UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE MATEMÁTICAS

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

Tesis de maestría

Res2Unet: Una Red Completamente Convolucional para la Segmentación del Parásito *Trypanosoma cruzi*

Allan Eduardo Ojeda Pat

Asesora

Anabel Martin González

Mérida, Yucatán, México

Junio, 2020

Dedicado a mi familia y amigos por el apoyo brindado durante estos dos años.

Agradecimientos

Agradezco a mis maestros y asesores que me apoyaron con el conocimiento, motivación y experiencias para la realización de mi proyecto de tesis. Agradezco también al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado para cursar el estudio de maestría.

Resumen

La enfermedad de Chagas es un problema de salud pública, que causa miles de muertes por año atribuidas a problemas cardíacos, digestivos, neurológicos o mixtos, y debido a que la mayoría de las personas infectadas no presentan síntomas, se considera una enfermedad potencialmente mortal. Un análisis de sangre resulta ser el método preferido para generar un diagnóstico de la enfermedad; sin embargo, es un proceso tardado, ya que requiere de mucho esfuerzo por parte de expertos para analizar grandes cantidades de muestras en búsqueda de parásitos. La implementación de sistemas automáticos que faciliten la segmentación del parásito en imágenes de muestras de sangre capturadas por microscopio puede ser de gran utilidad para ahorrar tiempo y esfuerzo. Por lo tanto, en este trabajo de tesis desarrollamos el modelo de segmentación semántica Res2Unet basado en aprendizaje profundo para facilitar la visualización de la morfología del parásito *Trypanosoma cruzi* en imágenes digitales de muestras de sangre. Comparamos el desempeño de segmentación obtenido con nuestra propuesta contra los modelos U-Net, ResUnet, y los clasificadores basados en aprendizaje automático Gauss y máquinas de soporte vectorial (SVM).

Índice

Lista de figuras

Lista de tablas

1. Introducción

El mal de Chagas o Tripanosomiasis americana es una enfermedad potencialmente mortal para la población mundial ocasionada por el parásito protozoario *Trypanosoma cruzi* [1]. El principal medio de contagio es por contacto con insectos infectados de la subfamilia *Triatominae*. Estos insectos habitan principalmente en regiones rurales de Latinoamérica y el humano usualmente contrae la enfermedad cuando las heces de los insectos hacen contacto directo con alguna lesión en la piel. En la etapa temprana de la enfermedad un gran número de parásitos circulan en la sangre. En la etapa tardía, los parásitos se alojan y reproducen en tejido muscular del corazón y demás órganos, pudiendo generar complicaciones que causen la muerte. Para el caso del Mal de Chagas, la identificación del parásito *Trypanosoma cruzi* en una muestra de sangre conlleva a la evidencia certera y potencial desarrollo de la enfermedad en humanos [2]. El método más conveniente para el diagnóstico de la enfermedad es mediante el análisis de frotis de sangre y es importante realizarlo en la etapa temprana para un diagnóstico eficaz [1,3]. Sin embargo, es un método tardado que requiere mucho tiempo y esfuerzo, ya que involucra el análisis de grandes volúmenes de muestras.

Debido a las altas implicaciones que puede ocasionar al portador del parásito *Trypanosoma cruzi*, diversos estudios basados en aprendizaje automático se han propuesto como alternativa de apoyo para los examinadores [4-7]. Sin embargo, estos trabajos aún presentan inconvenientes en la detección y/o segmentación del parásito *Trypanosoma cruzi*.

Este trabajo se desarrolló en una subárea del aprendizaje automático conocido como aprendizaje profundo. Proponemos la segmentación del parásito *Trypanosoma cruzi* en imágenes de muestra de sangre capturadas con microscopio. Una imagen de muestra de sangre contiene distintos elementos como células sanguíneas, y en caso de contagio contiene los parásitos y otros elementos teñidos de forma similar. La segmentación de una imagen consiste en separar los objetos principales asignándole una clase en particular, para diferenciarlos con otros objetos presentes de no interés [8]. Para el caso de un paciente con la enfermedad de Chagas, la segmentación conservaría los elementos característicos de un parásito y eliminaría los demás elementos como células. Con la segmentación del parásito *Trypanosoma cruzi* un técnico examinador podrá corroborar con facilidad su presencia en las imágenes al eliminar elementos no importantes, obteniendo información relevante de su morfología para un uso posterior. Para tal tarea, implementamos una propuesta de Red Neuronal Completamente Convolucional (FCNN por sus siglas en inglés), a la cual nombramos como Res2Unet para la segmentación de los parásitos *Trypanosoma cruzi* en imágenes de muestras de sangre.

1.1. Estado del arte

Diferentes métodos se han reportado para el diagnóstico de la enfermedad de Chagas recientemente. Podemos dividir estos métodos en tres categorías: métodos clínicos, métodos basados en aprendizaje automático, y métodos de aprendizaje profundo.

1.1.1. Métodos clínicos

Un análisis de sangre es esencial para detectar y tratar a tiempo la enfermedad de Chagas. Existen otros métodos que requieren del uso de pruebas portátiles especializadas para la detección del parásito *T. cruzi* en una muestra de sangre. Uno de ellos es el denominado *Chagas Stat-Pak*, presentado por Ponce et al. [9], cuyo método tiene un 99.6% de sensibilidad y 99.9% de especificidad; este rendimiento es comparable al que se obtiene con una prueba de ELISA. Otro método que requiere de una prueba portátil es el denominado *Chagas Detect Plus* [10]. Si bien, estas pruebas portátiles para diagnosticar la enfermedad de Chagas son rápidas y eficaces, obligatoriamente se requiere algún método manual o serológico para emitir un diagnóstico preciso. En caso de un resultado positivo, el paciente debe iniciar el tratamiento correspondiente [10].

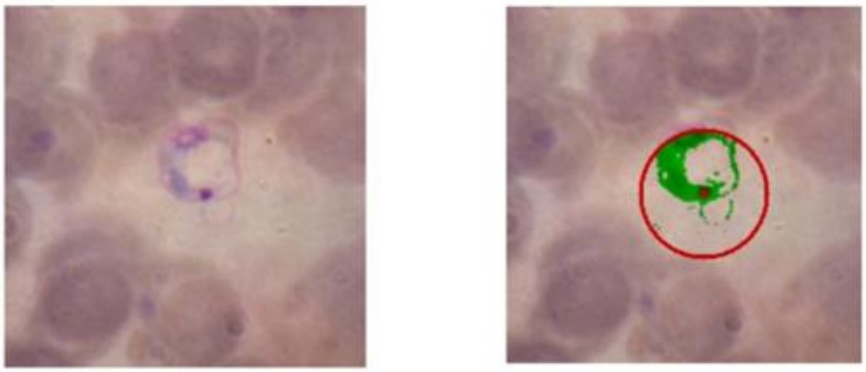
1.1.2. Métodos basados en aprendizaje automático

Detección del parásito T. cruzi.

Uc-Cetina et al. [4] proponen la detección del parásito de Chagas en imágenes de muestras de sangre mediante un algoritmo basado en el análisis del discriminante

Gaussiano. Su método consiste en analizar la información presente en el canal verde de las imágenes para extraer un vector con 121 características relevantes en zonas de la imagen donde existan posibles núcleos de parásitos. El problema de clasificación consiste en distinguir si el vector de características corresponde a un parásito de Chagas o no. Para ello, los autores implementaron el análisis de discriminante de Gauss para la construcción de dos modelos: $p(x|y=1)$, que modela la distribución de características de lo que es parecido a un parásito, y $p(x|y=0)$, que modela la distribución de características de lo que no parece un parásito; donde y indica si un ejemplo es un parásito ($y=1$) o no ($y=0$). En la clasificación de un nuevo conjunto de pixeles de entrada se calculan las probabilidades del modelo Bayesiano, y la mayor probabilidad indicará la clase dominante de la muestra. Como paso final del algoritmo, los autores implementaron un algoritmo de búsqueda para encontrar puntos negros en las imágenes, los cuales indican la presencia de posibles núcleos de parásitos. Por cada imagen de entrada, el algoritmo puede clasificar hasta 100 posibles candidatos de núcleos. Su base de datos cuenta con 120 imágenes, de las cuales, 60 cuentan con presencia de parásitos. Los autores propusieron tres distintos modelos de experimentos, pero el que mejor obtuvo resultados fue el experimento número dos que aplicó el análisis de componentes principales (PCA) [11], para reducir el tamaño del vector de características de entrada. Las tasas de rendimiento final que reportaron son: falsos negativos 0.0167, falsos positivos 0.1563, verdaderos negativos 0.8437 y verdaderos positivos 0.9833. En la Fig. 1.1 se observa un ejemplo de la detección del parásito con el algoritmo de los autores. Sin embargo, una de las desventajas de su propuesta radica en el algoritmo de búsqueda. Si una imagen de entrada

contiene demasiadas zonas de pixeles con una coloración similar al ADN del parásito, se realizarán búsquedas innecesarias en la imagen. De lo anterior es importante mencionar que dependiendo de la experiencia del técnico para teñir el frotis de sangre es muy probable que existan zonas de tinta en las imágenes que sean similares al tono del ADN del parásito, por lo que esta propuesta de algoritmo podría fallar.



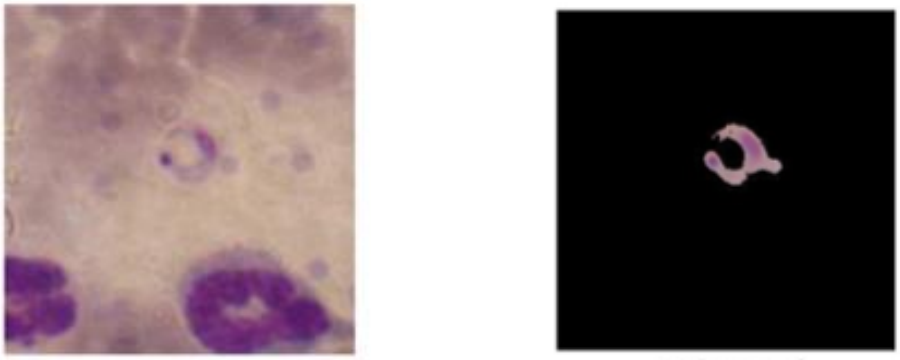
a) b)

Fig. 1.1. Detección del parásito *T. cruzi*. a) ejemplo de parásito; b) resultado de detección. Fotografías tomadas de Uc-Cetina et al. [4].

En el trabajo presentado por Soberanis-Mukul et al. [5], se propone un algoritmo automatizado para detectar el parásito usando diversas técnicas de segmentación y clasificación en imágenes de muestras de sangre. Su base de datos cuenta con 120 imágenes, de las cuales 60 tienen presencia de parásitos. En una primera etapa de preprocesamiento, aplican una máscara con el fin de conservar información presente del cuerpo del parásito. La máscara es el resultado de una umbralización sobre la imagen que contiene la diferencia de los canales azul y verde, previamente calculado. Posteriormente, implementan un algoritmo de etiquetado para separar las distintas regiones de pixeles y contabilizar sus áreas,

esto con el fin de conservar sólo aquellas regiones cuyas áreas sean similares al de un posible parásito. Como resultado de esta etapa de preprocesamiento, se calculan centroides en las imágenes a color en las ubicaciones de las regiones que se mantuvieron. La etapa de segmentación consiste en separar el objeto de interés (parásito) del fondo. Para ello, un clasificador Gaussiano se entrena con ejemplos positivos (lo que representa un objeto de interés) y negativos (lo que no representa un objeto de interés). Calculando las probabilidades que se presentan en el modelo de decisión Bayesiano, se obtienen regiones de pixeles de interés que pueden ser un parásito. Posteriormente, los autores realizan una intersección de las imágenes obtenidas en ambas etapas (preprocesamiento y segmentación) para conservar las regiones donde el área obtenida es clasificada como un parásito si, y sólo si, también forma parte de la región de interés del resultado del método Gaussiano. En una última etapa, las imágenes que se mantuvieron en la intersección pasan por un clasificador de *K* vecinos más cercanos [12], previamente entrenado, para realizar una clasificación binaria (si es parásito o no es parásito). Después del entrenamiento y pruebas, los autores reportaron 98% de sensibilidad y 85% de especificidad. En la Fig. 1.2 se muestra un ejemplo de la segmentación del parásito de Chagas con el algoritmo de los autores. La desventaja de esta propuesta radica principalmente en el algoritmo de etiquetado, ya que, si los candidatos de parásitos no sobrepasan un valor de umbral establecido para ser considerados como parásitos, estos no se clasificarán. Viceversa, si existen candidatos falsos con tamaños similares a un parásito se realizarán clasificaciones innecesarias. Por lo anterior, al no existir un tamaño promedio de un parásito, y dado que la mayoría de los parásitos pudieran estar

semi ocultos entre células sanguíneas, un método con un umbral no es eficaz en una propuesta de detección.

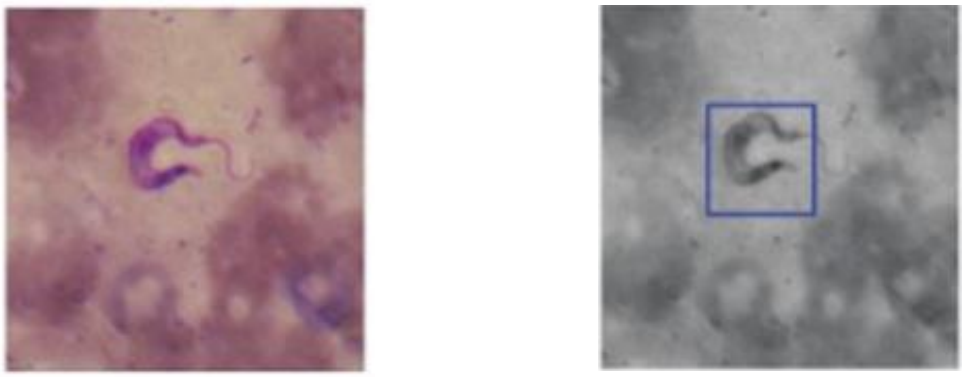


a) b)

Fig. 1.2. Detección del parásito *T. cruzi*. a) ejemplo de parásito; b) resultado de segmentación. Fotografías tomadas de Soberanis-Mukul et al. [5].

En el trabajo de Uc-Cetina et al. [6], se realiza una comparación de dos algoritmos robustos, AdaBoost y máquinas de soporte vectorial (Support Vector Machine, SVM), para la tarea de detección del parásito de *T. cruzi* en imágenes de muestras de sangre. El proceso de detección del parásito implementando AdaBoost consiste en cuatro etapas: 1) adquisición de las imágenes, 2) preprocesamiento de las imágenes, 3) detección de posibles parásitos entrenando un clasificador binario AdaBoost alimentado con características Haar específicas, y 4) un post procesamiento usando una técnica enfocada en la acumulación de ADN entrenando una máquina de soporte vectorial para descartar falsos positivos. Es importante mencionar que las plantillas Haar [13] propuestas en Uc-Cetina et al. [6] están diseñadas para representar la morfología común del parásito de *T. cruzi*. Su base de datos cuenta con 120 imágenes, de las cuales 60 tienen presencia de parásitos. Los autores realizaron una comparación del proceso de detección antes

mencionado (AdaBoost + SVM) con una implementación para reconocer el parásito de Chagas usando un único clasificador SVM. El clasificador SVM [14] se entrenó con diversas características propuestas en Ross et al. [15]. Los diversos experimentos que se realizaron fueron con los clasificadores propuestos: 1) AdaBoost, 2) AdaBoost + postprocesamiento, 3) SVM con kernel lineal, 4) SVM con kernel polinomial, y 5) SVM con kernel RBF. La clasificación usando AdaBoost más un post procesamiento, obtuvo los mejores resultados, reportando 100% de sensibilidad y 93.25% de especificidad. En la Fig. 1.3 se muestra un ejemplo de la detección de *T. cruzi* con el algoritmo de los autores. La desventaja principal se encuentra en el clasificador SVM que recibe información de los puntos negros (dark spots) en las imágenes para determinar si es un parásito. Como en una determinada zona de la imagen puede haber demasiados pixeles de color similar al parásito *T. cruzi*, resultaría difícil poder descartar falsos positivos.



a) b)

Fig. 1.3. Detección del parásito *T. cruzi*. a) ejemplo de parásito; b) resultado de detección. Fotografías tomadas de Uc-Cetina et al. [6].

Segmentación del parásito *T. cruzi*

Soberanis-Mukul [7] propone una comparación de tres clasificadores (SVM, AdaBoost y redes neuronales artificiales) para detectar y segmentar el parásito *T. cruzi*. Su metodología consiste en tres etapas: 1) cómputo de súper pixeles, 2) extracción de las características óptimas, y 3) entrenamiento de cada clasificador: redes neuronales usando el algoritmo de retro propagación; AdaBoost usando ensamble de perceptrones; y máquinas de soporte vectorial. Los súper pixeles están formados por conjuntos de pixeles que describen regiones continúas delimitadas a lo largo de imagen. La propuesta de usar grupos de pixeles permite que se extraigan características considerando el vecindario y la relación de los pixeles dentro de un grupo [7]. Su base de datos empleada consiste en 900 imágenes de ejemplos positivos (con presencia de parásitos) y 900 imágenes de ejemplos negativos (sin presencia de parásitos). Los vectores de características de entrada para los clasificadores contienen distintos valores con base en diferentes espacios de color de los súper pixeles. En una primera etapa de la experimentación, se escoge la configuración del vector de características que mejor desempeño obtenga en los tres clasificadores propuestos por el autor. Posteriormente, compara el desempeño de sus clasificadores contra tres algoritmos en el estado del arte: clasificador Gaussiano, clasificador de Bayes y el algoritmo propuesto en [16]. Al finalizar la experimentación, el autor reporta que el clasificador Gaussiano presentó el menor error cuadrático medio con 0.18568, seguido del clasificador SVM propuesto por el autor (con un tamaño de súper pixel de 100) que obtuvo 0.22635, y por último, el clasificador usando redes neuronales

con un error de 0.361. En la Fig. 1.4, se muestra un ejemplo de los resultados de clasificación para una imagen de entrada. El autor concluye que la técnica de súper pixeles permite adaptarse a los contornos del contenido de las imágenes, lo cual aísla de forma parcial o total el objeto de interés del fondo durante la clasificación. Podemos resaltar dos desventajas: (1) El cómputo de los super pixeles es una tarea de preprocesamiento que consume mucho tiempo, y (2) antes de la imagen segmentada de salida se aplica un método de umbral para conservar solo las regiones de pixeles que cumplan con un valor de área predefinido, por lo que el valor del umbral es manualmente establecido por cada imagen y no funcionaría en la práctica (dado que los parásitos tienen diferentes tamaños).

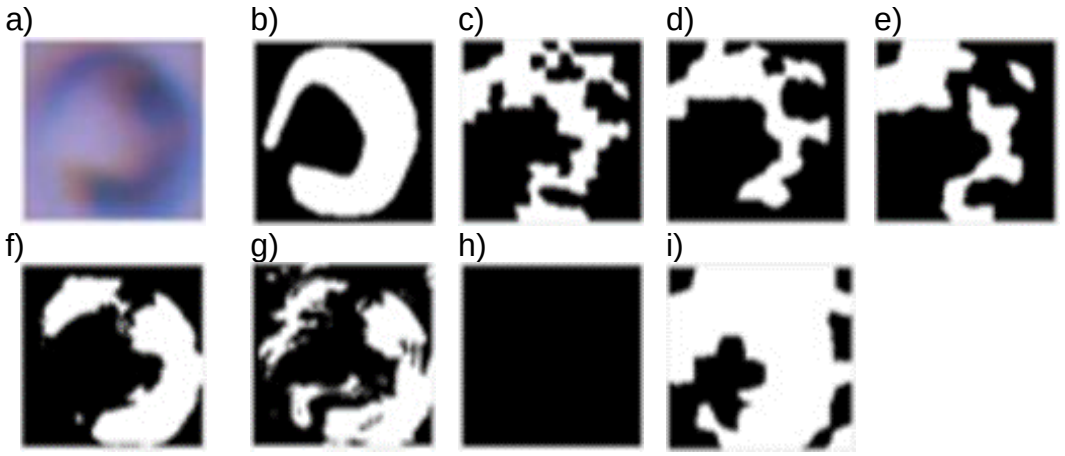


Fig. 1.4. Ejemplo de la segmentación. a) imagen original; b) segmentación manual; c), d) y e) segmentaciones con SVM y súper pixeles de 50, 100 y 150, respectivamente; f) clasificación Gaussiana; g) clasificación Bayesiana; h) clasificación con método de [16]; e i) clasificación con redes neuronales. Fotografías tomadas de Soberanis-Mukul [7].

1.1.3. Métodos basados en aprendizaje profundo

En 2020, Ojeda-Pat et al. [17] implementaron el modelo U-Net [18] para la segmentación binaria automática del parásito *T. cruzi* en imágenes de muestras de sangre. Su base de datos consiste en 1000 imágenes de tamaño 512 × 512 en formato RGB de las cuales 940 imágenes contienen presencia de parásitos y 60 no contienen parásitos. Los autores generaron manualmente el conjunto de imágenes del ground truth correspondiente. El modelo U-Net devuelve una imagen segmentada de resolución completa a través de la aplicación de zero-padding en cada operación de convolución y genera el mapa de segmentación usando una capa de convolución 1 × 1 (con un filtro) al final de la red. Debido a que en cada imagen de tamaño 512 × 512 aproximadamente solo el 1.8% de los pixeles corresponde a pixeles de la clase parásito, los autores implementaron la función de costo Weighted Binary Cross Entropy (WBCE) para mitigar el desbalance de clases y así evitar que la red solo devuelva predicciones de la clase de fondo. Los autores compararon los resultados obtenidos con otro entrenamiento de la U-Net usando la función de costo Binary Cross Entropy (BCE). Ambos modelos fueron entrenados con el optimizador de ADAM, un tamaño de lote (batch size) de 2 y una tasa de aprendizaje de 1E-4. Al finalizar la experimentación, los autores reportaron que el modelo entrenado con la función de costo WBCE obtuvo el mayor puntaje F2 con un valor de 0.80, un valor del Coeficiente de Dice de 0.68, un valor de Recall de 0.87 y un valor de Precisión de 0.63. En la Fig.1.5 se muestra un ejemplo de la segmentación de una imagen de entrada con los modelos entrenados. Una desventaja del trabajo reportado en [17], son los valores

bajos de las métricas del Coeficiente de Dice y de Precisión. Un bajo valor de Precisión refleja que las segmentaciones de los parásitos incluyen demasiado ruido o pixeles clasificados incorrectamente como parásito, lo que se refleja visualmente en menos detalle de los bordes del parásito segmentado.

