimientos con los científicos que utilizarán el software, esto para cubrir todos los aspectos que ellos

necesitan. Luego se tendría que hacer un diseño del software iterando con el usuario para a futuro no tener que rehacer código o funcionalidades y de esta manera lograr un software más intuitivo. Dentro de las varias funcionalidades que se podrían implementar en esta intefaz, una de las mas importantes sería el que el usuario pueda escoger entre los distintos formatos de dataset a utilizar y así evitar tener rutinas diferentes para cada formato existente (CZI o TIFF en el caso de este trabajo).

Con respecto al modelo utilizado, cabe destacar que actualmente la segmentación de cada imagen se realiza de manera independiente. Esto desaprovecha la alta correlación que existe entre los cuadros ya sea en el eje temporal o de profundidad. Considerando la profundidad, el modelo podría aprovechar que en algunas capas las células en las imágenes se ven o aprecian de mejor manera que en otras. Esta información se podría utilizar para mejorar tanto la segmentación como el tracking. Al considerar el contexto temporal de las imágenes, se podría evitar los casos donde se pierde momentáneamente la célula y/o hacer mejores interpolaciones en las series de tiempo de área, lo que mejoraría la precisión del cálculo de la periodicidad.

El trabajo actual contempla la expansión y compresión de las células de forma global. Sin embargo se puede dar que una célula expanda solo una parte y contraiga otra, manteniendo su área. Por esto como trabajo futuro se podría plantear el modelar la célula como una elipse o envolvente convexa. Además con el uso de la profundidad y el tiempo se podría realizar un algoritmo para estimar la oscilación del volumen de una célula. Esto por lo investigado en el estado del arte no se ha realizado y sería una funcionalidad que podría dar paso a nuevas investigaciones y hallazgos por los científicos.

También es importante destacar que en este trabajo se utilizó una base de datos externa para entrenar los modelos (ISBI challenge 2012), la cual contaba con etiquetas. Luego se realizó un trabajo de adaptación de las imágenes a procesar, para que se parecieran a las etiquetadas y así el modelo funcionara de mejor forma. Si se contara con etiquetas para al menos un fragmento de los dataset de interés se podría realizar un procedimiento de *transfer learning* sobre el modelo pre-entrenado obteniendo muy posiblemente mejores desempeños.

Referencias

Abràmoff, M. D., Magalhães, P. J. & Ram, S. J. (2004). Image processing with ImageJ. *Biophotonics international*, *11*(7), 36-42.

Araya, C., Tawk, M., Girdler, G. C., Costa, M., Carmona-Fontaine, C. & Clarke, J. D. (2014). Mesoderm is required for coordinated cell movements within zebrafish neural plate in vivo. *Neural development*, *9*(1), 1-16.

Araya, C., Ward, L. C., Girdler, G. C. & Miranda, M. (2015). Coordinating cell and tissue behavior during zebrafish neural tube morphogenesis. *Developmental Dynamics*, *245*(3), 197-208.

Arganda-Carreras, I., Turaga, S. C., Berger, D. R., Cireşan, D., Giusti, A., Gambardella, L. M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J. M. Et al. (2015). Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, *9*, 142.

Benson, C., Rajamani, K. & Lajish, V. (2015). A review on automatic marker identification methods in watershed algorithms used for medical image segmentation. *IJISET-International J. Innov. Sci. Eng. Technol*, *2*(9).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Chen, J., Shao, H. & Hu, C. (2017). Image Segmentation Based on Mathematical Morphological Operator. *Colorimetry and Image Processing*.

Digabel, H. & Lantuéjoul, C. (1978). Iterative algorithms, En *Proc. 2nd European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*. Stuttgart, West Germany: Riederer Verlag.

Dumoulin, V. & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Elad, M. (2002). On the origin of the bilateral filter and ways to improve it. *IEEE Transactions on image processing*, *11*(10), 1141-1151.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning* [http : / / www . deeplearningbook.org]. MIT Press.

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2017). Mask r-cnn, En *Proceedings of the IEEE international conference on computer vision*.

Jiang, R., Bush, J. O. & Lidral, A. C. (2006). Development of the upper lip: morphogenetic and molecular mechanisms. *Developmental dynamics: an official publication of the American Association of Anatomists*, *235*(5), 1152-1166.

Ke, J., Jiang, Z., Liu, C., Bednarz, T., Sowmya, A. & Liang, X. (2019). Selective detection and segmentation of cervical cells, En *Proceedings of the 2019 11th International Conference on Bioinformatics and Biomedical Technology*.

Kornilov, A. S. & Safonov, I. V. (2018). An overview of watershed algorithm implementations in open source libraries. *Journal of Imaging*, *4*(10), 123.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, *1*(4), https://doi.org/10.1162/neco.1989.1.4.541, 541-551. https://doi.org/10.1162/neco.1989.1.4.541

Long, J., Shelhamer, E. & Darrell, T. (2015). Fully convolutional networks for semantic segmentation, En *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. Et al. (2019). Pytorch: An imperative style, high-performance deep learning library, En *Advances in neural information processing systems*.

Patin, F. (2003). An introduction to digital image processing. *online]: http://www. programmersheaven. com/articles/patin/ImageProc. pdf*.

Paulauskaite-Taraseviciene, A., Sutiene, K., Valotka, J., Raudonis, V. & Iesmantas, T. (2019). Deep learning-based detection of overlapping cells, En *Proceedings of the 2019 3rd International Conference on Advances in Artificial Intelligence*.

Punn, N. S. & Agarwal, S. (2020). Inception U-Net Architecture for Semantic Segmentation to Identify Nuclei in Microscopy Cell Images. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *16*(1), 1-15.

Roerdink, J. B. & Meijster, A. (2000). The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, *41*(1, 2), 187-228.

Ronneberger, O., Fischer, P. & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation, En *International Conference on Medical image computing and computer-assisted intervention*. Springer.

Sasai, Y. (2013). Next-generation regenerative medicine: organogenesis from stem cells in 3D culture. *Cell stem cell*, *12*(5), 520-530.

Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Solon, J., Kaya-Copur, A., Colombelli, J. & Brunner, D. (2009). Pulsed forces timed by a ratchet-like mechanism drive directed tissue movement during dorsal closure. *Cell*, *137*(7), 1331-1342.

Subiabre, J. (2013). *Dynamic-weighted threshold: Algoritmo de segmentación para imágenes biológicas curvas y de alta densidad celular*. Universidad Austral de Chile.

Sultana, F., Sufian, A. & Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 106062.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going Deeper With Convolutions, En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tychsen-Smith, L. & Petersson, L. (2018). Improving object localization with fitness nms and bounded iou loss, En *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Vincent, L. & Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6), 583-598.

Xie, W., Noble, J. A. & Zisserman, A. (2018). Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, *6*(3), 283-292.

Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. (2014). How transferable are features in deep neural networks?

Zeiler, M. D. & Fergus, R. (2014). Visualizing and understanding convolutional networks, En *European conference on computer vision*. Springer.

Zhang, H. & Ma, J. (2020). Hartley Spectral Pooling for Deep Learning. *CSIAM Transactions on Applied Mathematics*, *1*(3), 518-529. https://doi.org/https://doi.org/10.4208/csiam-am.2020-0018

Zou, F., Shen, L., Jie, Z., Zhang, W. & Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop, En *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.