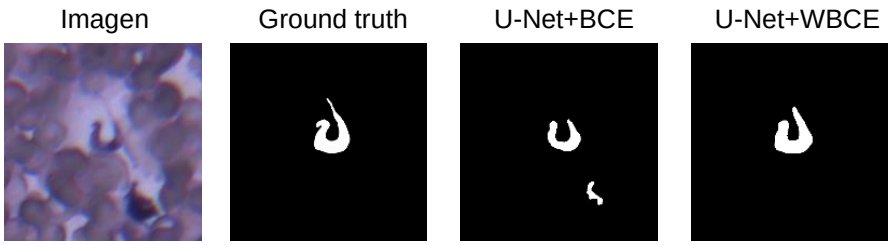


Fig. 1.5. Ejemplo de la segmentación con el modelo U-Net.

1.2. Formulación del problema

El análisis de frotis de sangre es una técnica frecuente y conveniente para el diagnóstico de la enfermedad de Chagas, pero también es un trabajo visual laborioso que implica tiempo y esfuerzo por parte de los técnicos de laboratorio entrenados, dado que involucra el análisis de muchas muestras de sangre.

Un sistema automático que ayude en el proceso de analizar podría reemplazar o asistir al frotis de sangre, eliminando o reduciendo el error humano, especialmente, durante la fase aguda de la enfermedad y en zonas altamente endémicas.

Aunque la enfermedad de Chagas es potencialmente mortal para la población mundial, se han reportado un número limitado de trabajos para la detección y segmentación automática del parásito *Trypanosoma cruzi* en imágenes de

muestras de sangre [4-7]. La mayoría de estos trabajos se entrenan con características que los expertos proponen después de analizar sus imágenes. Si bien, resulta conveniente definir qué técnica utilizar para la extracción de características, las redes neuronales completamente convolucionales (FCNN) aprenden a extraer automáticamente las características más importantes obteniendo mejores resultados de clasificación [8, 19-20].

1.3. Objetivos

1.3.1. Objetivo general

El objetivo general de este trabajo de tesis es desarrollar una red completamente convolucional basada en el modelo U-Net y el aprendizaje residual para la segmentación automática del parásito *Trypanosoma cruzi* en imágenes de frotis de sangre.

1.3.2. Objetivos específicos

- Generar una base de datos de imágenes para el entrenamiento del modelo.
- Implementar una arquitectura de FCNN (modelo Res2Unet) basada en la U-Net y el aprendizaje residual.
- Entrenar y evaluar el modelo Res2Unet con diversas funciones de costo e hiper parámetros.
- Comparar los resultados obtenidos con métodos del estado del arte.

1.4. Publicaciones

Los trabajos publicados con base a la investigación realizada de esta tesis son los siguientes:

1. Ojeda-Pat, A., Martin-Gonzalez, A., & Soberanis-Mukul, R. (2020). Convolutional Neural Network U-Net for *Trypanosoma cruzi* Segmentation. In book: Intelligent Computing Systems, pp.118-131. doi: 10.1007/978-3-030-43364-2_11
2. Ojeda-Pat, A., Martín-González, A., & Uc-Cetina, V. (2020). Revisión de métodos de aprendizaje automático para detectar al parásito de la enfermedad de Chagas. Investigación y Ciencia de la Universidad Autónoma de Aguascalientes, 28(80), xx-xx. (en proceso de publicación).

2. Marco teórico

En este capítulo se aborda una descripción completa de la enfermedad de Chagas, y se presentan los conocimientos previos necesarios para el desarrollo del trabajo realizado.

2.1. Enfermedad de Chagas

El mal de Chagas o Tripanosomiasis americana es una infección crónica ocasionada por el parasito protozoario Trypanosoma cruzi [1], el cuál fue originalmente descubierto en 1909 por el físico brasileño Carlos Chagas [21].

El parásito *T. cruzi* (Fig. 2.1) pertenece a la familia Trypanosomatidae y habita en la sangre de algunos mamíferos. El parásito se compone de un flagelo para moverse, un cineoplasto y un núcleo vesiculoso en la parte central de su cuerpo [22]. El cineosplasto contiene tres espirales de ADN que se pueden identificar de forma sencilla por un tono obscuro.

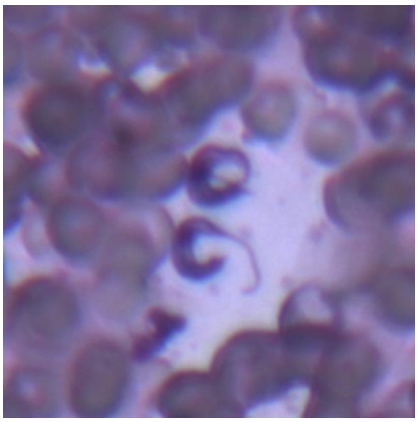


Fig. 2.1. Parásito *Trypanasoma cruzi* en una fotografía digital de microscopio.

La infección es mayormente transmitida a humanos través del contacto con insectos triatominos infectados. Los insectos triatominos (ver Fig. 2.2) también conocidos como chinches besuconas o insectos pic en Yucatán, viven principalmente en áreas rurales de Latino América. Los insectos triatominos contraen el parásito al alimentarse de la sangre de mamíferos. Las personas usualmente se infectan si las heces de las chinches besuconas entran por alguna membrana mucosa o piel herida, justo después de frotar el área de la picadura [24]. A esta forma de transmisión por medio de un agente externo se le conoce como transmisión por vector. Algunos otros modos de transmisión no vectorial son por vía congénita (de madre a bebe), por trasplante de órgano o transfusión de sangre [1]. También es posible infectarse por medio de alimentos contaminados.

Se estima que de 6 a 8 millones de personas están infectadas con el mal de Chagas en el mundo, principalmente en México, América Central y Sudamérica donde es endémica [1, 25], con una tasa promedio de 10 mil muertes al año atribuidas al mal de Chagas [4, 26].

Fig. 2.2. Insecto triatomino principal transmisor del parásito *T. cruzi*. Fotografía tomada por Guhl [23].

La mayoría de las personas infectadas no saben que tienen la enfermedad de Chagas, y arriba del 20-30% de los infectados desarrollará una condición médica tal como enfermedades del corazón, los cuales pueden ocasionar una muerte repentina, y arriba del 10% sufrirá de problemas digestivos, neurológicos o mixtos [1, 24]. La enfermedad de Chagas consiste en dos fases: la fase aguda, fase indeterminada y la fase crónica. Todas las fases pueden ser libres de síntomas o puede que pongan en riesgo la vida años después de la infección. La fase aguda ocurre en las primeras semanas o meses después de la infección, y durante esta fase un alto número de parásitos se encuentran circulando en la sangre del huésped. Por lo general la fase aguda es asintomática y algunos pueden desarrollar fiebre, dolor de cabeza, escalofríos, pérdida del apetito, entre otros [1]. También es posible que exista una inflamación en la zona donde el insecto se alimentó, y puede que persista por un tiempo en la fase aguda. Después de la fase aguda, sigue una fase indeterminada o fase de latencia. En esta fase, el portador del parásito no presenta síntomas y puede mantenerse así por varios años, e incluso de por vida [1]. A continuación, le sigue la fase crónica. La fase crónica

puede aparecer años o décadas después de la infección inicial. Durante esta fase, los parásitos están principalmente alojados en el corazón y músculos digestivos, dañando el tejido muscular [1]. La enfermedad de Chagas en el corazón suele ser mortal debido a cambios en el ritmo cardiaco y funcionamiento del corazón [1]. En esta última etapa, es casi imposible localizar un parásito en una muestra común de sangre usando un microscopio [1].

El diagnóstico de la enfermedad dependerá primordialmente de la etapa de esta. Un diagnóstico certero consiste en confirmar la presencia del parásito *T. cruzi* [2].

Es de suma importancia realizar los estudios de detección durante la fase inicial, dado que un gran número de parásitos se encuentran circulando en la sangre, para obtener un diagnóstico eficaz y un pronto tratamiento de la enfermedad [1]. El método más conveniente para diagnosticar esta enfermedad es mediante el análisis de frotis sanguíneo [3]. Un frotis sanguíneo es una técnica que consiste en colocar una gota de sangre sobre un portaobjetos. Posteriormente, la muestra se tiñe para resaltar las propiedades del parásito y después analizarla bajo el microscopio. Para la fase crónica, uno de los métodos para diagnosticar la enfermedad de Chagas es a través de la prueba de ELISA [3, 24], la cual trata de una prueba de laboratorio para detectar anticuerpos en la sangre. Es importante resaltar que la efectividad del tratamiento de la enfermedad, una vez ya diagnosticada, es cerca del 80% para la fase aguda. Para la fase crónica, el tratamiento se enfocará directamente en tratar los padecimientos ocasionados por la enfermedad, pero no curará la enfermedad de Chagas [1].

La detección de parásitos a través de inspección microscópica de frotis de sangre es una técnica muy común y conveniente, pero en ocasiones es un proceso que requiere de mucho tiempo y esfuerzo, ya que involucra el análisis de muchas muestras de sangre [27-30].

2.2. Procesamiento de imágenes

En el mundo de la computación, la visión por computadora es un campo altamente estudiado debido a los grandes beneficios que otorga. Diferente a las computadoras, los humanos pueden realizar tareas de identificación y localización intuitivamente. El objetivo de la visión por computadora es otorgar las mismas habilidades a los sistemas inteligentes para poder replicar las habilidades del sistema visual humano [8, 31]. Con los avances en la tecnología, los métodos de visión por computadora se han usado hoy en una amplia variedad de aplicaciones, como en aplicaciones médicas, y diferentes problemas del mundo real que pueden mejorarse con un enfoque en visión computacional [8, 32].

Para trabajar con la segmentación del parásito *T. cruzi* necesitamos los fundamentos primordiales del uso de las imágenes digitales, las herramientas o metodologías usadas en visión por computadora para la segmentación en aprendizaje automático y aprendizaje profundo. En las siguientes subsecciones se abordará este contenido.

2.2.1. Análisis y procesamiento de imágenes

El procesamiento y análisis de imágenes es aplicado en la descripción y segmentación de imágenes para las aplicaciones de visión por computadora [33]. Para realizar la segmentación de una imagen es necesario trabajar con la información que está contenida en ella en busca de patrones u operadores que permitan separar los elementos de interés de la imagen.

2.2.2. Imagen digital

Una imagen se define como una función de dos dimensiones $I(x, y)$donde x y y son las coordenadas de un plano conteniendo todos sus puntos, y $I(x, y)$ es la intensidad o nivel de gris de la imagen en el punto (x, y). Si los valores de intensidad de la función $I()$ y las coordenadas x y y son discretos, se trata de una imagen digital [33].

Una imagen digital está compuesta de un número finito de elementos, donde cada elemento tiene una localidad y un valor en particular. Estos elementos son conocidos como pixeles. El dominio de la función $I(x, y)$ está dado por las dimensiones de la imagen en valores enteros y el punto inicial de la imagen está ubicado en la esquina superior izquierda de una imagen en la posición 0,0. La representación matemática de una imagen como una matriz A de dimensiones *m* × *n* de la siguiente forma:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,0} & a_{m,1} & \cdots & a_{m-1,n-1} \end{bmatrix} \quad (2.1)$$

donde n es la cantidad de columnas de pixeles (ancho), m la cantidad de filas de pixeles (alto), y a es un píxel de la imagen.

Una imagen en escala de grises representa un arreglo bidimensional o matriz de valores. Una imagen en escala de grises de 8 bits, cada píxel a se representa con un número entero positivo como un valor de intensidad de 0 a 255 [33]. Un valor de intensidad de 0 representa el color negro, y un valor de intensidad de 255 un color blanco. Los valores intermedios representarían una tonalidad o intensidad de gris creciente hasta alcanzar el máximo valor de 255. En la Fig. 2.3 se muestra un ejemplo de una imagen en escala de grises.

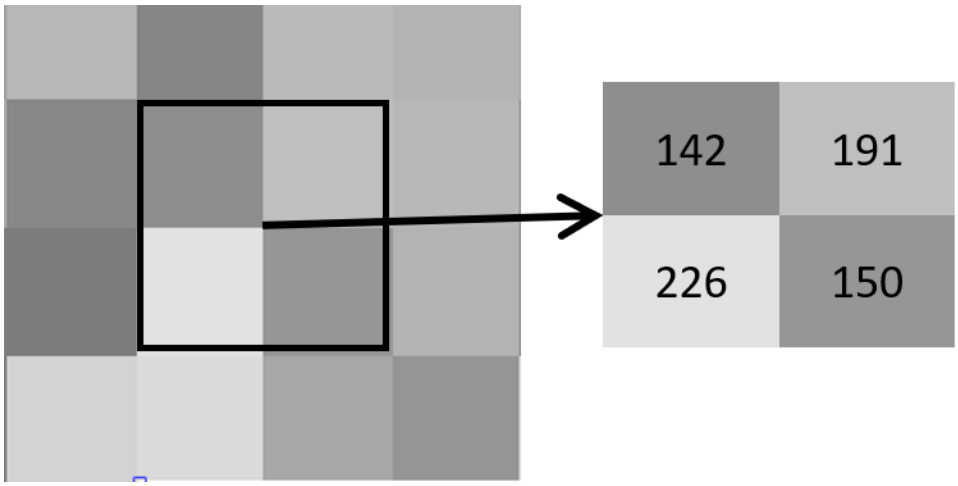


Fig. 2.3. Imagen en escala de grises.

Una imagen digital tiene una resolución fija. La resolución es la cantidad de pixeles contenidos en la imagen, normalmente expresado como *m* × *n*. La calidad de una imagen está relacionada directamente con su resolución [33].

2.2.3. Imagen a color

Una imagen digital a color puede ser definido de forma similar, con la excepción de que cada elemento o píxel es descrito y codificado de forma distinta de acuerdo con el espacio de color utilizado [33]. Por ejemplo, el espacio de color más utilizado es el RGB (color verdadero) [33], donde cada píxel P se representa como un vector con tres valores de intensidad de los canales de color Rojo (R), Verde (G) y Azul (B) de la siguiente forma:

$$P = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.2}$$

Esto origina un color a partir de tres valores de intensidad por cada píxel en una imagen RGB (Fig. 2.4). Cada canal de color puede tener un rango de intensidad de 0 a 255.

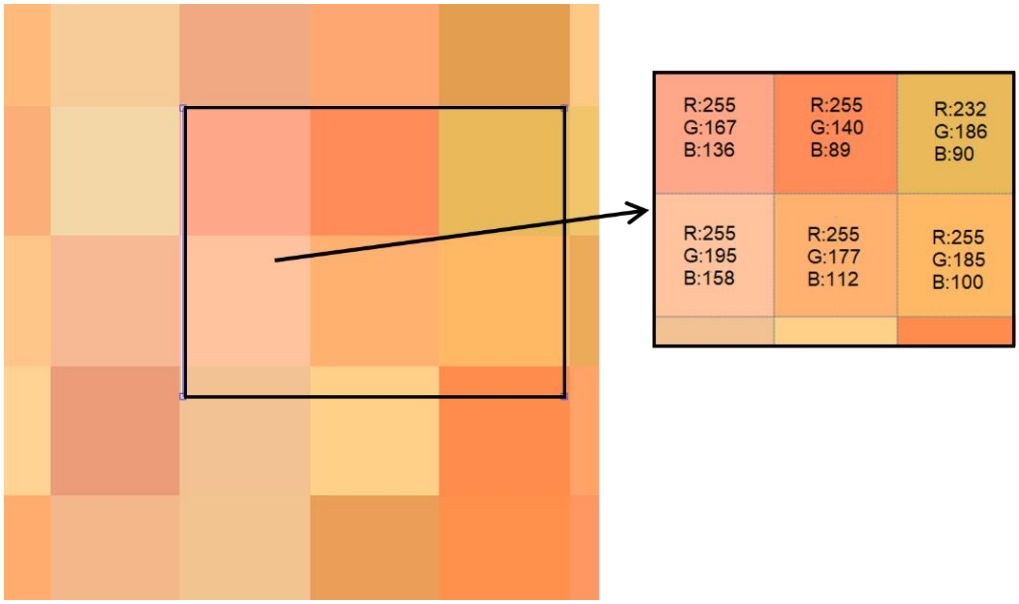


Fig. 2.4. Imagen a color en formato RGB.

La representación de coordenadas del modelo RGB se realiza mediante un cubo unitario con los ejes R, G y B [33]. El modelo RGB es uno de los más utilizados para crear y reproducir los colores en dispositivos como monitores y pantallas [33].

2.2.3.1. Métodos en el dominio espacial

Los métodos de procesamiento de imágenes se dividen en algoritmos en el dominio espacial y algoritmos en el dominio de la frecuencia [33]. Para el interés de este proyecto se describirán los métodos en el dominio espacial.

Los métodos en el dominio espacial procesan una imagen píxel por píxel o considerando la información de un grupo de pixeles cercanos (vecindario) [33].

Un píxel p situado en una imagen en las coordenadas x, y tiene 4 vecinos ubicados vertical y horizontalmente conocidos como $N4(p)$, con las siguientes coordenadas [34]:

$$(\mathrm{x}, \mathrm{y}+1), (\mathrm{x}, \mathrm{y}-1), (\mathrm{x}+1, \mathrm{y}), (\mathrm{x}-1, \mathrm{y}) \tag{2.3}$$

Mientras que sus vecinos diagonales se conocen como $ND(p)$ con las coordenadas [34]:

$$(x-1, y+1), (x-1, y-1), (x+1, y+1), (x-1, y+1) \tag{2.4}$$

El píxel p y sus vecinos crean una región. Esta región establece que dos pixeles son adyacentes si, y solo si, tienen en común alguna de sus fronteras, o al menos una de sus esquinas [33].

El conjunto de pixeles vecinos al píxel actual se le llama ventana o plantilla. Una ventana común es de tamaño 3 x 3, que incluye el vecindario $N8(p)$ de 8 pixeles que rodean a un píxel p [34]. La Fig. 2.5 muestra las relaciones de vecindad descritas.

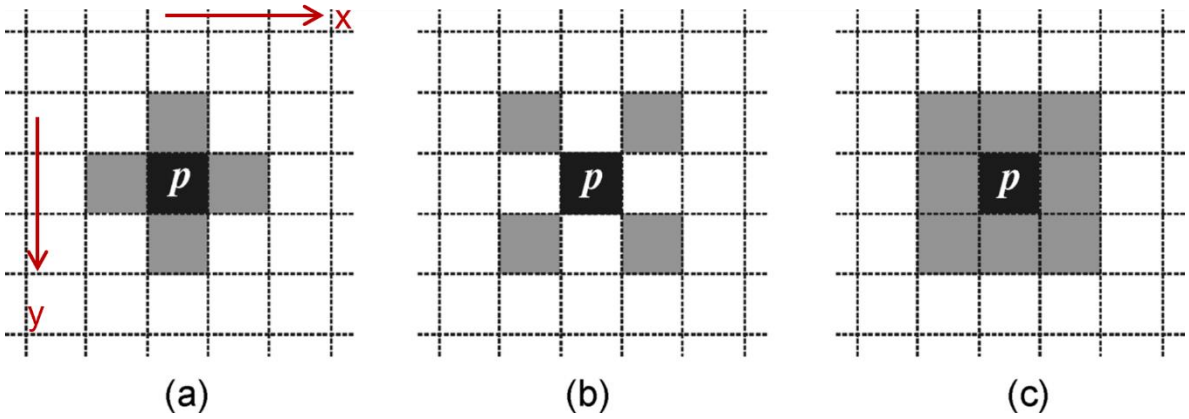


Fig. 2.5. Vecindarios de pixeles: (a) $N4(p)$, (b) $ND(p)$ y (c) $N8(p)$. Imagen tomada de [35].

Las operaciones espaciales reciben una imagen, y recorren cada uno de los pixeles, usando una ventana de vecindades de tamaño N × M. De tal forma que se procesan todos sus elementos aplicando una transformación sobre ellos, lo cual se define como:

$$\mathrm{g(x,y)} \;=\; \mathrm{T[\,f(x,y)\,]} \tag{2.5}$$

donde $\mathrm{f(x,y)}$ es la imagen de entrada, $\mathrm{g(x,y)}$ es la imagen resultante, y T es un operador que se aplica sobre la imagen, el cual se define sobre los vecinos del píxel $(\mathrm{x,y})$. Si el operador solo aplica al píxel actual, T sería una matriz de tamaño 1 × 1 tomando un valor constante.

2.2.3.2. Filtrado y convolución

El filtrado de imágenes es una operación que modifica el valor de un píxel p usando un filtro proporcionado. Una máscara de filtrado es un arreglo bidimensional con valores reales conocidos como pesos [33].

El punto de origen de una máscara simétrica o con dimensiones impar se sitúa en el centro de esta misma. Para máscaras no simétricas su origen puede ser situado en cualquier punto de esta misma. Para aplicar la máscara ubican su centro sobre un píxel p, entonces cada valor de esta será multiplicado por los elementos que componen al vecindario del píxel, y posteriormente se suman los resultados para sustituir el valor del píxel p [33]. El proceso de filtrado consiste en aplicar la operación anterior a cada píxel de la imagen y este es conocido como filtrado correlacional [8].

La convolución es una operación similar, pero con una regla establecida para multiplicar los valores del filtro con los pixeles del vecindario [8], la cual se define de forma matemática para un píxel en la posición x, y como:

$$g(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x+i, y+j)\, A(i,j) \qquad (2.6)$$

donde A es una máscara de convolución de dimensiones m × n, I una imagen de M × N, $\mathrm{a} = (\mathrm{n}-1)/2$ y $\mathrm{b} = (\mathrm{m}-1)/2$. Un ejemplo de la operación de convolución se presenta en la Fig. 2.6, donde la operación de convolución se denota por el operador $*$.

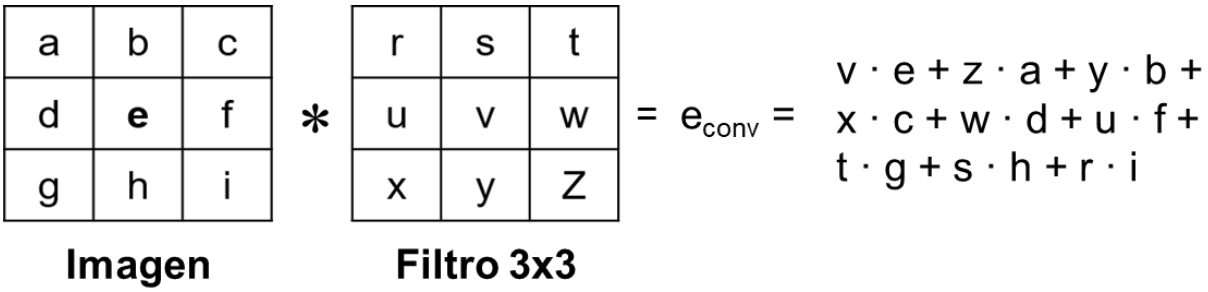


Fig. 2.6. Convolución operando sobre el píxel *e*.

La operación de convolución se repite para todos los pixeles de la imagen conforme el filtro de convolución recorre la imagen. Al tamaño del salto que realiza el filtro en la convolución se conoce como stride [8]. Con stride = 1, el filtro avanza por la imagen realizando un salto entre cada columna de píxel. Si aumentamos el número de saltos, entonces la imagen de salida tendrá dimensiones aún más cortas [8] (ver Fig. 2.7).

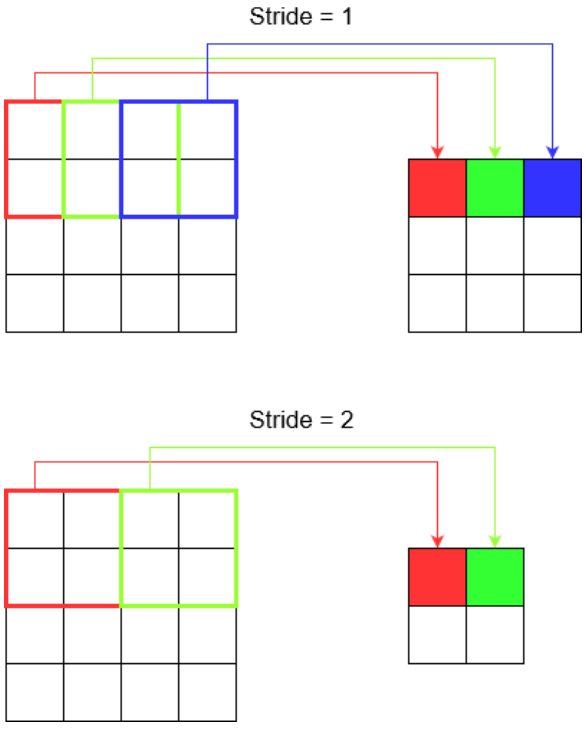


Fig. 2.7. Visualización de la operación de convolución con diferentes saltos. Imagen tomada de [36].

Sin importar el tamaño del salto, después de la operación de convolución existe una reducción de las dimensiones de la imagen resultante. Para conservar las dimensiones de la imagen de salida igual al tamaño de la imagen de entrada, se puede agregar bordes de ceros (zero-padding) en las dimensiones horizontales y verticales de la imagen original [8]. Un ejemplo de convolución con stride $S = 1$ y zero-padding $T = 1$ se muestra en la Fig. 2.8.

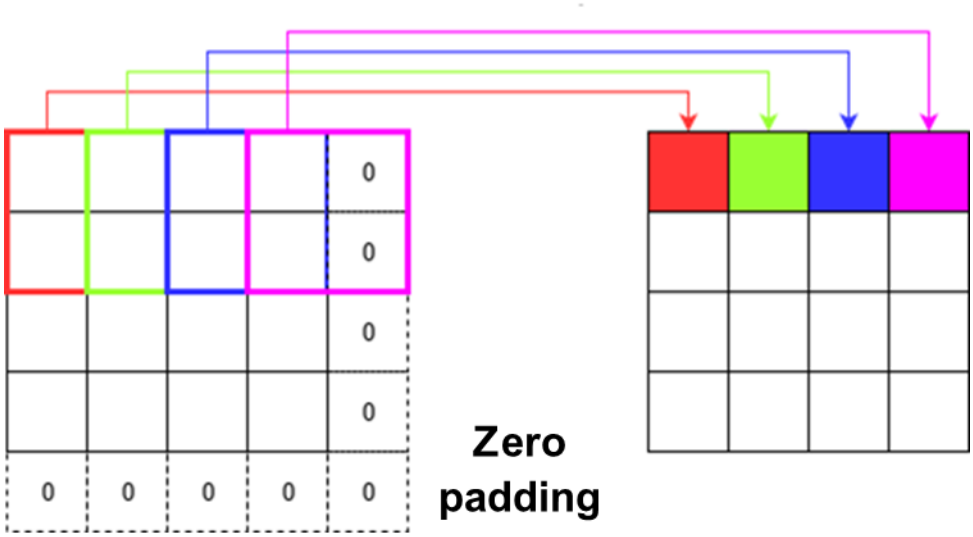


Fig. 2.8. Convolución con T=1: imagen de entrada 4 x 4 e imagen de salida 4 x 4. Imagen tomada de [36].

Para un filtro de convolución de tamaño F × F, una imagen de tamaño M × N, stride de tamaño S, y T el incremento de cada dimensión (usando zero-padding), las nuevas dimensiones M' × N' de la imagen resultante se calculan de la siguiente forma [8]:

$$M' = \left\lfloor \frac{M - F + S + T}{S} \right\rfloor, N' = \left\lfloor \frac{N - F + S + T}{S} \right\rfloor \quad (2.7)$$

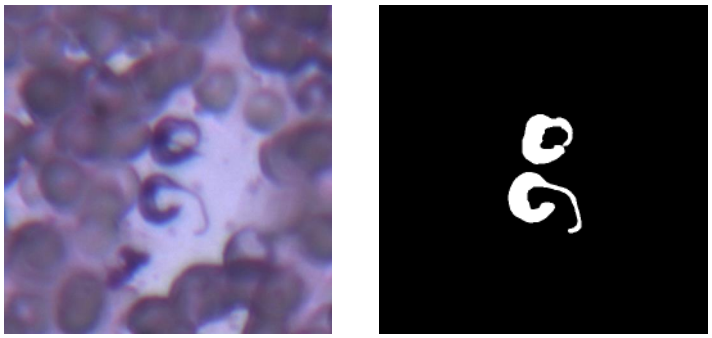
El filtrado y convolución de imágenes es empleado para diversas aplicaciones, como la minimización de ruido, la aproximación de derivadas, detección de bordes en imágenes, entre otras [33]. Los filtros convolucionales también son usados para extraer elementos descriptivos o características de las imágenes que ayudan a las tareas de reconocimiento de patrones de sistemas inteligentes. La extracción de características es un método que toma una imagen como entrada y extrae atributos de interés sobre ella [8]. La extracción de características es la primera etapa en la inteligencia de un sistema de visión por computadora y constituye un inmenso campo de estudio e investigación para una multitud de aplicaciones [8]. Un sistema de visión artificial necesita extraer de la forma más robusta, eficaz y rápida, las características de la imagen para proporcionar información que se necesita para un paso posterior de interpretación. Por ejemplo, una red neuronal completamente convolucional (FCNN) (de la cual se abordará más adelante) encuentra los pesos adecuados de los filtros convolucionales para resolver un problema de segmentación en imágenes [8].

2.2.4. Segmentación semántica

La segmentación semántica, también conocida como clasificación a nivel de píxel, es una tarea crítica en visión por computadora [33]. Con la segmentación se identifican las partes de una imagen entendiendo a que clase pertenecen. La segmentación de una imagen involucra el etiquetado de todos sus pixeles con una en clase particular [8]. Estas clases son semánticamente interpretables, y corresponden a categorías del mundo real. Esto resulta en la partición de la imagen en diferentes segmentos, de tal forma que cada segmento pertenece a

una entidad diferente, eliminando la necesidad de considerar a los pixeles individuales como unidades de observación.

Por ejemplo, para el problema de segmentación del parásito *T. cruzi*, la agrupación de pixeles correspondiente al parásito representaría a la clase principal (foreground) y los pixeles de no interés a la clase de fondo (background). En la Fig. 2.9 podemos observar un ejemplo de una imagen segmentada, en donde los pixeles blancos representan a la clase principal y los pixeles negros a la clase de fondo.



(a) (b)

Fig. 2.9. Ejemplo de segmentación. (a) imagen original, (b) Imagen segmentada.

La segmentación es útil es muchas áreas, aunque tal vez la principal es en aplicaciones médicas [37]. Un sistema inteligente que pueda identificar partes específicas relacionadas con alguna afección ocasionada por enfermedades, parásitos u otros componentes del cuerpo entendiendo el contexto puede generar un impacto profundo en el cuidado médico [38].

Un método básico de segmentación es la umbralización [33]. Este proceso suele ser aplicado en imágenes en escala de grises, y consiste en comparar el valor de cada píxel de una imagen con una bandera o umbral definido. Si el píxel es menor

o igual al valor de la bandera, al píxel se le asigna la clase de fondo. En caso contrario, si es mayor a la bandera se le asigna la clase principal. Un píxel perteneciente a la clase principal se le asigna el valor de 1 que se escala posteriormente a 255. El proceso lo podemos definir para un píxel en la posición x, y como:

$$g(x,y) = \begin{cases} 1, & si\ I(x,y) > T \\ 0, & si\ I(x,y) \leq T \end{cases} \quad (2.8)$$

donde T es un umbral, $I()$ es una imagen y $g()$ es la nueva imagen segmentada.

La segmentación también es conocida como una clasificación a nivel de píxel debido a que se le asigna a cada píxel una clase dependiendo del método de discriminación usado para segmentar [33]. Si el problema de clasificación solo requiere asignar dos posibles clases, esta se trata de una segmentación binaria.

Para realizar la segmentación de imágenes a color existen diversas técnicas pertenecientes al área de aprendizaje automático y aprendizaje profundo que son útiles [33]. Para el problema de la segmentación del parásito *T. cruzi* implementamos un método basado en aprendizaje profundo para clasificar cada píxel de una imagen a color.

2.3. Aprendizaje automático

Los algoritmos de aprendizaje automático y de aprendizaje profundo han hecho importantes contribuciones en el procesamiento de imágenes y de visión por computadora [39]. Estas metodologías se han convertido en herramientas esenciales para la predicción y la toma de decisiones en muchas disciplinas [38].

El aprendizaje automático es el nombre de la disciplina que engloba técnicas y herramientas tecnológicas que les permiten a las computadoras realizar tareas complejas con una alta exactitud [38]. En aplicaciones médicas, los métodos basados en aprendizaje automático pueden ayudar a resolver problemáticas relacionadas con el diagnóstico y predicción de enfermedades [40].

El objetivo del aprendizaje automático es diseñar métodos que automáticamente realicen el aprendizaje usando observaciones del mundo real (conocido como datos de entrenamiento), sin una definición de explicita de las reglas o la lógica que intervendrá [8]. El aprendizaje de las computadoras se logra si el desempeño en la solución de un problema mejora a medida que su experiencia aumenta. Por ejemplo, para un problema sencillo de reconocimiento de objetos, la experiencia es adquirida a través características que un clasificador recibe durante el entrenamiento para resolver el problema, y el desempeño comúnmente se mide con el porcentaje de elementos reconocidos de forma correcta [33].

La mayoría de los métodos de aprendizaje automático son métodos de aprendizaje supervisado debido a que han obtenido mayor rendimiento comparado otros [8]. En un método de aprendizaje supervisado, los datos de entrenamiento toman la forma de una colección de pares (datos: x, etiqueta: y) y el objetivo es producir una predicción y^* en respuesta a una muestra de entrada x. Los métodos de aprendizaje automático aproximan una función de mapeo $f(x)$ la cual puede predecir las variables de salida y para una estrada x [8]. La entrada x puede ser un vector de características o datos más complejos como imágenes para el problema de segmentación semántica.

2.3.1. Clasificación binaria

La segmentación binaria puede ser vista como un problema de clasificación binaria debido a que se debe asignar a cada píxel una de dos posibles clases (principal y fondo) [33]. En la clasificación binaria [8] intervienen un conjunto de características (instancias x) pertenecientes a un espacio de instancias X, un conjunto de clases $y \epsilon \{0,1\}$, y cada instancia está relacionada con una clase la cual se denota como (x, y) donde $x \epsilon X$ y $y \epsilon \{0,1\}$. En la segmentación binaria de una imagen, un clasificador recibe un conjunto de características representativas a un píxel o grupo de pixeles $P,$ y asigna una clase y para cada píxel contenido en la imagen.

Algunos clasificadores útiles para la segmentación binaria son las Máquinas de Soporte Vectorial (SVM), K-means clustering, análisis de la discriminante de Gauss, clasificador Bayesiano, clasificador AdaBoost, entre otros [37]. Algo característico de estos métodos convencionales de aprendizaje automático es la selección o definición de las características importantes que se extraerán como un paso inicial antes de realizar el proceso de clasificación [8, 33].

2.3.2. Aprendizaje profundo

El aprendizaje profundo pertenece a una rama avanzada de aprendizaje automático. Esta rama trae inteligencia a las computadoras de tal forma en que puedan extraer patrones de grandes cantidades de datos y procesarlas para un razonamiento automático. Las metodologías basadas en aprendizaje profundo han mejorada por mucho a los métodos convencionales de aprendizaje automático, ya

que no se necesita definir propiamente que o cuáles características se extraerán, si no que los algoritmos aprenden a encontrarlos de forma autónoma durante el entrenamiento para lograr la clasificación de forma más eficiente [8, 37, 41].

Los algoritmos basados en las Redes Neuronales Convolucionales (CNN), Redes Neuronales Completamente Convolucionales (FCNN), Redes Neuronales Profundas y Redes Neuronales Recurrentes, incluidas en la categoría de aprendizaje profundo, se han convertido rápidamente en una metodología de uso para el análisis de imágenes biomédicas debido al alto desempeño que se ha obtenido en los resultados de clasificación, detección y segmentación de objetos [28-30, 37, 40-45]. Para la segmentación de imágenes la metodología adecuada a usar corresponde a las Redes Neuronales Completamente Convolucionales (FCNN).

2.3.2.1. Red Neuronal Completamente Convolucional

Una red neuronal completamente convolucional (FCNN) es una versión modificada de una CNN [8]. Las CNN son usadas principalmente en visión por computadora para la clasificación de imágenes, reconocimiento de caras e identificación y clasificación de objetos, conducción autónoma, etc. Las CNN usan una arquitectura tridimensional con capas convolucionales para analizar las imágenes, identifican y extraen características importantes, y las utilizan para clasificar la imagen por medio de unas capas conocidas como Redes Neuronales Completamente Conectadas (FC) para obtener un resultado final de detección o clasificación [8]. A diferencia de una CNN, las FCNN solo están compuestas de

capas convolucionales y son usadas para la segmentación de imágenes [8], como se muestra en la Fig. 2.10.

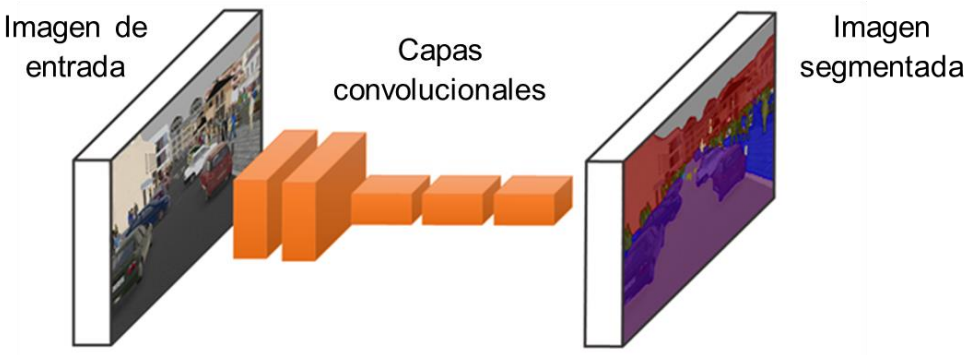


Fig. 2.10. Ejemplificación de una FCNN. Imagen tomada de [46].

Una FCNN pertenece a la clasificación de aprendizaje supervisado por lo que es necesario contar con las etiquetas o segmentaciones verdaderas de nuestro conjunto de imágenes para lograr el entrenamiento de la red [8]. A este conjunto de imágenes se le suele conocer como ground truth. La primera capa en una FCNN es la imagen de entrada, con un tamaño de $h \times w \times$ d, donde h y w son dimensiones espaciales y d es la dimensión de los canales. Inicialmente una imagen de entrada pasa por una capa de convolución. Cada salida de una capa convolucional es un arreglo tridimensional de tamaño $h \times w \times d$, donde d es la dimensión de las características o del canal [47]. A la dimensión del canal también suele referirse como profundidad.

La convolución con filtros es la operación central en una capa convolucional (de ahí su nombre) y fue explicado previamente en la sección 2.2.3.2. En una capa convolucional, un filtro es tomado para ser operado por toda la imagen o volumen de entrada para obtener un mapa de salida o de características (ver Fig. 2.11). Si se desea mantener la resolución de la imagen con el mismo ancho y alto que la

imagen de entrada se implementa zero-padding en todas las operaciones de convolución de la FCNN.

Cada píxel del mapa de características es resultado de la convolución sobre una zona concreta de la imagen de entrada, y al tamaño de la zona se le conoce como campo receptivo de la red (RF) (Fig. 2.11). Cuando apilamos una serie de capas convolucionales, una detrás de otra, el campo receptivo (RF) de cada capa se convierte en una función de los campos receptivos de todas las capas previas [8].

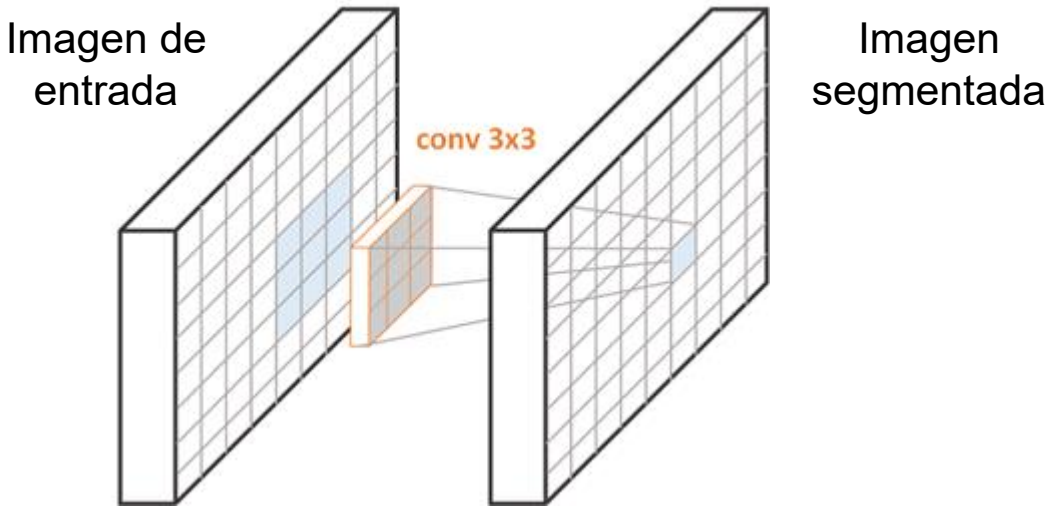


Fig. 2.11. FCNN con una capa de convolución. Cada píxel de salida corresponde a un RF de 3 × 3 de la imagen de entrada. Imagen tomada en [48].

La aplicación sistemática de un filtro por toda la imagen recorriendo cada píxel es una herramienta poderosa. Si el filtro está diseñado para detectar un tipo específico de características en la entrada, entonces la aplicación de ese filtro en toda la imagen permite descubrir esa característica en cualquier lugar de la imagen. Esto es conocido como invarianza a la traslación y permite que una red

convolucional pueda detectar las características en cualquier región de la imagen incluso si estas regiones cambian de posición [47].

La operación de convolución en una red neuronal permite que los valores de los filtros (pesos) sean aprendidos durante el entrenamiento de la red. La red aprenderá que tipos de características extraer de las entradas. De forma más precisa, se entrena a la red para encontrar las características que minimicen el error de una tarea específica que se intenta realizar [8] (se describe a detalle en la sección 2.3.2.5). Por ejemplo, extraer aquellas características útiles para la segmentación semántica del parásito *T. cruzi*.

Dado que las FCNN son una herramienta poderosa, en la práctica, una capa convolucional aprende más de un filtro de convolución por entrada. De hecho, suelen aprender de 32 hasta 1024 filtros en paralelo para una entrada, dependiendo del objetivo y la estructura de la red. Esto resulta en hasta 1024 formas de extraer características de una entrada, logrando extraer las características más importantes para el problema específico [8]. La salida de una capa convolucional tiene profundidad o dimensión de canal d igual al número de filtros utilizados.

La descripción hecha hasta ahora de las capas convolucionales aplica para las entradas con un solo canal ($d = 1$), como una imagen en escala de grises o un mapa de características con profundidad 1.

En una imagen a color en formato RGB, d es de dimensión 3. Visto de forma matricial, una imagen RGB de entrada en una FCNN es en realidad tres imágenes

apiladas por canal. Por lo que un filtro de convolución debe de tener una profundidad igual a la dimensión del canal de la imagen o mapa de características que entra a la capa convolucional. Por ejemplo, para una imagen RGB un filtro de convolución 3 × 3 tendría profundidad 3. Resultando así en un tamaño de filtro de 3 × 3 × 3. La convolución en imágenes con 3 canales se mantiene como la misma operación de producto de los elementos con el filtro de convolución, teniendo pesos específicos para cada uno de los tres canales, de tal forma que al final se suman todos los resultados obteniendo un solo valor (ver Fig. 2.12).

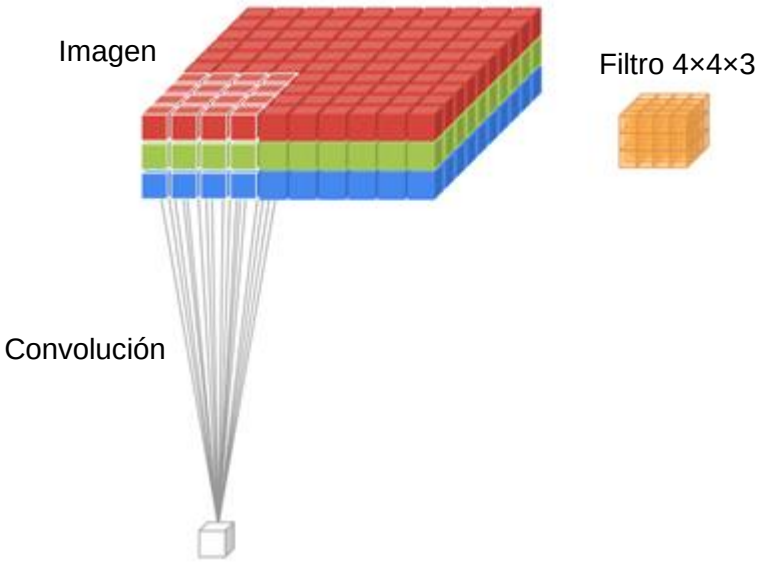


Fig. 2.12. Convolución de un filtro 3D en una imagen RGB. Imagen tomada de [49].

De igual manera, si la capa convolucional tiene, por ejemplo, 4 filtros de convolución, obtendremos un mapa de características de salida con profundidad 4 (ver Fig. 2.13).

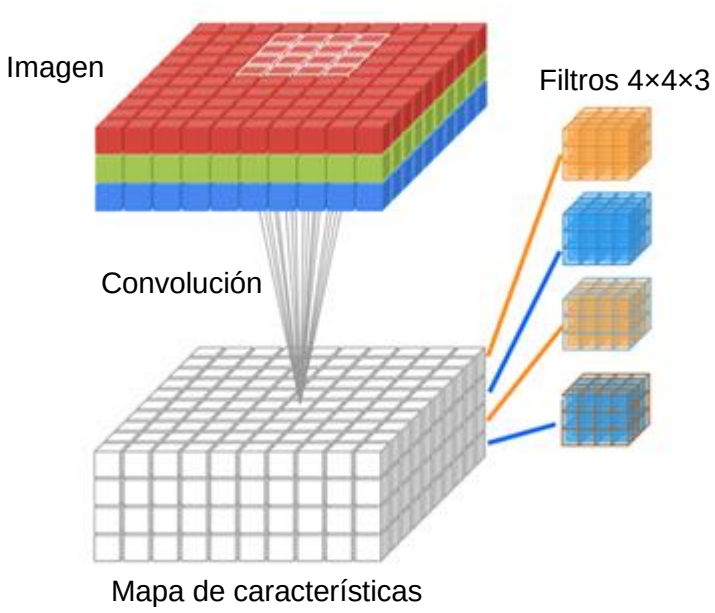


Fig. 2.13. Mapa de características de salida con profundidad 4. Imagen tomada de [50].

Una capa convolucional puede recibir una imagen o un mapa de características devuelto de otra capa convolucional. El apilamiento de las capas convolucionales permite una descomposición jerárquica de las entradas. De tal forma que las primeras capas de convolución extraen características de bajo nivel, haciendo que las siguientes capas procesen su información a partir de estas. Este proceso continúa hasta alcanzar una profundidad de la red, por lo que las características que se encuentran en las capas más profundas se conocen como características de alto nivel. Aquí las características representan información más abstracta que las de bajo nivel [8]. El mapa de características devuelto codifica en cierto grado la presencia o ausencia de las características detectadas [8]. El mapa de características devuelto por la última capa convolucional en una FCNN tendrá una profundidad igual a la cantidad de clases del problema de segmentación (ver Fig. 2.14). Este mapa de características de salida corresponde a la imagen

segmentada final y tiene un gran campo receptivo correspondiente al ancho y alto de la imagen de entrada original [8, 47].

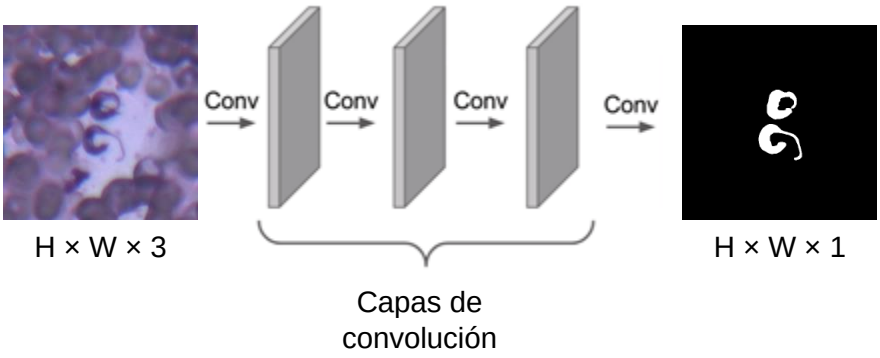


Fig. 2.14. Segmentación binaria con una FCNN.

2.3.2.2. Funciones de activación

Un mapa de características devuelto por una capa en una FCNN es comúnmente seguido por una función de activación función no lineal [8]. A estas funciones también se les conoce como Capas de Activación. Son usadas para incrementar la no-linealidad de la red sin afectar los campos receptivos de las capas convolucionales [8]. La Rectifier Linear Unit (ReLU) es la función de activación más usada en las redes convolucionales [8]. La salida de una ReLU está dada por la función:

$$f(x) = \max(0, x) \tag{2.9}$$

donde x es un valor del mapa de características. La función genera un nuevo volumen de salida de tal forma que, al igual que los datos del mundo real, la red solo aprende valores no negativos [8].

Otra función de activación es la sigmoidal [8]. La sigmoidal está dada por la función:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.10}$$

esta función recibe un valor real de entrada x, y devuelve un número en el rango de 0 y 1.

En la práctica se suele preferir el uso de la activación ReLU para un entrenamiento más rápido de las capas convolucionales, mientras que la función sigmoide puede ser usada en la capa final de una FCNN para obtener las probabilidades de los pixeles que generarán un mapa de segmentación binario [8].

2.3.2.3. Downsampling y Upsampling

Las técnicas para cambiar o disminuir la resolución de un mapa de características se le conocen como métodos de downsampling. Algunos métodos de downsampling se mencionaron previamente en sección 2.2.3.2, lo cuales incluyen el uso de stride y el zero-padding en la operación de convolución. Otro método común implementado en las redes neuronales es la capa de Pooling [8].

El Pooling, también conocido como submuestro o método de reducción de dimensiones (subsampling), tiene como propósito retener o comprimir la información importante de un volumen de datos. Una capa de Pooling reduce las dimensiones (ancho y alto) de un mapa de características de entrada, pero conservando la dimensión de los canales [8]. El Pooling puede ser realizado de distintos tipos, siendo el Max Pooling el más usado en una CNN.

El Max Pooling toma el elemento más grande de un vecindario de pixeles rectangular definido por una ventana de tamaño $n \times n$. Al igual que la operación de convolución se define un tamaño de stride para realizar el Pooling (ver Fig. 2.15). Es importante mencionar que la operación de Max Pooling es un paso a parte a la convolución.

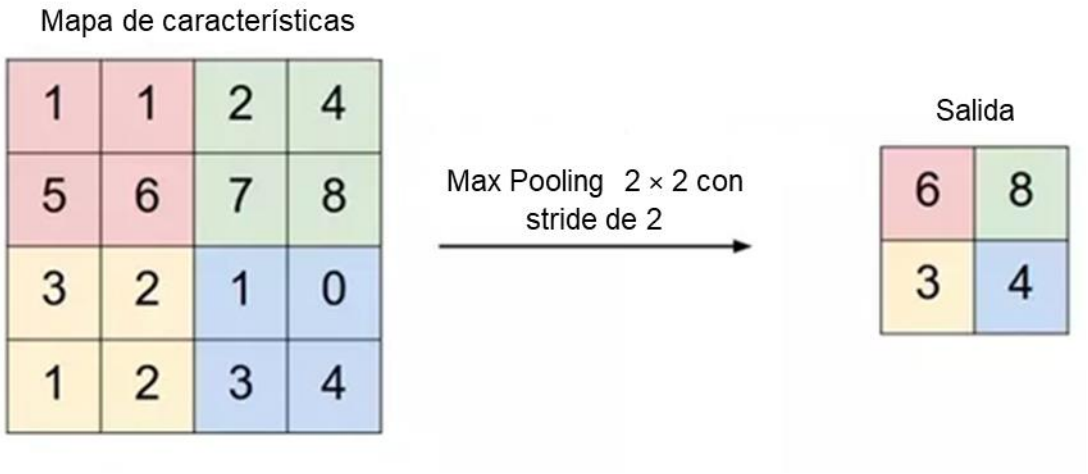


Fig. 2.15. Ejemplo de Max Pooling de tamaño 2 × 2. Imagen tomada de [51].

Apilar una serie de capas de convolución para trabajar con imágenes en alta resolución tiene un alto costo computacional. Si se requiere mantener un tamaño grande de la imagen de salida, se necesita agregar más capas de convolución para poder generar las características de bajo y alto nivel, lo que resulta computacionalmente ineficiente preservar la resolución de cada mapa de características en la red para obtener la segmentación final [47]. Para lograr que las redes sean eficientes con la cantidad de parámetros que usan se recurre a métodos de downsampling conforme aumenta la profundidad de una FCNN. Sin embargo, un problema que se presenta con esto es la pérdida de la ubicación espacial (en el espacio de pixeles) de las características [47]. Es decir, la

segmentación final es una imagen con una resolución muy pequeña comparada a la resolución de la imagen de entrada. Por lo que una FCNN recurre a métodos de upsampling para obtener nuevamente la información de la ubicación espacial de las características para generar el mapeo y clasificación de los pixeles de la imagen de salida [47].

Mientras el objetivo de un método de downsampling es capturar la información semántica/contextual, el método de upsampling recupera su información espacial [8, 47]. Un método de upsampling es la convolución transpuesta [8]. Esta también es erróneamente conocida como deconvolución. La convolución transpuesta, va en un sentido opuesto a la convolución de tal forma que permite trasladar los valores de un mapa de características a un mayor tamaño, a través de la ampliación de la resolución del mapa de características (Fig. 2.16).

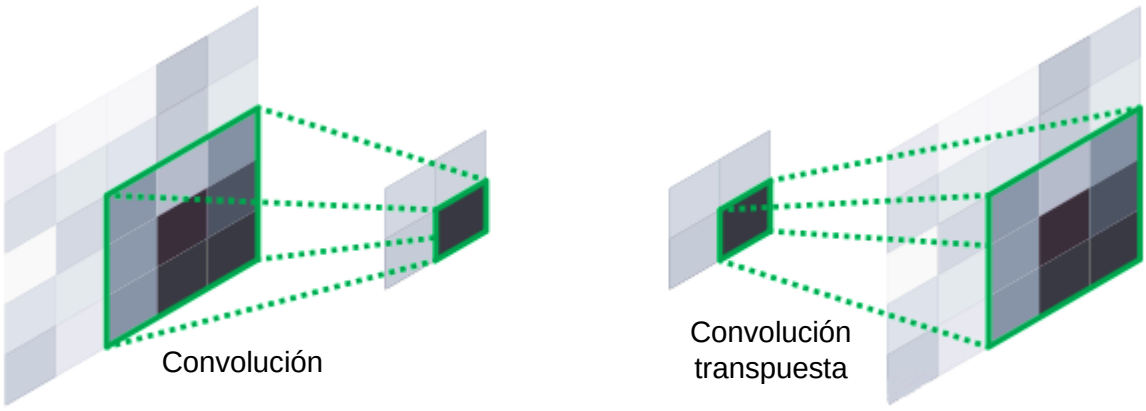


Fig. 2.16. Ilustración de la operación de convolución transpuesta. Imagen tomada de [52].

En la convolución transpuesta, cada valor de un mapa de características se multiplica con los pesos de un filtro de convolución transpuesta, proyectando los nuevos valores en un mapa de características de salida, de tal forma que se

genera una salida más grande [53]. Al igual que la convolución, la convolución transpuesta contiene pesos que se aprenden en una FCNN para poder generar el upsampling correcto. Un ejemplo de la operación de convolución transpuesta para un arreglo de una dimensión se presenta en la Fig. 2.17.

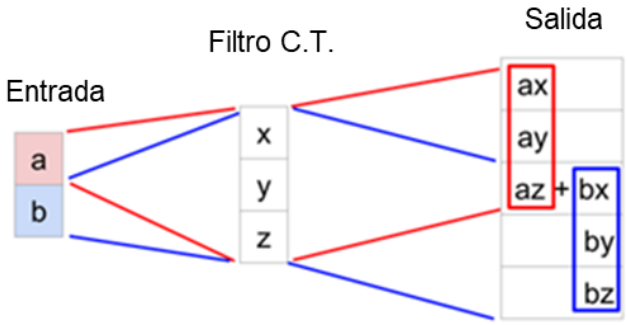


Fig. 2.17. Convolución transpuesta en una dimensión. Imagen tomada de [54].

Un ejemplo de la convolución transpuesta de dos dimensiones se presenta en la Fig. 2.18.

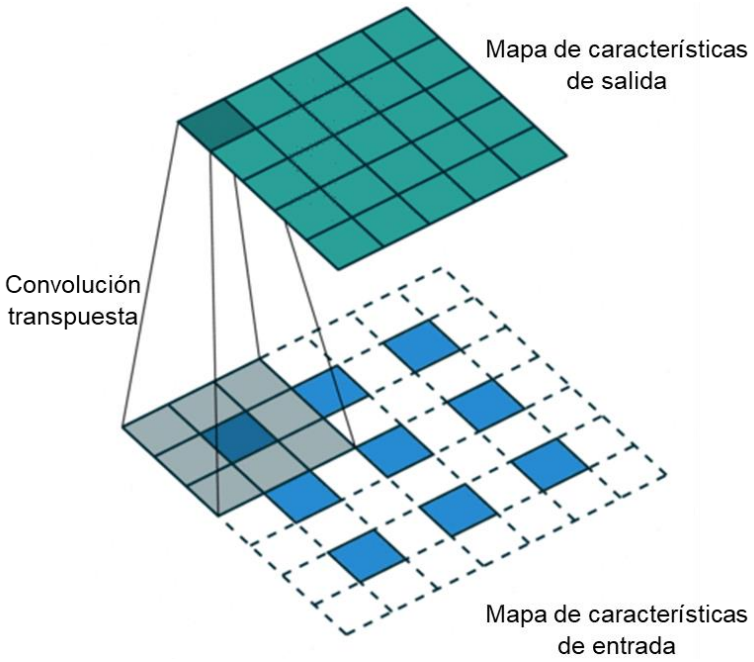


Fig. 2.18. Convolución transpuesta con un filtro 3 × 3 y stride = 2. Imagen tomada de [55].

En la siguiente sección se describirá la arquitectura común de una FCNN que utiliza métodos de downsampling y upsampling para generar la segmentación de una imagen de forma eficiente.

2.3.2.4. Configuración de una FCNN

La idea de usar una arquitectura completamente convolucional entrenada de inicio a fin para la tarea de segmentación semántica fue introducida por Long et al. [47] en 2014. El objetivo deseable en una FCNN es lograr obtener una imagen de segmentación de resolución completa. Una FCNN está diseñada para procesar distintos tamaños de entrada de forma eficiente con una arquitectura que incluye comúnmente una red codificadora y una red decodificadora [8].

Una red codificadora contiene series de capas convolucionales apiladas consecutivamente, mientras que la red decodificadora contiene series de capas de convolución y de convolución transpuesta apiladas consecutivamente. La arquitectura de una FCNN se muestra en la Fig. 2.19.

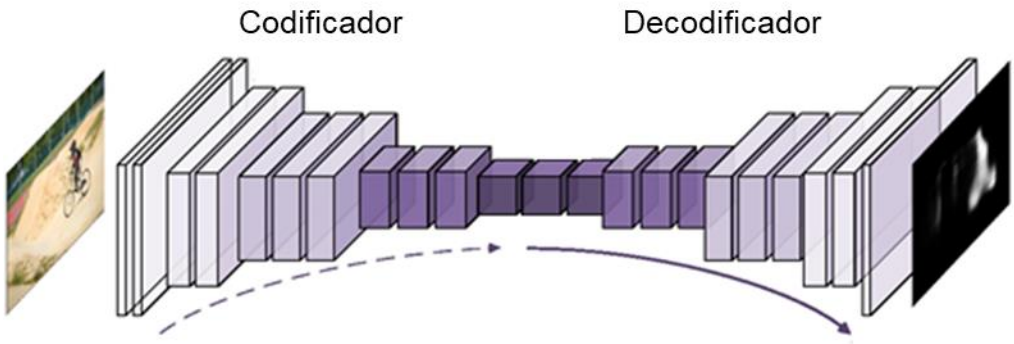


Fig. 2.19. Arquitectura eficiente de una FCNN. Imagen tomada de [56].

Una capa perteneciente a la red codificadora incluye el apilamiento de una serie de capas de convolución, seguidas de una capa de activación (función de activación) y una capa de Pooling. Una capa en la red decodificadora incluye una capa de convolución transpuesta y una serie de capas convolución seguidas de una capa de activación.

Debido a que la red codificadora reduce la resolución de los mapas, la segmentación resultante carece de bordes bien definidos ya que algo de la información se ha perdido previamente con el método de downsampling [47] (ver. Fig. 2.20).

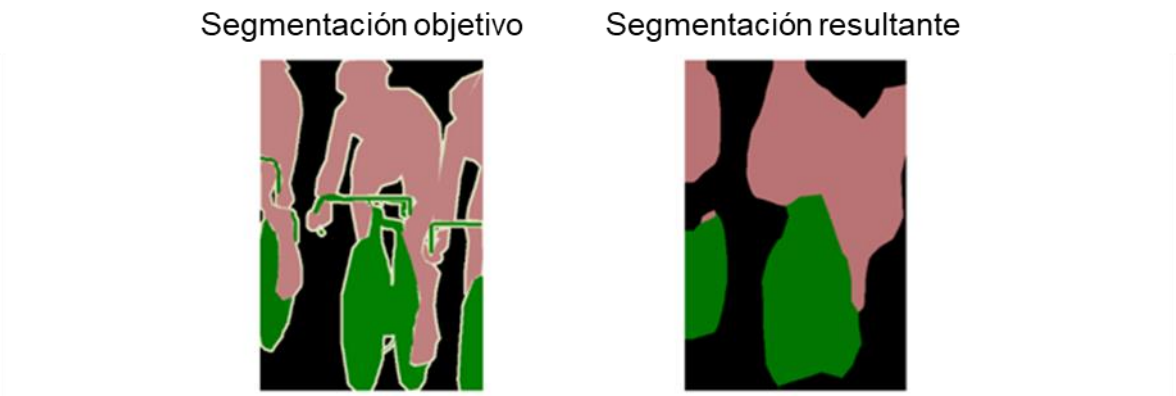


Fig. 2.20 Segmentación resultante con una FCNN tradicional. Imagen tomada de [47].

Para mitigar este problema los autores en [47] propusieron las conexiones de salto (Skip Connections). Las conexiones de salto a traviesan diversas capas y transfieren su información a otras. En las FCNN las capas de la red codificadora pasan su información intacta a las capas de la red decodificadora. Esta retroalimentación de capas ayuda a mejorar los detalles de la segmentación con formas y bordes mucho más precisos. Una arquitectura de FCNN que implementa

las conexiones de salto y ha tenido éxito en la segmentación de imágenes biomédicas es la U-Net [57].

2.3.2.5. Funciones de costo

Para medir que tan bien se está realizando la tarea de segmentación se utiliza una función de costo. Una función de costo mide el error de las predicciones durante el entrenamiento para ayudar a los optimizadores a actualizar los parámetros de la red con el fin de minimizar el error [58, 59]. El costo será alto si las imágenes segmentadas devueltas por la red son incorrectas, y bajo si son similares al conjunto de imágenes del ground truth.

Para lograr un entrenamiento óptimo de una FCNN necesitamos seleccionar una función de costo adecuada al problema para obtener una segmentación precisa [8, 58]. El costo se calcula en la última capa de la red por cada imagen de entrenamiento y representa un valor numérico que se promedia con el número total de muestras utilizadas para el entrenamiento [8].

Las funciones de costo más comunes para la segmentación binaria se basan en la similitud de los pixeles entre las predicciones y las muestras del ground truth, como la Binary Cross Entropy (BCE) y Weighted Binary Cross Entropy (WBCE) [58].
La función de costo BCE se define como:

$$BCE(p, \hat{p}) = -(plog(\hat{p}) + (1 - p)\, log(1 - \hat{p})), \tag{2.11}$$

donde $Y = 1$ representa la clase principal, $Y = 0$ es la clase de fondo, $P(Y = 1) = p$ y $P(Y = 0) = 1 - p$ son los valores del ground truth para ambas clases, y las probabilidades predichas para ambas clases están dadas por la función sigmoide:

$$P(\hat{Y} = 1) = \frac{1}{1+e^{-x}} = \hat{p} \text{ y } P(\hat{Y} = 0) = 1 - \hat{p} \quad (2.12)$$

La función de costo WBCE esta definida en términos de la función BCE:

$$WBCE(p, \hat{p}) = -(\beta p log(\hat{p}) + \alpha(1 - p)\, log(1 - \hat{p})), \quad (2.13)$$

donde β representa la proporción de la clase principal y α la proporción de la clase de fondo. Darle mayor peso a la clase principal permite evitar un desbalance de clases durante el entrenamiento.

Aunque con estas funciones de costo, los autores han logrado un buen rendimiento de segmentación [60], las imágenes segmentadas devueltas pueden incluir ruido y algunos pixeles en los límites de los objetos pudieran perderse incluso si el costo es bajo [60]. Para solucionar este problema, Chen et al. [60] propuso la función de costo de Contornos Activos (AC) como una nueva función de costo para la segmentación de imágenes biomédicas. La función de costo AC penaliza ambos: la longitud del contorno del objeto, y el área del objeto. Los autores aseguran que la función de costo AC puede llevar a una segmentación precisa debido a que la medición del error es más precisa al usar información geométrica.

La función de costo AC se define como:

$$AC = Longitud + \lambda \cdot Area, \quad (2.14)$$

con una representación a nivel de píxel de la Longitud en (2.15) como:

$$Longitud = \sum_{\Omega}^{i=1,j=1} \sqrt{| \left(\nabla P_{x_{i,j}}\right)^2 + \left(\nabla P_{y_{i,j}}\right)^2 | + \epsilon}, \quad (2.15)$$

donde Ω = $\mathbb{R}^2$, P es la segmentación predicha, x y y son las direcciones horizontales y verticales respectivamente, y ϵ es un parámetro para evitar que la Longitud sea cero. En los experimentos de los autores seleccionaron el parámetro ϵ =1E-8. El *Area* se define en (2.16) como:

$$Area = \left|\sum_{\Omega}^{i=1,j=1} P_{i,j}\left(C_1 - T_{i,j}\right)^2\right| + \left|\sum_{\Omega}^{i=1,j=1}(1 - P_{i,j})\left(C_2 - T_{i,j}\right)^2\right|, \quad (2.16)$$

donde T es la muestra del ground truth, $C_1 = 1$ representa la energía dentro del objeto (foreground) y $C_2 = 0$ la energía de afuera (background). Chen et al. [60] propone experimentar con diferentes valores del parámetro λ para seleccionar el que mejor rendimiento de segmentación devuelva.

2.3.2.6. Optimizador de ADAM

Una FCNN puede ser descrita como una función que trabaja sobre las imágenes para devolver una imagen en donde todos los pixeles son clasificados correctamente. Vista de forma matemática como:

$$P = g(X, W), \quad (2.17)$$

donde X es una imagen, P es la segmentación predicha por la red, W es una serie de pesos y $g(\cdot)$ es una FCNN. Los parámetros o los pesos de los filtros de las capas

convolucionales $W = (w_1, \dots, w_l)$ para l capas, deben ser aprendidos de tal manera que la función $g(\cdot)$ logre que las imágenes segmentadas predichas sean consistentes con las imágenes del ground truth por cada entrada X.

El entrenamiento de una FCNN es esencialmente la minimización de una función de costo J [8]. El trabajo de un optimizador es encontrar el conjunto de pesos W adecuado que minimice la función de costo J a través de la actualización de los pesos en la dirección opuesta del gradiente de la función objetivo $\nabla_w J(W)$ con respecto a los parámetros W. La tasa de aprendizaje η determina el tamaño de los pasos de actualización para que la función de costo alcance un mínimo local. Este mínimo es conocido como el valor mínimo que la función de costo puede alcanzar. Los gradientes son calculados usando un método eficaz como el algoritmo de retro propagación [8].

Dependiendo de la cantidad de datos de entrenamiento, se debe realizar una compensación entre la exactitud de la actualización de los parámetros y el tiempo que se invierte en actualizarlo. El algoritmo más conveniente para tal tarea es conocido como descenso de gradientes por lotes [8], que actualiza los parámetros después de cada mini lote de n ejemplos de entrenamiento de la siguiente forma:

$$W = W - \eta \cdot (\nabla_w J(W; x^{(i:i+n)}; y^{(i:i+n)}), \tag{2.18}$$

donde x^i es un ejemplo de entrenamiento y y^i es su correspondiente etiqueta. El algoritmo conlleva a una convergencia estable, y permite una optimización común en las librerías de aprendizaje profundo para realizar el cómputo del gradiente de manera eficiente. Sin embargo, una desventaja que se presenta con el descenso

de gradientes es que mantiene una sola tasa de aprendizaje para todas las actualizaciones de los pesos y no cambia durante el entrenamiento [8]. Si es muy pequeño puede ser demasiado lento el aprendizaje, y si es demasiado grande puede no converger y hacer fluctuar la función de costo cerca de un mínimo local.

El optimizador de ADAM (Adaptive Moment Estimation) [8] es una variante construida a partir del descenso de gradientes. El optimizador es comúnmente usado dado que realiza el cómputo de las tasas de aprendizaje para cada parámetro de la red para aplicar pequeñas o grandes actualizaciones dependiendo de la frecuencia de las características. El optimizador de ADAM almacena un promedio exponencialmente decreciente de los gradientes pasados m_t, similar al *momentum* [8]. El cómputo de los promedios decrecientes de los gradientes pasados y al cuadrado m_t y v_t respectivamente se calculan como:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) {g_t}^2, \end{aligned} \tag{2.19}$$

donde m_t y v_t son estimados del primer momento (la media) y el segundo momento (varianza no centrada) de los gradientes respectivamente, g_t detona el gradiente en un paso del tiempo t. Como m_t y v_t tienden a estar sesgados a cero, especialmente durante las primeras iteraciones, se calculan nuevas estimaciones de primer y segundo momento corregidas por sesgo:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2.20}$$

Por último, las actualizaciones de los parámetros están dada por la función:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \varepsilon}\widehat{m_t} \tag{2.21}$$

Los autores proponen los valores por default de $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10E - 8$.

Diversos resultados empíricos demuestras que ADAM trabaja bien en algoritmos de aprendizaje profundo para la segmentación de imágenes y ha sido comparado favorablemente contra otros métodos de optimización [8].

2.3.2.7. Inicializador de pesos

Escoger los valores adecuados para la inicialización es necesario para un entrenamiento eficiente [8]. Si inicializamos los pesos con valor cero puede llevar a la red a aprender las mismas características durante el entrenamiento, por lo que no se lograría el aprendizaje [8]. Por otro lado, inicializarlos con valores de forma aleatoria sin conocer su distribución puede llevar a dos posibles casos: si son muy pequeños la red presenta un problema de desvanecimiento de gradientes y si son muy grandes se presenta una explosión de gradientes durante la optimización [8].

El inicializador de pesos propuesto por He et al. [61] es una versión mejorada del inicializador de Xavier [8]. Esta una técnica de inicialización de pesos que hace que la varianza de las salidas de una capa sea igual a la varianza de sus entradas, pero con la diferencia que incluye un factor de multiplicación por 2. Los pesos son inicializados tomando en consideración el tamaño de la capa previa lo que resulta en obtener un mínimo global de la función de costo eficientemente [8]. Idealmente el inicializador de He et al. [61] debe ser aplicado para las capas que implementan la función de activación ReLU y los pesos resultantes son valores aleatorios que

difieren en su rango. El inicializador de He et al. [61] genera las muestras de una distribución normal truncada centrada en 0 con:

$$Desviación\ estándar\ (w) = \sqrt{\frac{2}{fan_{in}}}, \tag{2.22}$$

donde w son pesos de la red, y fan_in es el número de unidades de entrada.

2.3.2.8. Aumentación de datos

Como las redes convolucionales profundas tienen un largo número de parámetros, estos tienden a sobre ajustarse con los datos de entrenamiento durante el proceso de aprendizaje. El sobreajuste, ocurre cuando el modelo se desempeña bien con los datos de entrenamiento, pero falla en generalizar bien con nuevos datos nunca vistos [8]. Este problema puede ser identificado cuando la red devuelve valores de rendimiento muy altos para el conjunto de entrenamiento; mientras que los valores obtenidos con un conjunto desconocido son peores y no mejoran después de determinadas épocas. Puede ocurrir que la red aprende demasiado rápido sobre los ejemplos de entrenamiento dependiendo de la complejidad del modelo usado [8].

Los enfoques de regularización ayudan a mitigar el sobreajuste [8]. Un método común es aumentar la cantidad de imágenes de entrenamiento, el cual es conocido como aumentación de datos. Al trabajar con imágenes fácilmente se puede implementar rutinas para generar nuevas muestras aplicando operaciones sobre las imágenes como rotaciones, escalamiento, deformaciones, volteos

horizontales y verticales, efectos espejo, corrimiento de pixeles, etc. De tal manera que cada nueva muestra sea diferente, simulando tener más información (ver. Fig 2.21). Esta técnica ayuda a mejorar la eficacia de la segmentación logrando reducir el sobreajuste.

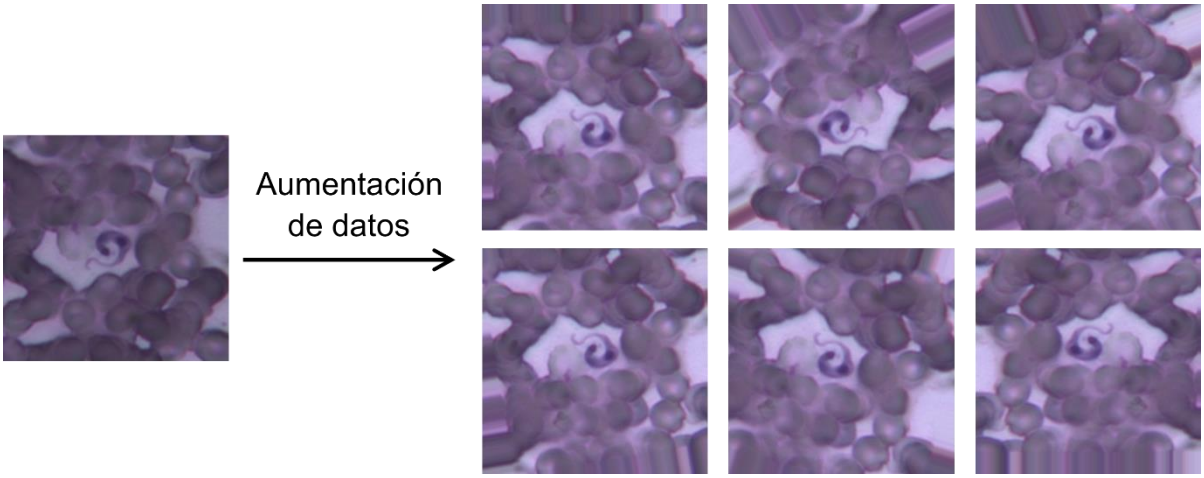


Fig. 2.21. Ejemplo de generación de nuevas muestras aplicando rotaciones, efecto espejo, variaciones de color, y corrimiento de pixeles.

2.3.2.9. Normalización de las entradas

Una práctica común en aprendizaje automático es normalizar las entradas de las capas en una red convolucional para acelerar el entrenamiento de la red. Por ejemplo, al recibir una imagen de entrada, por lo general se escala las intensidades de los pixeles a valores entre 0 y 1. Con esto se logra que los pesos de las capas iniciales estén en una misma escala [8], evitando que se requiere mayores épocas de entrenamiento hasta que se escalen nuevamente con ayuda del optimizador.

Tradicionalmente en una capa de convolución a las salidas se les aplica una función de activación para obtener un mapa de activación (visto previamente en la sección 2.3.2.2), lo cual se puede representar como:

$$h_i(x) = g\big(a_i(x)\big), \tag{2.23}$$

donde a_i es un mapa de características de la capa i, x es un valor perteneciente al mapa de características, $g()$ es una función de activación, y h_i un mapa de activación resultante. El mapa de activación resultante será la entrada de otra capa convolucional, por lo que, si se normaliza o no la primera entrada de una red, los valores de activación posteriores siempre variarán en escala conforme se aumenta la profundidad de la red. Para mantener todos los valores de activación de las capas de activación en una misma escala es necesario normalizar cada mapa de características antes de aplicar la función de activación.

Un método eficaz para normalizar los datos de entrada logrando una convergencia rápida y reduciendo el tiempo de entrenamiento, se conoce como Batch Normalization o normalización por lotes. El nombre de Batch Normalization se debe a que se calcula la media y la desviación estándar usando un lote de imágenes fijo [8]. La normalización se realiza individualmente en cada capa de convolución sobre el mapa de características a^i de la siguiente forma:

$$a_i^{norm} = \frac{a^i - \mu}{\sigma}, \tag{2.24}$$

donde μ es la media, σ es la desviación estándar y a_i^{norm} es el mapa de activación de salida normalizada.

Dependiendo de la construcción de la red, se puede primero optar por normalizar y luego aplicar la función de activación, o primero aplicar la función de activación y luego normalizar [8]. La normalización por lotes también actúa como un método de regularización para evitar el sobreajuste y otorga resistencia a la red para evitar el desvanecimiento de gradientes [8]. El entrenamiento de un modelo se vuelve menos sensible a la selección de los hiper parámetros (como la tasa de aprendizaje) cuando se usa la normalización por lotes [8].

2.3.2.10. Aprendizaje residual

Estudios recientes han sugerido mejorar el aprendizaje simplemente agregando más capas a las redes neuronales [62-64]. Sin embargo, agregar más y más capas puede resultar en el desvanecimiento de gradientes [64]. El problema del desvanecimiento de gradientes ocurre cuando la red no puede predecir bien debido a que los gradientes (durante la optimización) se vuelven más pequeños debido a una alta profundidad de la red, y como resultado el rendimiento es peor [65, 66]. Una forma de resolver este problema es implementando el framework de Aprendizaje Residual Profundo [63]. Las redes residuales o ResNets consisten en arquitecturas modularizadas que apilan bloques de construcción con la misma forma de conexión para alimentar un mapeo residual en lugar de un mapeo subyacente, permitiendo redes sustancialmente más profundas y un aprendizaje mejorado [63]. Estos bloques de construcción son conocidos como Unidades Residuales (RU). La idea detrás de su uso es introducir las funciones residuales y conexiones de salto corto (short-skip-connections) como solución de las dificultades de hacer redes profundas [63, 67]. Una conexión de salto corto es un término para

describir la conexión de la entrada de una Unidad Residual a la salida de una Unidad Residual diferente, hacia adelante omitiendo capar intermedias. Las ResNets aprenden la función residual aditiva $F(\cdot)$ con respecto a $h(x)$, la forma general puede ser representada como:

$$y_n = h(x_n) + F(x_n)$$

$$x_{n+1} = f(y_n), \tag{2.25}$$

donde x_n es el mapa de características de entrada de la n-th Unidad Residual, x_{n+1} es el mapa de características de salida, $F(\cdot)$ es una función residual, $f(y_n)$ es una función de activación posterior, y $h(x_n)$ es un mapeo de identidad representando la función de atajo. En el trabajo presentado por He et al. [68] se propuso los mapeos de identidad para $h(x_n)$ y $f(y_n)$, para propagar la información a la red entera a través de "caminos directos" de una Unidad Residual a cualquier otra Unidad Residual.

Una arquitectura que toma ventaja del aprendizaje residual es la ResUnet [62], una versión modificada del modelo U-Net [57] como propuesta para mejorar la segmentación de imágenes.

3. Metodología

3.1. Modelo propuesto Res2Unet

Inspirados en la arquitectura de la ResUnet [62], la cual toma ventaja del modelo U-Net y el aprendizaje residual, en este trabajo de tesis proponemos una versión modificada para la segmentación semántica del parásito *T. cruzi*. Nuestro modelo propuesto tiene por nombre Res2Unet e introduce las conexiones residuales dobles con el fin de mejorar la propagación de la información para un mejor rendimiento de segmentación. El esquema de segmentación de nuestra propuesta se presenta en la Fig. 3.1.

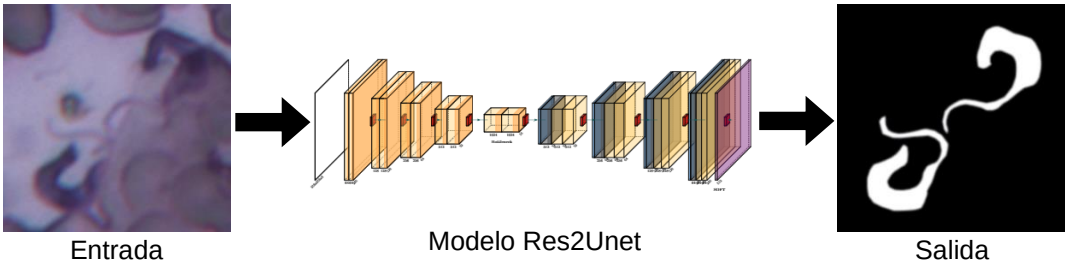


Fig. 3.1. Marco de segmentación del parásito *T. cruzi.*

La Res2Unet consiste en una arquitectura modular de 9 niveles construida con Unidades Residuales. La Res2Unet cuenta con tres partes: un camino codificador, un puente y un camino decodificador. El puente sirve como una conexión entre los

caminos codificador y decodificador. El número de filtros de convolución en las capas del codificador y puente contiene los valores crecientes de 64, 128, 256, 512 y 1024; y el número de filtros de convolución decrementa en las capas del decodificador con los valores de 512, 256, 128 y 64. El modelo no aplica una operación de Pooling y la única forma de disminuir la resolución de las características es controlado usando strides en las capas de convolución después de la primera Unidad Residual. En las capas del decodificador se implementa las convoluciones transpuestas para generar un mapa de segmentación de alta resolución.

La Res2Unet mantiene las conexiones de salto largo (long-skip-connections) entre las capas del codificador y decodificador, y las conexiones de salto corto (conexiones residuales) entre las Unidades Residuales para un entrenamiento fácil y una retroalimentación precisa de la segmentación [57, 62]. Una conexión de salto largo realiza una concatenación de las resoluciones de características correspondientes en el camino codificador y decodificador [57], y las conexiones de salto corto realizan una adición de las características entre las Unidades Residuales (previamente descrito en la sección 2.3.2.10).

Nuestro modelo agrega una conexión residual doble entre dos Unidades Residuales consecutivas únicamente en los caminos codificador y decodificador. Al final de la red se aplica una capa de convolución 1 × 1 con un filtro y una función de activación sigmoidal para obtener un mapa de segmentación binario.

La Res2Unet recibe una imagen resolución 256 × 256 y devuelve una imagen segmentada de salida con la misma resolución implementando zero-padding en todas las capas de convolución.

La arquitectura de la Res2Unet se muestra en la Fig. 3.2. En la Tabla 3.1 se detalla la configuración de la red con los tamaños de salida de las capas. El modelo Res2Unet implementa el contenido de las Unidades Residuales presentadas en la Fig. 3.3, originalmente propuesto por He et al. [68] para alcanzar una convergencia rápida de entrenamiento.

Como se mencionó en la sección 2.3.2.10, una conexión residual en una ResNet solo suma la entrada con la salida de una Unidad Residual. En una conexión residual doble en vez de solo adición, promediamos la entrada de una Unidad Residual con la salida de una segunda Unidad Residual (lo que llamamos como Unidad Res2Unet) como se muestra en la Fig. 3.2. Este pequeño cambio es de hecho muy relevante ya que al promediar somos consistentes con algo conocido como el método de Heun para resolver numéricamente una ecuación diferencial ordinaria. En la siguiente sección, daremos soporte matemático a la arquitectura Res2UNet.

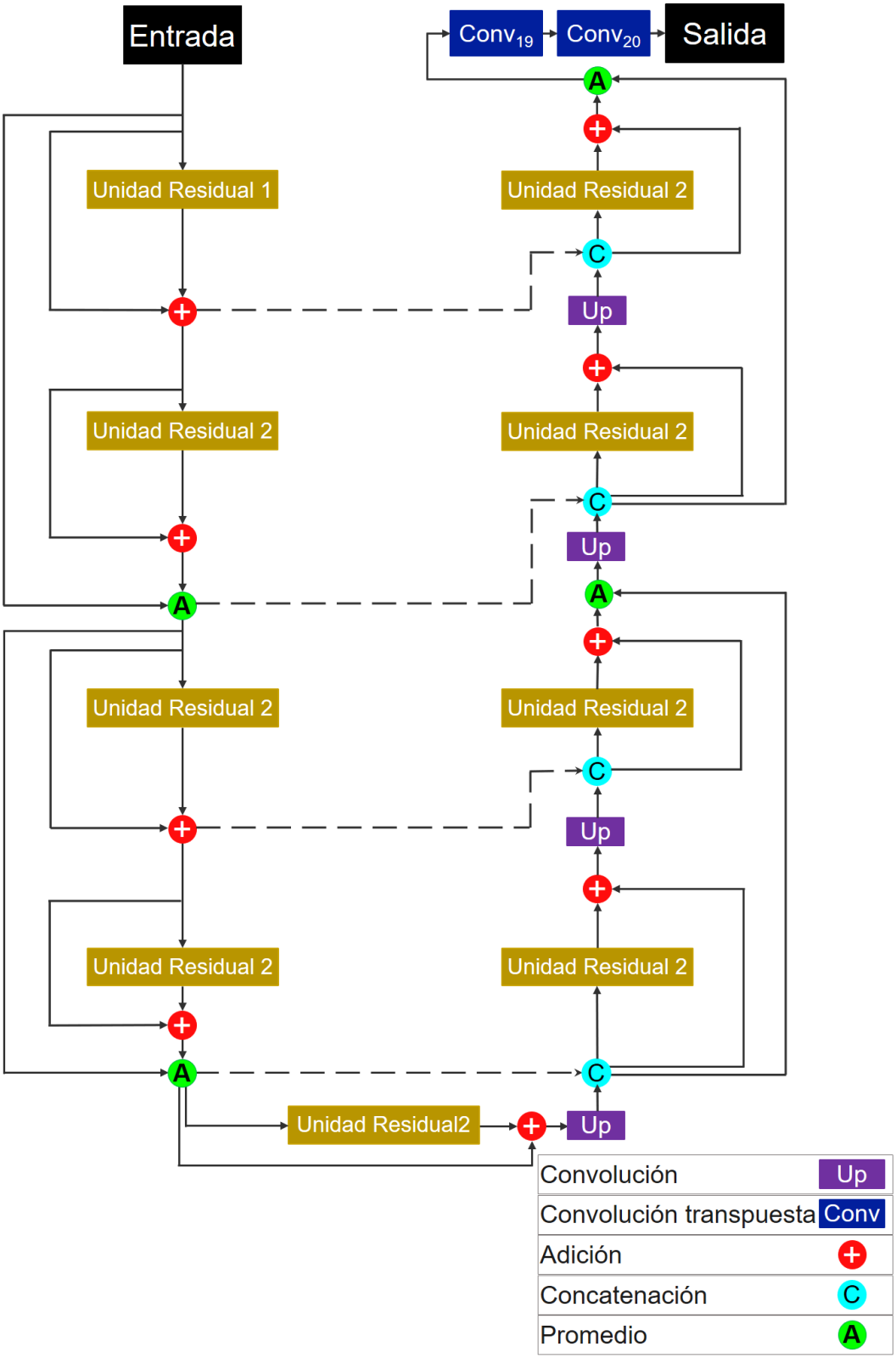


Fig. 3.2. Nuestro modelo de segmentación Res2Unet.

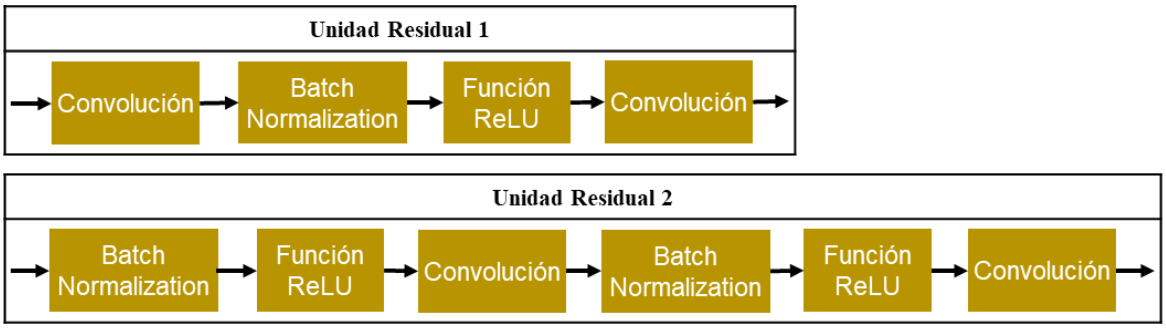


Fig. 3.3. Unidades Residuales del modelo Res2Unet: (a) variante 1, y (b) variante 2.

	Nivel	Capa de Convolución	Filtro	Stride	Tamaño de salida
Entrada					256 × 256 × 3
Codificador	Nivel 1	Conv 1	3 × 3 /64	1	256 × 256 × 64
		Conv 2	3 × 3 /64	1	256 × 256 × 64
	Nivel 2	Conv 3	3 × 3 /128	2	128 × 128 × 128
		Conv 4	3 × 3 /128	1	128 × 128 × 128
	Nivel 3	Conv 5	3 × 3 /256	2	64 × 64 × 256
		Conv 6	3 × 3 /256	1	64 × 64 × 256
	Nivel 4	Conv 7	3 × 3 /512	2	32 × 32 × 512
		Conv 8	3 × 3 /512	1	32 × 32 × 512
Puente	Nivel 5	Conv 9	3 × 3 /1024	2	16 × 16 × 1024
		Conv 10	3 × 3 /1024	1	16 × 16 × 1024
Decodificador	Nivel 6	Conv 11	3 × 3 /512	1	32 × 32 × 512
		Conv 12	3 × 3 /512	1	32 × 32 × 512
	Nivel 7	Conv 13	3 × 3 /256	1	64 × 64 × 256
		Conv 14	3 × 3 /256	1	64 × 64 × 256
	Nivel 8	Conv 15	3 × 3 /128	1	128 × 128 × 128
		Conv 16	3 × 3 /128	1	128 × 128 × 128
	Nivel 9	Conv 17	3 × 3 /64	1	256 × 256 × 64
		Conv 18	3 × 3 /64	1	256 × 256 × 64
Salida 2		Conv 19	1 × 1 /2	1	256 × 256 × 2
Salida 1		Conv 20	1 × 1 /1	1	256 × 256 × 1

Tabla 3.1. Configuración de la Res2Unet.

3.2. Análisis matemático de la arquitectura Res2Unet

Las ResNests [63] pueden ser vistas como un sistema obedeciendo la siguiente ecuación diferencial [69, 70]:

$$\dot{x}(t) = f\big(t, x(t)\big),\ x(t_0) = x_0 \tag{3.1}$$

Quizás el método numérico más popular para resolver (3.1) es el método explícito de Euler dado por:

$$x_{i+1} = x_i + hf(t_i, x_i) \tag{3.2}$$

donde h es un tamaño de paso, y $a < t_i < b$ es un intervalo de integración tal que $t_{i+1} = t_i + h$ produciendo la cuadrícula de $n + 1$ puntos $a = t_0 < t_1 < \ldots < t_n = b$.

El error de truncamiento local del método de Euler [71] es de orden h^2 y dado por:

$$e_i = \frac{h^2}{2}\ddot{x}(c), \tag{3.3}$$

para alguna c (desconocida) satisfaciendo $t_i < c < t_{i+1}$. Mientras e_i puede mantenerse bastante limitado por h, no es tan fácil por su error de truncamiento global g_i definido como [71]:

$$g_i \leq \frac{Ch^k}{L}(e^{L(t_i - a)} - 1) \tag{3.4}$$

para alguna constante $C, k \geq 0$ y L la constante de Lipschitz de f. El problema es que durante largos períodos de integración (b grande), el error global crece exponencialmente y por lo tanto para mantenerlo acotado, h tiene que ser seleccionado muy pequeño.

Con respecto a la arquitectura de una ResNet donde cada Unidad Residual se asemeja a un paso del método de Euler [72], cada t_i puede verse como una capa de la red y, por lo tanto, una h pequeña significa muchas capas o una red muy profunda.

Esto de ninguna manera es óptimo, ya que está documentado los inconvenientes de tener redes muy profundas [69, 73]. Por lo tanto, se pueden usar arquitecturas más eficientes para reducir el error global.

En este trabajo, elegimos usar una arquitectura basada en el método de Heun para resolver ecuaciones diferenciales ordinarias. Este método también se conoce como el método de Euler mejorado o modificado [74]. El método de Heun también puede verse como un método Runge-Kutta de segundo orden de dos etapas. El método de Heun reduce el error global cuadráticamente [71], por lo tanto, permite tener menos capas de la red residual. El error global es relevante ya que afecta directamente en la precisión de las predicciones de la red.

El método de Heun equivale a usar la siguiente iteración:

$$x_{i+1} = x_i + \frac{h}{2}\Big(f(x_i) + f\big(x_i + hf(x_i)\big)\Big) \tag{3.5}$$

Analizando la Eq. (3.5), el método de Heun mejora el método de Euler al elegir una pendiente más precisa calculada promediando las pendientes de Euler en t_i y t_{i+1}.

Una Unidad Res2Unet es la concatenación de dos Unidades Residuales con retroalimentación hacia delante de la entrada x_i, y su promedio con la salida de la segunda Unidad Residual (ver Fig. 3.4).

Sea x_i, x_{i+1} la entrada y salida de la Unidad Res2Unet respectivamente. Sean también x_1, x_2 las salidas de la primera y segunda Unidades Residuales respectivamente, entonces:

$$x_1 = x_i + hf(x_i)$$

$$x_2 = x_1 + hf(x_1) \tag{3.6}$$

$$= x_i + hf(x_i) + hf\big(x_i + hf(x_i)\big)$$

y,

$$x_{i+1} = \frac{1}{2}(x_i + x_2)$$

$$= \frac{1}{2}\Big(x_i + x_i + hf(x_i) + hf\big(x_i + hf(x_i)\big)\Big) \tag{3.7}$$

$$= x_i + \frac{h}{2}\Big(f(x_i) + f\big(x_i + hf(x_i)\big)\Big)$$

Exactamente el mismo resultado como en (3.5). El esquema numérico puede representarse usando un diagrama de bloques de la Unidad Res2UNet en la Fig. 3.4.

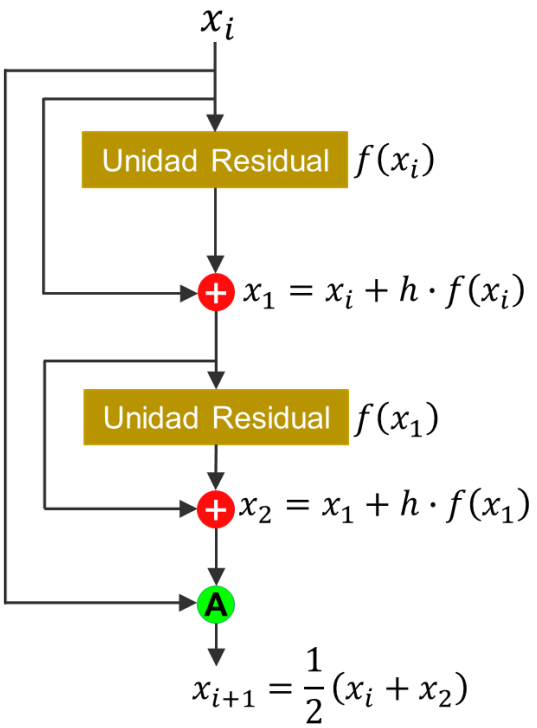


Fig. 3.4. Unidad Res2Unet basada en el método de Heun para mejorar la exactitud.

3.3. Base de datos

La base de datos original consiste en 974 imágenes en el espacio de color RGB de tamaño 2560 × 1920 pixeles. Las imágenes fueron tomadas usando un microscopio con muestras de sangre extraídas de ratones infectados con la enfermedad de Chagas. Las muestras de sangre fueron teñidas para una mejor visualización de la morfología de los parásitos. Después, un conjunto de 1040 sub imágenes de tamaño 256 x 256 fueron recortadas de la base de datos original. De este conjunto de sub imágenes, 978 imágenes contienen parásitos y 62 imágenes no contienen parásitos. Separamos el conjunto de imágenes en tres conjuntos, el conjunto de entrenamiento con 626 imágenes, el conjunto de validación con 207 imágenes, y el conjunto de prueba con las 207 imágenes restantes. Con la ayuda de expertos, manualmente asignamos a los pixeles en todas las imágenes su etiqueta de clase correspondiente (clase principal y fondo) para crear el conjunto de imágenes del ground truth. En la Fig. 3.5, podemos ver un ejemplo de una imagen de muestra de sangre y su correspondiente ground truth.

En una imagen de muestra de sangre de tamaño 256 × 256 pixeles, aproximadamente 8% de los pixeles corresponden a la clase principal o parásito, y el 92% de los pixeles a la clase de fondo. Por lo que el tamaño predefinido 256 × 256 de las imágenes fue seleccionado para prevenir que nuestra red de segmentación prediga principalmente la clase dominante [30].

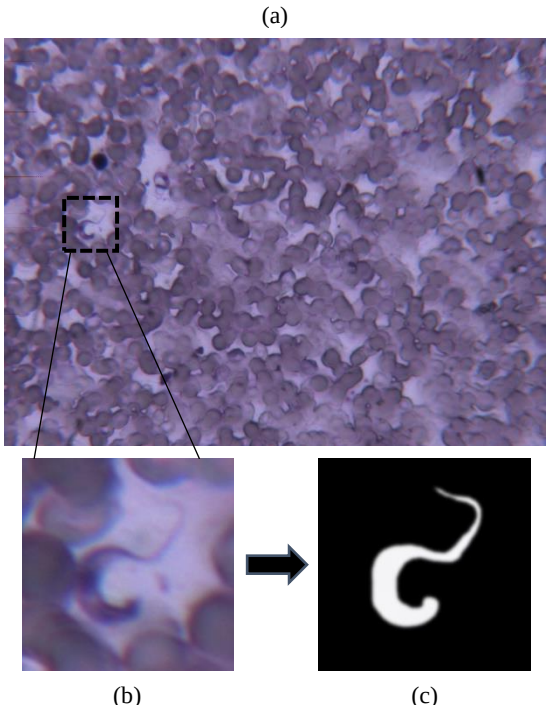


Fig. 3.5. Generación del conjunto de datos de imágenes. (a) imagen original, (b) imagen recortada y (c) segmentación manual.

Incrementamos el número de imágenes del conjunto de entrenamiento aplicando aumentación de datos de forma local hasta obtener un total de 3824 imágenes. La aumentación de datos que se implementó extrae hasta 9 imágenes de tamaño 256 × 256 alrededor de cada parásito del conjunto de imágenes de entrenamiento mediante un corrimiento de pixeles predeterminado como se muestra en la Fig. 3.6. También generamos las imágenes del ground truth correspondientes a cada nueva imagen de entrenamiento aumentada. Para el conjunto de validación y pruebas no aplicamos aumentación de datos local.

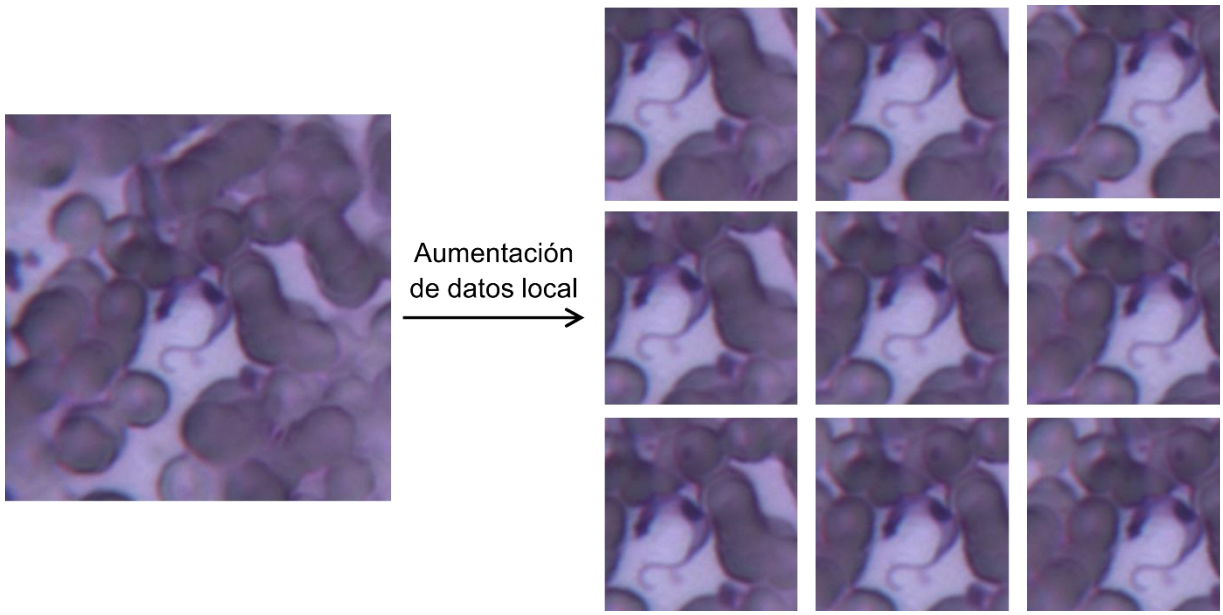


Fig. 3.6. Aumentación de datos local para incrementar el conjunto de entrenamiento.

3.4. Detalles de la implementación

Evaluamos la arquitectura de la Res2Unet adoptando una metodología de validación cruzada con los conjuntos de entrenamiento, validación y prueba. Durante el entrenamiento la red utiliza el conjunto de entrenamiento para aprender los pesos adecuados, el set de validación es usado para ajustar los hiper parámetros del modelo después del entrenamiento (aquellos parámetros que necesitan ser modificados para mejorar la eficacia de la segmentación como la tasa de aprendizaje, el tamaño del lote, número de filtros por capas, valores de la función de costo, etc.), y el set de pruebas es usado al final de la experimentación para obtener el desempeño final de la red mediante el cálculo de las métricas de rendimiento.

Comparamos el rendimiento del modelo Res2Unet propuesto con las arquitecturas U-Net [57] y ResUnet [62]. Las arquitecturas fueron implementadas usando el framework de Keras en la plataforma de Google CoLab [75].

La aumentación de datos se implementó en tiempo de entrenamiento aplicando operaciones de rotación, zooming, efecto espejo, volteos horizontal y vertical, sobre las imágenes del conjunto de entrenamiento. El generador de imágenes de Keras devuelve la aumentación de datos de forma aleatoria, por lo que no se podrá obtener dos épocas de entrenamiento con un conjunto de imágenes idéntico. El parámetro de pasos por época permite controlar la cantidad de generación de imágenes aumentadas. La fórmula para obtener el valor parámetro está dada por:

$$Pasos\ por\ época\ = \frac{G}{t_n} \tag{3.8}$$

donde G es el número de imágenes del conjunto de entrenamiento y t_n el tamaño del mini lote de imágenes por época.

Iniciamos el entrenamiento con un tamaño de mini lote de dos, un valor de paso por época de 1912 y estableciendo ADAM como algoritmo optimizador. Probamos con diferentes tasas de aprendizaje en el intervalo de 5E-5 a 1E-4 para lograr una convergencia óptima. Estos hiper parámetros fueron manualmente ajustados durante el entrenamiento. Entrenamos los modelos de 30 a 50 épocas hasta que el rendimiento del conjunto de validación alcance el valor más alto.

Cada modelo de segmentación se entrenará con las funciones de costo BCE, WBCE y AC para comparar los resultados obtenidos. Para el modelo Res2Unet, encontramos que la función de costo AC provee los mejores resultados preservando la forma del parásito y devolviendo una segmentación limpia con bordes marcados.

3.4.1. Métricas de rendimiento

Para evaluar el rendimiento de segmentación, seleccionamos el puntaje del coeficiente de Dice (Dice Coefficient, DC) [76], el puntaje de la intersección sobre la unión (IoU) [77], los valores de precisión y Recall [77].

El puntaje del coeficiente de Dice (DC) sirve como una métrica para comparar la similitud de pixeles entre una predicción P y la segmentación correcta T, donde un valor del puntaje DC cercano a uno representa una correspondencia perfecta. El puntaje DC se calcula como:

$$Coeficiente\ de\ Dice = \frac{2*|P \cap T|}{|P|+|T|} \tag{3.9}$$

La métrica de IoU mide la intersección sobre la unión de los pixeles clasificados por cada clase y devuelve su promedio [77]. En comparación con el puntaje del coeficiente de Dice, IoU penaliza más las clasificaciones erróneas. El cálculo de IoU se realiza en términos de los Verdadero Positivos (TP), Falsos positivos (FP), y Falsos Negativos (FN), de la siguiente forma:

$$IoU = \frac{TP}{TP + FP + FN} \tag{3.10}$$

cuya notación se detalla en la Tabla 3.2.

		Clase predicha	
		Positiva	Negativa
Clase verdadera	Positiva	Verdaderos Positivos (TP)	Falsos Negativos (FN)
	Negativa	Falsos Positivos (FP)	Verdaderos Negativos (TN)

Tabla 3.2 Matriz de confusión de clase a nivel de píxel.

La Precisión describe la pureza de todas las predicciones de pixeles positivas relativas al ground truth [77] y se define como:

$$Precisión = \frac{TP}{TP + FP} \tag{3.11}$$

El Recall describe la completitud de las predicciones de pixeles positivas relativas al ground truth [77] y se define como:

$$Recall = \frac{TP}{TP + FN} \tag{3.12}$$

En la práctica si obtenemos un puntaje elevado de DC o de IoU en correlación con un valor alto de Precisión y de Recall, se espera obtener una imagen correctamente segmentada sin una clasificación errónea de píxeles y con la morfología completa del parásito *T. cruzi*.

3.4.2. Comparación con otros métodos

Comparamos el rendimiento de nuestro modelo Res2Unet con dos métodos tradicionales de aprendizaje automático presentados previamente para la segmentación del parásito *T. cruzi* en la sección 1.1.2 por Soberanis-Mukul [7]. El

primer método consiste en un clasificador Gaussiano el cual obtuvo el mejor rendimiento de segmentación; el segundo método es un clasificador SVM con implementación de super pixeles, el cual obtuvo el segundo mejor rendimiento de segmentación en [7]. Ambos métodos fueron ejecutados y entrenados en C++ usando nuestro conjunto de validación cruzada. Para el caso del clasificador SVM con implementación de super pixeles distintos tamaños de super pixeles (SP) fueron probados para obtener el rendimiento adecuado.

4. Experimentación y resultados

4.1. Experimentación de la Res2Unet

Para mostrar la efectividad del modelo Res2Unet, primero realizamos dos experimentos para ajustar la mejor configuración de la red. El modelo Res2Unet toma alrededor de 10 horas ser entrenado y tiene cerca de 300 megabytes de información.

4.1.1. Selección del valor lambda de AC

Entrenamos el modelo Res2Unet usando diferentes valores lambda de la función de costo AC para elegir el que mejor rendimiento devuelva con el conjunto de validación. En la Fig. 4.1 se muestran los resultados obtenidos en una gráfica.

Un valor de lambda entre 3 y 10 obtiene un rendimiento muy similar del puntaje DC, por lo que seleccionamos $\lambda = 5$ como el valor por defecto para los siguientes experimentos con nuestro modelo.

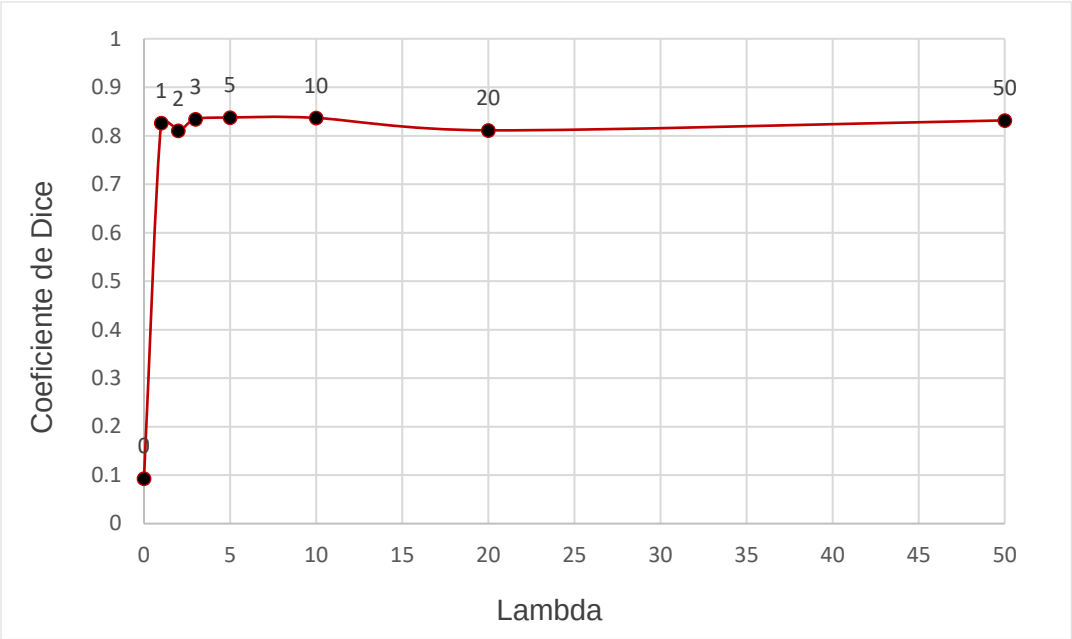


Fig. 4.1. Rendimiento del puntaje DC para diferentes entrenamientos con λ.

4.1.2. Selección del valor *h* para el método de Heun

Entrenamos el modelo Res2Unet para seleccionar el valor h del método de Heun presentado anteriormente en la sección 3.2. Nuestro objetivo es encontrar la mejor actualización de las capas en una Unidad Res2Unet. Las métricas de rendimiento obtenidas para diferentes entrenamientos con distintos valores h se presentan en la Tabla 4.1.

Entrenamiento con *h*	Puntaje DC	Función de costo AC (λ=5)	IoU	Precisión	Recall
1	0.8374	**7063.7765**	0.7347	0.7869	**0.8075**
0.5	**0.8401**	7141.4490	**0.7364**	**0.8050**	0.7852
0.25	0.8391	7192.7894	0.7341	0.7888	0.7981
0.1	0.8345	7400.0915	0.7290	0.7853	0.7963

Tabla 4.1. Métricas de rendimiento de la Res2Unet para el conjunto de validación.

Analizando las métricas obtenidas en la Tabla 4.1, podemos concluir que los resultados son muy competitivos, aunque para $h = 0.5$ logramos obtener el mejor

rendimiento del puntaje DC. Para una selección óptima del valor h, realizamos también un análisis cualitativo de las predicciones devueltas por los modelos entrenados. En la Fig. 4.2 se observan algunas imágenes segmentadas del conjunto de validación.

Fig. 4.2. Imágenes segmentadas obtenidas para diferentes valores *h*.

Seleccionamos el entrenamiento con $h = 0.5$ como el principal rendimiento de segmentación de nuestro modelo Res2Unet debido al puntaje DC más alto obtenido en la Tabla 4.1 y porque obtenemos las imágenes segmentadas del parásito *T. cruzi* con menos ruido en la Fig. 4.2.

Las curvas de entrenamiento (para $h = 0.5$) del puntaje DC se presentan en la Fig. 4.3, y en la Fig. 4.4 se presentan las curvas de entrenamiento obtenidas de la función de costo AC.

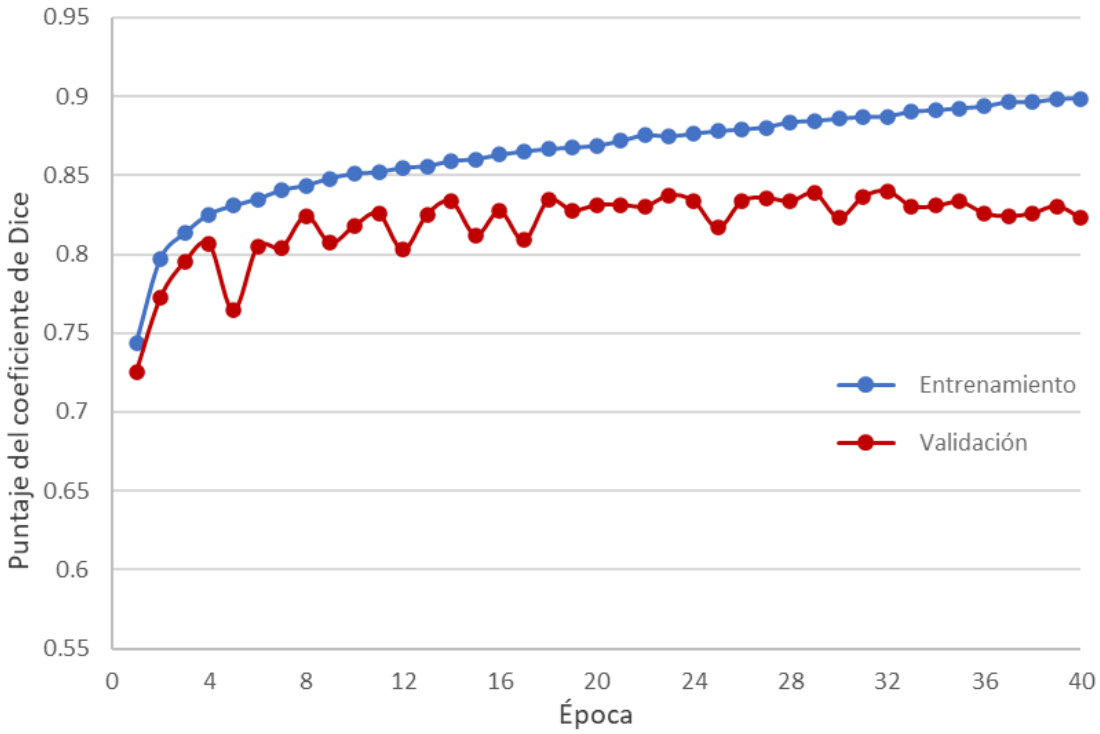


Fig. 4.3. Curvas de entrenamiento (Res2Unet con $h = 0.5$) del puntaje DC.

Después de analizar el comportamiento de las curvas de entrenamiento en las Figs. 4.3 y 4.4, podemos confirmar que el entrenamiento es estable y converge rápidamente usando nuestra arquitectura y el contenido de las Unidad Residuales propuesto en [68].

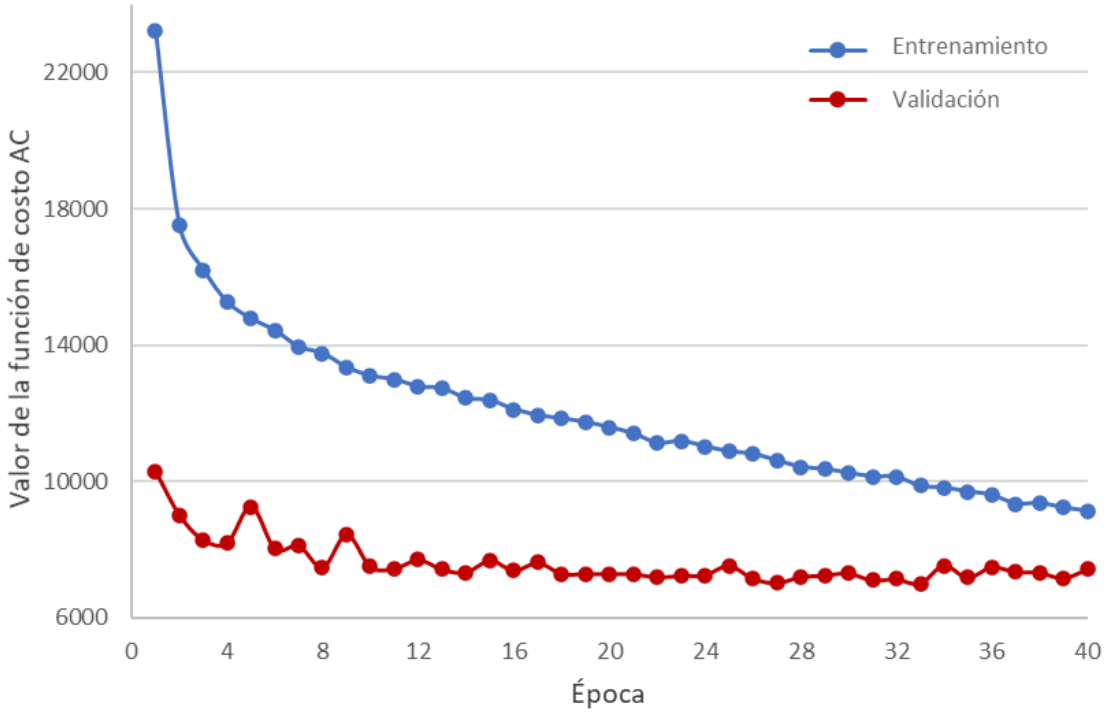


Fig. 4.4. Curvas de entrenamiento (Res2Unet con $h = 0.5$) de la función AC.

Se destaca que el puntaje DC a partir de la época número 6 de entrenamiento empieza notablemente a converger para el conjunto de validación, fluctuando muy poco y alcanzando el valor más alto del puntaje DC en la época número 32 (Fig. 4.3). En cuanto a la gráfica de la función de costo (Fig. 4.4), el costo hace una disminución drástica en las primeras 10 épocas hasta mantenerse estable en las siguientes épocas para el conjunto de validación. Es importante destacar que el modelo fue probado con distintas tasas de aprendizaje y alcanzó los mejores resultados con $\eta = 8E - 05$, por lo que disminuyendo aún más su valor no mejoraba el rendimiento y sólo hacía más lento el aprendizaje. De hecho, cuando $\eta < 7E - 5$ el modelo no logra el aprendizaje después de un número determinado de épocas (no hay cambios en los parámetros).

4.2. Resultados finales de rendimiento

Después de experimentar con el modelo Res2Unet y la función de costo AC, se realizó los entrenamientos de la Res2Unet con las funciones de costo BCE y WBCE (usando los parámetros $\alpha = 0.2$ y $\beta = 0.6$) para una comparación completa de la segmentación del parásito *T. cruzi* contra los modelos U-Net y ResUnet. También comparamos nuestros resultados de segmentación contra los clasificadores Gaussiano y SVM. Las métricas de rendimiento que se obtuvieron para el conjunto de prueba se presentan en la Tabla 4.2.

Modelo	Tasa de aprendizaje	Valor de costo	Puntaje DC	IoU	Precisión	Recall
U-NET+BCE	4.00E-05	0.0473	0.73	0.59	0.80	0.80
U-NET+WBCE	4.00E-05	0.0156	0.77	0.65	0.72	**0.87**
U-NET+AC	4.00E-05	23014.23	0.06	0.06	0.0	0.0
ResUnet+BCE	5.00E-05	0.0525	0.75	0.63	0.78	0.79
ResUnet+WBCE	5.00E-05	0.0209	0.76	0.64	0.73	0.84
ResUnet+AC	5.00E-05	11204.20	0.82	0.71	**0.81**	0.78
Res2Unet+BCE	1.00E-04	0.0522	0.76	0.65	**0.81**	0.78
Res2Unet+WBCE	1.00E-04	0.0212	0.78	0.67	0.76	0.82
Res2Unet+AC	8.00E-05	6655.38	**0.84**	**0.73**	0.80	0.79
Gaussiano	**---**	**---**	0.22	0.12	0.13	0.80
SVM (SP=50)	**---**	**---**	0.13	0.07	0.07	**0.94**

Tabla 4.2. Resultados de segmentación para el conjunto de pruebas.

Como se muestra en la Tabla 4.2, la Res2Unet entrenada con la función de costo AC obtiene los puntajes DC y IoU más altos junto con valores balanceados de Precisión y Recall. Esto podría ser resultado de una retroalimentación más precisa

de las capas en una Unidad Res2Unet devolviendo menos pixeles clasificados erróneamente. En la Fig. 4.5 se muestran algunas imágenes segmentadas devueltas por los modelos de segmentación de la Tabla 4.2 para realizar un análisis cualitativo.

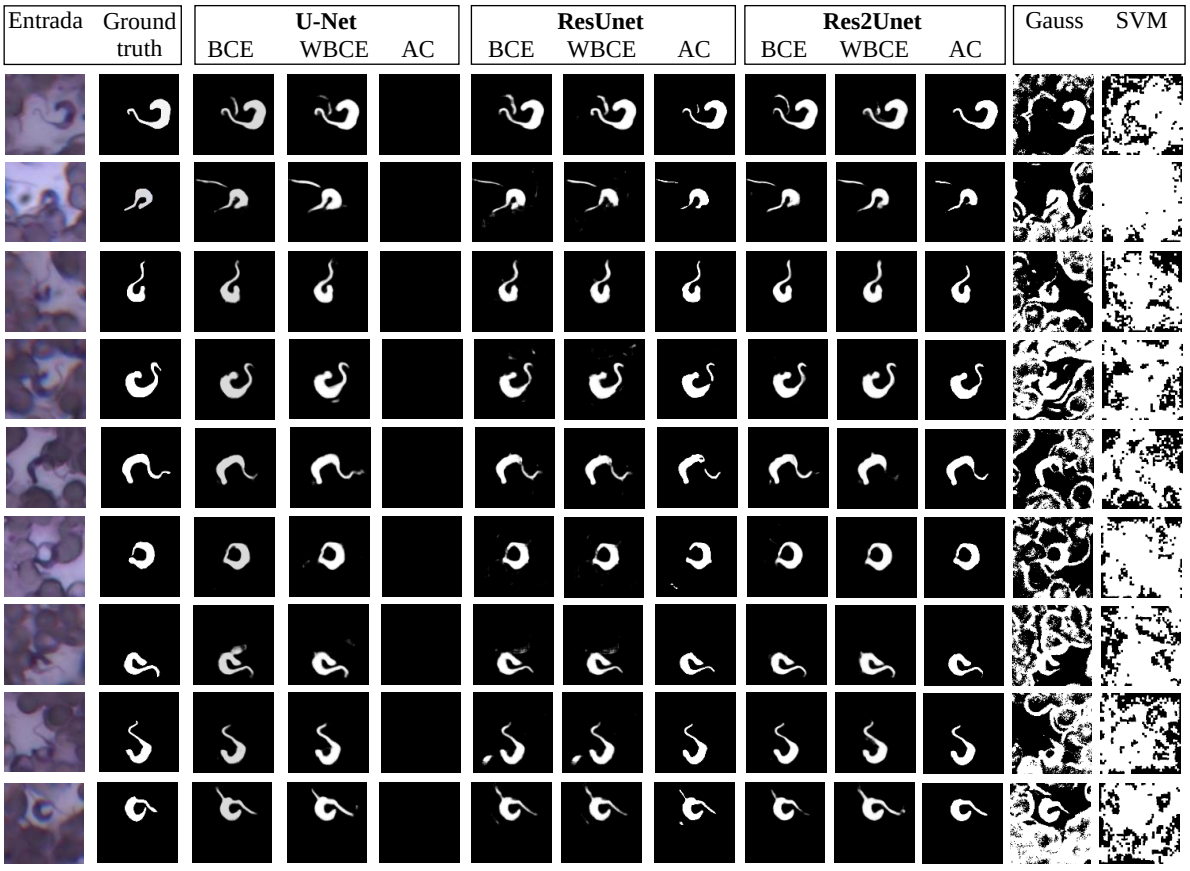


Fig. 4.5. Imágenes segmentadas de los modelos U-Net, ResUnet, y Res2Unet, clasificadores Gaussiano y SVM.

Es importante destacar que el entrenamiento **Res2Unet+AC** predice con más exactitud los bordes de los parásitos *T. cruzi* en comparación con los entrenamientos usando las funciones de costo BCE y WBCE. Otro beneficio importante de nuestro modelo es que no se necesita de un paso de procesamiento posterior para obtener una imagen segmentada libre de ruido; esto es debido a los

valores de probabilidad altos que el modelo Res2Unet devuelve para realizar las predicciones de los pixeles de la imagen y como resultado, se refleja en un valor elevado del puntaje DC y del IoU. Los modelos U-Net y ResUnet devuelven pixeles en escala de grises en los bordes de los parásitos, requiriendo de un paso de procesamiento posterior (como un método de umbral) para refinar los resultados de la segmentación. Por otro lado, los clasificadores Gaussiano y SVM son los que desempeñan peor devolviendo demasiados pixeles falsos positivos, haciendo muy complicado poder distinguir a un parásito. Esto podría ser resultado de realizar las predicciones con características basadas principalmente en el color. Como se notó en la Tabla 4.2 y en la Fig. 4.5, el modelo U-Net no logra el aprendizaje con la función de costo AC, lo cual podría ser resultado de un problema del desvanecimiento de gradientes ya que el modelo U-Net no implementa el aprendizaje residual.

Para un análisis cualitativo final, evaluamos una imagen de muestra de sangre de tamaño 2560 × 1920 pixeles (perteneciente a la base de datos original) para comparar las salidas de los modelos U-Net+WBCE, ResUnet+AC, **Res2Unet+AC**, clasificadores Gaussiano y SVM. De esta imagen extraemos un total de 266 sub imágenes de tamaño 256 × 256 pixeles con una superposición de 128 pixeles entre cada sub imagen, para tener una cobertura completa de las regiones de pixeles donde existieran parásitos. Alimentamos los modelos con el conjunto de 266 imágenes para posteriormente reconstruir la imagen segmentada de tamaño 2560 × 1920 pixeles a partir de las 266 predicciones. Aplicamos un método umbralización (con umbral $T = 0.5$) para descartar los pixeles en escala de grises

para una óptima reconstrucción de la imagen segmentada. La idea de este análisis es corroborar cual modelo genera la menor cantidad de pixeles falsamente clasificados como clase principal (falsos positivos). En la Fig. 4.6 podemos observar las imágenes segmentadas devueltas por los modelos.

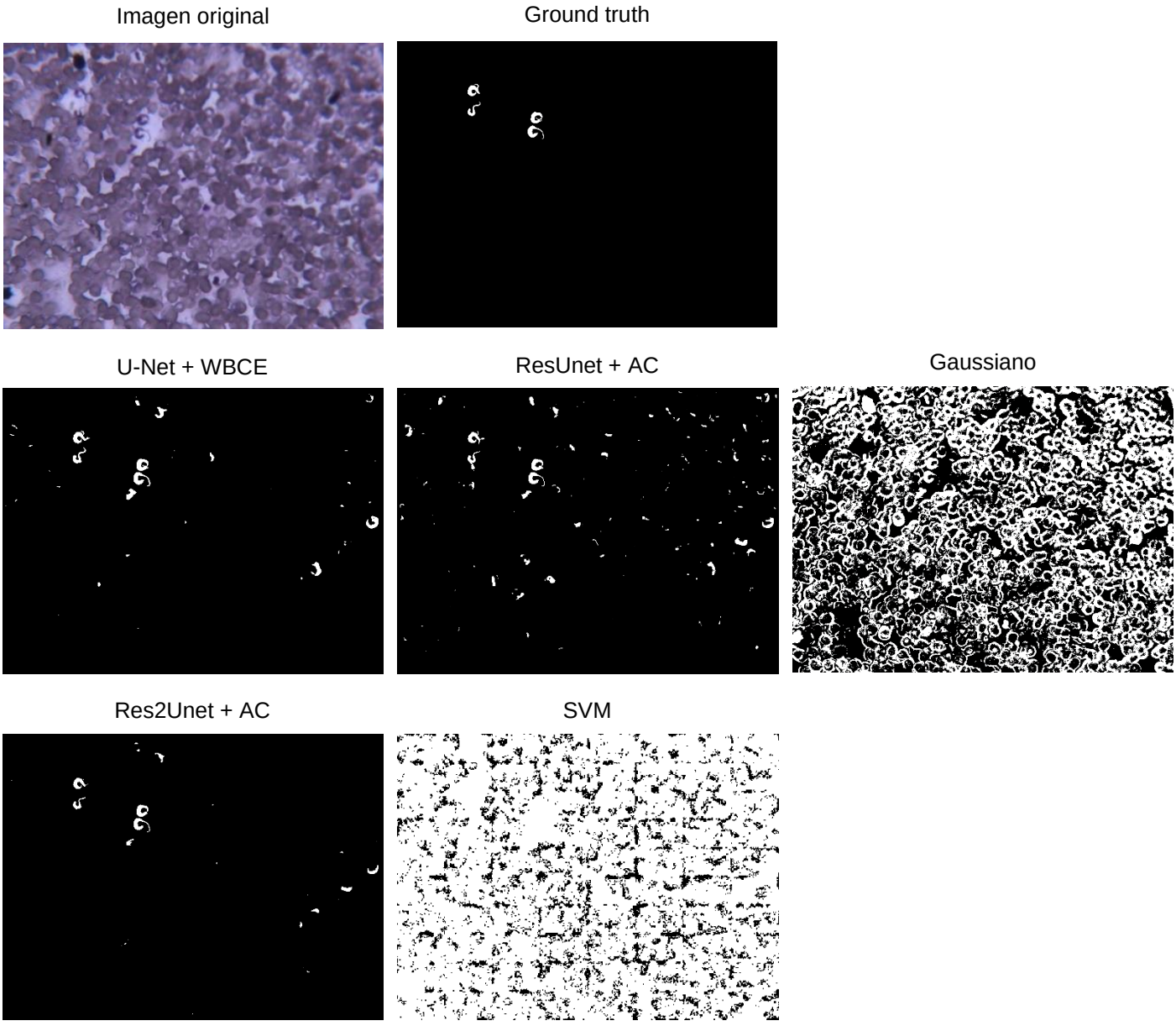


Fig. 4.6. Análisis cualitativo final para la segmentación del parásito *T. cruzi*.

Los modelos ResUnet y U-Net se desempeñan bien, pero incluyen grandes regiones de pixeles falsamente clasificados en la Fig. 4.6. Los clasificadores Gaussiano y SVM no son métodos ideales para la segmentación debido a que devuelven demasiados pixeles falsos positivos resultando impráctico y difícil de diferenciar visualmente a un parásito. Nuestro modelo **Res2Unet+AC** propuesto devuelve la imagen segmentada más limpia y una segmentación correcta de los parásitos en la Fig. 4.6. Por lo tanto, ya que obtuvimos las métricas de rendimiento más altas en la Tabla 4.2, la **Res2Unet+AC** es seleccionada como el mejor modelo para la segmentación de los parásitos *T. cruzi* con nuestra base de datos de imágenes.

5. Conclusiones

En este trabajo proponemos una arquitectura basada en FCNN, a la cual nombramos Res2Unet para la segmentación automática del parásito *Trypanasoma cruzi* en imágenes de muestras de sangre. Nuestro modelo está basado en las metodologías del aprendizaje residual y del modelo U-Net, y propone las conexiones residuales dobles y una arquitectura basada en el método de Heun para obtener un alto rendimiento de segmentación. Con una imagen correctamente segmentada del parásito *T. cruzi* los técnicos podrán fácilmente visualizar su morfología, distinguirlo entre otros componentes y recolectar información para uso posterior.

Obtuvimos el mejor rendimiento de segmentación con la función de costo de Contornos Activos, la cual está diseñada para conservar la morfología de los objetos en imágenes médicas, resultando mejor que las funciones de costo más comunes basadas en la similitud de los pixeles. Esto afirma la importancia de la elección de la función de costo de acuerdo con el modelo de segmentación implementado y el problema de segmentación que se esté abordando dado que en la segmentación de imágenes médicas se requiere una alta precisión para asistir a los expertos.

Los trabajos reportados hasta ahora para la segmentación del parásito causante de la enfermedad de Chagas implementan algoritmos tradicionales de aprendizaje automático que se entrenan con características que los autores proponen después de analizar sus imágenes para el entrenamiento. Si bien, resulta conveniente definir qué técnica utilizar para la extracción de características, las FCNN aprenden a extraer automáticamente las características más importantes y complejas, y las procesan para obtener las predicciones de forma más eficiente. En comparación nuestro modelo Res2Unet obtuvo un rendimiento de segmentación superior a los métodos del estado del arte como el clasificador de Gauss y Support Vector Machine con super pixeles.

Las tres contribuciones finales de este trabajo incluyen (1) la creación de una base de datos de imágenes extensa del parásito *T. cruzi* para el entrenamiento de cualquier clasificador de aprendizaje supervisado, (2) una nueva arquitectura de aprendizaje profundo Res2Unet para mejorar la segmentación de imágenes biomédicas, y (3) el aporte del modelo entrenado Res2Unet+AC para asistir a los expertos en la segmentación del parásito *T. cruzi* en imágenes digitales de muestras de sangre.

5.1. Trabajo a futuro

Como trabajo a futuro se propone experimentar con nuevas arquitecturas basadas en aprendizaje profundo para la segmentación de imágenes como la conocida DenseUnet que propone el paso de información entre capas usando operaciones de concatenación en vez de adición (como en una ResNet) para obtener

características de alto nivel. Por otro lado, se recomienda ampliar la base de datos con imágenes con una mayor gama de condiciones como iluminación, enfoque y color, dado que la calidad de las imágenes capturadas de los frotis de sangre depende de la experiencia del técnico para teñir las muestras y estas pueden variar entre expertos.

Como paso a futuro, se plantea la implementación de un sistema completo de hardware y software para adaptar una computadora con un microscopio equipado con una cámara para realizar las capturas y segmentación de forma automática en laboratorio. Un sistema de dos pasos podría refinar la segmentación obtenida con un modelo entrenado usando alguna otra heurística.

6. Referencias

1. W. H. Organization Homepage, Chagas disease American Trypanosomiasis, https://www.who.int/en/news-room/fact-sheets/detail/chagas-disease-(american-trypanosomiasis), last accessed on 2019-04-02.
2. Guía para la atención del paciente infectado con *Trypanosoma Cruzi* (Enfermedad de Chagas). Centro Nacional de Diagnóstico e Investigación de Endemoepidemias, Noviembre 2006.
3. Egüez, K.E., Alonso-Padilla, J., Terán, C.B., Chipana, Z., Garcia, W., Torrico, F.T., … Pinazo, M. (2017). Rapid diagnostic tests duo as alternative to conventional serological assays for conclusive Chagas disease diagnosis. PLoS Neglected Tropical Diseases, 11(4). doi: 10.1371/journal.pntd.0005501
4. Uc-Cetina, V., Brito-Loeza, C., & Ruiz-Piña, H. (2013). Chagas parasites detection through gaussian discriminant analysis. *Abstraction and Application*, 8, 6-17. Recuperado de redi.uady.mx:8080/bitstream/handle/123456789/770/UcBritoRuiz_2013.pdf
5. Soberanis-Mukul, R., Uc-Cetina, V., Brito-Loeza, C., & Ruiz-Piña, H. (2013). An automatic algorithm for the detection of *Trypanosoma cruzi* parasites in blood

sample images. *Computer Methods and Programs in Biomedicine*, 112(3), 633-639. doi: 10.1016/j.cmpb.2013.07.013

6. Uc-Cetina, V., Brito-Loeza, C., & Ruiz-Piña, H. (2015). Chagas parasite detection in blood images using adaboost. *Computational and Mathematical Methods in Medicine*, 2015, 1-13. doi: 10.1155/2015/139681
7. Soberanis-Mukul, R. (2014). Algoritmos de segmentación de *Trypanosoma cruzi* en imágenes de muestras sanguíneas (Tesis de maestría). Universidad Autónoma de Yucatán, Mérida Yucatán.
8. Khan, S., Rahmani, H., Shah, S. and Bennamoun, M. (2018). A Guide to Convolutional Neural Networks for Computer Vision. Morgan and Claypool.
9. Ponce, C., Ponce, E., Vinelli, E., Montoya, A., Aguilar, V., Gonzalez, A., … Silveira, J. (2005). Validation of a rapid and reliable test for diagnosis of Chagas' disease by detection of *Trypanosoma cruzi*-specific antibodies in blood of donors and patients in Central America. *Journal of Clinical Microbiology*, 43(10), 5065 -5068. doi: 10.1128/JCM.43.10.5065-5068.2005
10. Egüez, K.E., Alonso-Padilla, J., Terán, C.B., Chipana, Z., Garcia, W., Torrico, F.T., … Pinazo, M. (2017). Rapid diagnostic tests duo as alternative to conventional serological assays for conclusive Chagas disease diagnosis. *PLoS Neglected Tropical Diseases*, 11(4). doi: 10.1371/journal.pntd.0005501
11. Duda, R., Hart, P., & Stork, D. (2001). *Pattern Classification (2nd Edition)*. NY, USA: Wiley-Interscience New York.
12. Cover, T.M., & Hart, P.E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21-27. doi: 10.1109/TIT.1967.1053964

13. Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference*, 511-518. doi: 10.1109/CVPR.2001.990517
14. Vapnik, V. (1998). Statistical learning theory. Wiley. ISBN: 978-0-471-03003-4
15. Ross, N.E., Pritchard, C.J., Rubin, D.M., & Dusé, A.G. (2006). Automated image processing method for the diagnosis and classification of malaria on thin blood smears. *Medical and Biological Engineering and Computing*, 44(5), 427-436. doi: 10.1007/s11517-006-0044-2
16. Soberanis-Mukul, R. (2012). Detección de Trypanosoma cruzi en imágenes obtenidas a partir de muestras sanguíneas (Tesis de pregrado). Universidad Autónoma de Yucatán, Mérida Yucatán.
17. Ojeda-Pat, A., Martin-Gonzalez, A., & Soberanis-Mukul, R. (2020). Convolutional Neural Network U-Net for *Trypanosoma cruzi* Segmentation. In book: Intelligent Computing Systems, pp.118-131. doi: 10.1007/978-3-030-43364-2_11
18. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
19. Razzak, M., Naz, S., & Zaib, A. (2018). Deep learning for medical image processing: overview, challenges and the future. In N. Dey, A. Ashour, & S. Borra (Eds.), *Classification in BioApps* (pp. 323-350). doi: 10.1007/978-3-319-65981-7_12

20. Deepa, N., & Chokkalingam, S. (2019). Deep convolutional neural networks (CNN) for medical image analysis. *International Journal of Engineering and Advanced Technology (IJEAT)*, 8(3). Recuperado de https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11290283S19.pdf
21. Chagas, C. "Nova tripanozomiaze humana: estudos sobre a morfolojia e o ciclo evolutivo do Schizotrypanum cruzi n. gen., n. sp., ajente etiolojico de nova entidade morbida do homem". Memórias do Instituto Oswaldo Cruz, 1(2), (1909). https://dx.doi.org/10.1590/S0074-02761909000200008
22. Bravo, T. (2004). *Trypanosoma cruzi*: Historia natural y diagnóstico de la enfermedad de Chagas. Revista Mexicana de Patología Clínica, pp. 205–219.
23. Guhl, F. (2008). Photograph showing an adult specimen of Triatoma dimidiata from Colombia [Fotografía]. Recuperado de https://es.wikipedia.org/wiki/Archivo:Triatoma_dimidiata-adult.jpg
24. National Library of Medicine, Medline Plus: Chagas Disease, https://medlineplus.gov/chagasdisease.html, last accessed on 2020-01-05.
25. W. H. Organization, Chagas disease American trypanosomiasis, https://www.who.int/chagas/disease/en/), last accessed on 2019-05-22.
26. Centers for Disease Control and Prevention, Chagas Disease, https://www.cdc.gov/parasites/chagas/, last accessed on 2020-01-05.
27. Peñas, K.D., Rivera, P.T., & Naval, P.C. (2017). Malaria parasite detection and species identification on thin blood smears using a convolutional neural network. 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 1-6. doi: 10.1109/CHASE.2017.51

28. Mehanian, C., Jaiswal, M., Delahunt, C.B., Thompson, C., Horning, M.P., Hu, L., … Bell, D.J. (2017). Computer-automated malaria diagnosis and quantitation using convolutional neural networks. 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), 116-125. doi: 10.1109/ICCVW.2017.22
29. Poostchi, M., Silamut, K., Maude, R. J., Jaeger, S., & Thoma, G. (2018). Image analysis and machine learning for detecting malaria. Translational Research, 194, 36–55. doi: 10.1016/j.trsl.2017.12.004
30. Górriz, M., Aparicio, A., Raventós, B., Vilaplana, V., Sayrol, E., & Lopez-Codina, D. (2018). Leishmaniasis parasite segmentation and classification using deep learning. In F. Perales, & J. Kittler (Eds.), Articulated Motion and Deformable Objects. doi: 10.1007/978-3-319-94544-6_6
31. Poppe, R. (2010). A survey on vision-based human action recognition. In Image and Vision Computing, vol. 28, no. 6, pp. 976-990.
32. Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Texts in Computer Science, Springer London.
33. Solomon, C.J., Breckon, T.P. (2010). *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. Wiley-Blackwell. doi:10.1002/9780470689776. ISBN 978-0470844731.
34. Gonzalez, R & Woods, R. (2002). Digital Image Processing, 2nd ed., Prentice Hall.
35. Reema, K. and Wafa, K. (2015). Imágenes de tipos de vecindarios de pixeles [Imagen]. Recuperado de https://www.researchgate.net/figure/Fig-11-a-Pixel-p-and-its-4-neighbors-b-diagonal-neighborhood-c-8-neighborhood_fig10_301201506

36. Ganegedara, T. (2019). Imagen de la operación de stride y padding [Imagen]. Recuperado de https://stackoverflow.com/questions/42883547/intuitive-understanding-of-1d-2d-and-3d-convolutions-in-convolutional-neural-n/44628011
37. Razzak, M., Naz, S., & Zaib, A. (2018). Deep learning for medical image processing: overview, challenges and the future. In N. Dey, A. Ashour, & S. Borra (Eds.), *Classification in BioApps* (pp. 323-350). doi: 10.1007/978-3-319-65981-7_12
38. Sidey-Gibbons, J., & Sidey-Gibbons, C. (2019). Machine learning in medicine: a practical introduction. BMC Medical Research Methodology, 19(64). doi: 10.1186/s12874-019-0681-4
39. Thakur, A., & Ranjan, R. (2019). Image Segmentation and Semantic Labeling using Machine Learning. International Journal of Recent Technology and Engineering, vol. 7. ISSN: 2277-3878
40. Magoulas, G.D., & Prentza, A. (2001). Machine learning in medical applications. In G. Paliouras, V. Karkaletsis, & C. Spyropoulos (Eds.), *Machine Learning and Its Applications* (pp. 300-307). doi: 10.1007/3-540-44673-7_19
41. Deepa, N., & Chokkalingam, S. (2019). Deep convolutional neural networks (CNN) for medical image analysis. *International Journal of Engineering and Advanced Technology (IJEAT)*, 8(3). Recuperado de https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11290283S19.pdf
42. Liang, Z., Powell, A.D., Ersoy, I., Poostchi, M., Silamut, K., Palaniappan, K., … Thoma, G.R. (2016). CNN-based image analysis for malaria diagnosis. *2016*

IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 493-496. doi: 10.1109/BIBM.2016.7822567

43. Quinn, J.A., Nakasi, R., Mugagga, P.K., Byanyima, P., Lubega, W., & Andama, A. (2016). Deep convolutional neural networks for microscopy-based point of care diagnostics. *Proceedings of International Conference on Machine Learning for Health Care*, 56(0). Recuperado de https://arxiv.org/abs/1608.02989
44. Dong, Y., Jiang, Z., Shen, H., Pan, W., Williams, L.A., Reddy, V.V., … Bryan, A.W. (2017). Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells. *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, 101-104. doi: 10.1109/BHI.2017.7897215
45. Gopakumar, G.P., Swetha, M., Sai-Siva, G., & Sai-Subrahmanyam, G. (2018). Convolutional neural network-based malaria diagnosis from focus stack of blood smear images acquired using custom-built slide scanner. *Journal of. Biophotonics*, 11(3). doi: 10.1002/jbio.201700003
46. Pollak, L. (2019). Ejemplo de una FCNN [Imagen]. Recuperado de https://towardsdatascience.com/efficient-method-for-running-fully-convolutional-networks-fcns-3174dc6a692b
47. Long, J., Shelhamer, E., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. Computer Vision and Pattern Recognition. arXiv:1411.4038
48. Pollak, L. (2019). FCNN con una capa de convolución [Imagen]. Recuperado de https://towardsdatascience.com/efficient-method-for-running-fully-convolutional-networks-fcns-3174dc6a692b

49. Gorner, M. (2019). 3D convolutional network [Imagen]. Recuperado de https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8
50. Gorner, M. (2019). Mapa de características multi capa [Imagen]. Recuperado de https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8
51. CS231n. (2018). Max pooling [Imagen]. Recuperado de https://cs231n.github.io/convolutional-networks/#conv
52. Noh, H., Hong, S., and Han, B. (2015). Ejemplificación de la deconvolución [Imagen]. Recuperado de https://towardsdatascience.com/review-deconvnet-unpooling-layer-semantic-segmentation-55cf8a6e380e
53. Noh, H., Hong, S., and Han, B. (2015). Learning Deconvolution Network for Semantic Segmentation. In Computer Vision (ICCV) 2015 IEEE International Conference on.
54. CS231n. (2019). Convolución traspuesta 1D [Imagen]. Recuperado de https://aman.ai/cs231n/detection/#down-sampling-and-up-sampling-using-fully-convolutional-architectures
55. Dumoulin, V., and Visin, F. (2016). Transposed convolution 2D [Imagen]. Recuperado de https://github.com/vdumoulin/conv_arithmetic
56. Wang, W., Shen, J., and Shao, L. (2018). Diagrama de una CNN [Imagen]. Recuperado de https://www.semanticscholar.org/paper/Video-Salient-Object-Detection-via-Fully-Networks-Wang-Shen/927e0a7b1f06102725a026f690077654fa53c76e/figure/1

57. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
58. Nieradzik, L. (2020). Losses for segmentation, https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/, last accessed 2019/09/03.
59. Fernandez-Moral, E., Martins, R., Wolf, D., & Rives, P. (2018). A New Metric for Evaluating Semantic Segmentation: Leveraging Global and Contour Accuracy. *2018 IEEE Intelligent Vehicles Symposium* (IV). doi: 10.1109/IVS.2018.8500497
60. Chen, X., Williams, B., Vallabhaneni, S., Czanner, G., Williams, R., & Zheng, Y. (2019). Learning Active Contour Models for Medical Image Segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR). doi: 10.1109/CVPR.2019.01190
61. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proc. of the IEEE International Conference on Computer Vision, pp. 1026-1034. doi: 10.1109/iccv.2015.123. 71.
62. Zhang, Z. & Liu, Q. (2017). Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*. doi: 10.1109/LGRS.2018.2802944
63. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR). doi: 10.1109/CVPR.2016.90

64. Zhang, K., Sun, M., Han, T., Yuan, X., Guo, L., & Liu, T. (2018). Residual Networks of Residual Networks: Multilevel Residual Networks. In *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6), pp. 1303-1314. doi: 10.1109/TCSVT.2017.2654543.
65. Lu, Y., Zhong, A., Li, Q., & Dong, B. (2018). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *35th International Conference on Machine Learning*, ICML 2018, vol. 7, pp. 5181-5190. Taken from https://arxiv.org/abs/1710.10121
66. Bengio, Y., Simard, P., & Frasconi, D. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, vol. 5, pp. 157-66. doi: https://doi.org/10.1109/72.279181
67. Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., & Pal, C. (2016). The Importance of Skip Connections in Biomedical Image Segmentation. In*: Deep Learning and Data Labeling for Medical Applications*. doi: 10.1007/978-3-319-46976-8_19
68. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity Mappings in Deep Residual Networks. In Computer Vision – ECCV 2016, vol. 9908. doi: 10.1007/978-3-319-46493-0_38
69. Haber, E. & Ruthotto, L. (2017). Stable architectures for deep neural networks. Inverse Problems, vol.34, pp. 014004. doi: 10.1088/1361-6420/aa9a90
70. Lu, Y., Zhong, A., Li, Q., & Dong, B. (2018). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *35th International Conference on Machine Learning*, ICML 2018, vol. 7, pp. 5181-5190. Taken from https://arxiv.org/abs/1710.10121

71. Sauer, T. (2017). Numerical Analysis. Pearson, 3rd. Edition. ISBN-13 978-0134697338
72. Chen, R., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural Ordinary Differential Equations. In *32nd Conference on Neural Information Processing Systems*. Tomado de https://arxiv.org/abs/1806.07366
73. Bengio, Y., Simard, P., & Frasconi, D. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, vol. 5, pp. 157-66. doi: https://doi.org/10.1109/72.279181
74. Süli, E. & Mayers, D. (2003). An Introduction to Numerical Analysis. Cambridge University Press. ISBN 0-521-00794-1.
75. Google. Google Colaboratory (CoLAB), https://colab.research.google.com, last accessed 2020/04/05.
76. Nieradzik, L. (2020). Losses for segmentation, https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/, last accessed 2019/09/03.
77. Fernandez-Moral, E., Martins, R., Wolf, D., & Rives, P. (2018). A New Metric for Evaluating Semantic Segmentation: Leveraging Global and Contour Accuracy. *2018 IEEE Intelligent Vehicles Symposium (IV)*. doi:10.1109/IVS.2018.8500497