UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas

PROYECTO FINAL DE CARRERA
INGENIERÍA INFORMÁTICA

Diseño y desarrollo de una herramienta de segmentación automática de cromosomas

Alumno: Fenoglio, Sebastián

Director: Dr. Martínez, César

Co-Director: Dr. Gerard, Matías

Santa Fe, Abril de 2019

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Resumen

En el presente trabajo se desarrolló una herramienta que permite la segmentación automática de cromosomas con tinción en banda G. Las diferentes partes de la herramienta se desarrollaron secuencialmente y de forma separada en módulos que luego se integran. Cada uno de ellos fue documentado para facilitar la continuidad del proyecto.

En el módulo de pre-procesamiento se utilizaron técnicas de procesamiento de imágenes para eliminación del ruido que puede haber en la imagen de entrada y la extracción de los objetos de la misma. Entre las mencionadas técnicas se pueden mencionar la ecualización adaptada de histograma con contraste limitado (CLAHE), el umbral adaptado de Otsu y operaciones morfológicas.

A continuación, se desarrolló el módulo encargado de la separación de solapamientos mediante un enfoque basado en redes convolucionales profundas. Al no tener datos etiquetados, también se creó una herramienta de generación de solapamientos sintéticos para poder entrenar dichas redes. Luego de diferentes experimentos que varían la complejidad del problema, el modo de creación de datos y las arquitecturas utilizadas, se obtienen los mejores resultados con una red propuesta que es similar a U-net [1]. Se obtienen resultados cercanos a 89 % en Recall y superior a 83 % en el coeficiente de Jaccard con la red HuV3BNN.

El tercer módulo corresponde al post-procesamiento, en el que se exploraron métodos que permiten corregir las máscaras de segmentación obtenido previamente en la separación. Nuevamente, se realizaron distintos experimentos en los que se varía los métodos utilizados y la forma de combinarlos entre sí. Los mejores resultados se obtienen acoplando una segunda instancia de la red HuV3BNN a la salida de la utilizada para la separación y añadiendo los métodos de eliminación de pequeñas imperfecciones (EPI) y corrección por canales (CC). Se obtiene un desempeño mayor al 93 % en Recall, superior a 91 % en el coeficiente de Jaccard y de 90 % en la nueva medida confiabilidad.

Por último, se integraron los módulos y se desarrolló una interfaz de usuario por línea de comandos. Además, se probó la herramienta con cromosomas reales, obteniéndose resultados prometedores.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Índice general

Resumen I
Índice general III
Lista de figuras VII
Lista de tablas IX

1. Introducción 1
1.1. Justificación 1
1.2. Objetivos 3
1.2.1. Objetivo general 3
1.2.2. Objetivos específicos 4
1.3. Alcance 4
1.3.1. Requerimientos funcionales 4
1.3.2. Requerimientos no funcionales 4
1.3.3. Restricciones 4
1.4. Descripción del problema y estado del arte 4
1.4.1. Extracción de objetos 5
1.4.2. Análisis de objetos 6
1.4.3. Separación de solapamientos 7
1.4.4. Clusters de tres o más cromosomas 8
1.5. Propuesta 9

2. Pre-procesamiento 11
2.1. Método propuesto 11
2.1.1. Análisis de fondo y realce de imagen 11
2.1.2. Umbral 13
2.1.3. Procesamiento morfológico 13
2.1.4. Análisis de componentes conexas 15
2.2. Resultados y discusión 16
2.2.1. Umbral, eliminación de componentes espurias y relleno de agujeros 17
2.2.2. Umbral adaptado y umbral de Otsu 17
2.2.3. Eliminación de objetos grandes 19
2.2.4. Eliminación de objetos en el borde 20
2.2.5. Integración 20

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

2.3. Conclusiones 23

3. Separación de solapamientos 24

3.1. Enfoque utilizado 24
3.2. Generación de datos 26
3.2.1. Método de generación 26
3.2.2. Datasets generados 28
3.3. Arquitecturas consideradas 29
3.3.1. Modelo inspirado en U-net 29
3.3.2. Otras arquitecturas consideradas 31
3.4. Resultados y discusión 32
3.4.1. Solapamientos de 2 cromosomas fijos 34
3.4.2. Solapamientos de hasta 5 cromosomas de un mismo cariograma 35
3.4.3. Solapamientos de hasta 5 cromosomas provenientes de 10 cariogramas 35
3.4.4. Solapamientos de hasta 5 cromosomas provenientes de todos los cariogramas 37
3.4.5. Evaluación del mejor método con cromosomas reales . 38
3.5. Conclusiones 45

4. Post-procesamiento 46

4.1. Métodos propuestos 46
4.1.1. Eliminación de pequeñas imperfecciones 47
4.1.2. Corrección por canales según umbral para la determinación de clases confiables 47
4.1.3. k-NN mediante el uso de un umbral para determinar los píxeles confiables 48
4.1.4. Red convolucional W 49
4.2. Resultados y discusión 49
4.2.1. Eliminación de pequeñas imperfecciones 50
4.2.2. Umbrales para detectar clases confiables 50
4.2.3. Corrección por canales según umbral para determinar las clases confiables 51
4.2.4. k-NN mediante el uso de un umbral para determinar las clases confiables 52
4.2.5. Red convolucional profunda de corrección 53
4.2.6. Comparación de los mejores métodos 56
4.2.7. Evaluación del mejor método sobre cromosomas reales 57
4.3. Conclusiones 62

5. Integración 64

5.1. Materiales 64
5.2. Pre-procesamiento 65

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

5.3. Generación de datos 68
5.4. Entrenamiento de red convolucional 70
5.5. Separación 73
5.6. Post-procesamiento 74
5.7. Integración e interfaz 76
5.8. Resultados y discusión 80

6. Conclusiones 82
6.1. Conclusiones 82
6.2. Trabajo a futuro 83
6.2.1. Pre-procesamiento 83
6.2.2. Separación de solapamientos 84
6.2.3. Post-procesamiento 84
6.2.4. Integración e interfaz 85

Bibliografía 86

A. Documentación de la herramienta 92

B. Documentación de la generación de datos 108

C. Documentación del entrenamiento de la red convolucional 116

D. Anteproyecto 132

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Índice de figuras

1.1. Cromosomas humanos con tinción en banda G y su cariograma. 2
1.2. Ejemplo de posibles segmentaciones para un solapamiento. . . 3
1.3. Diagrama de bloques del proceso de segmentación. 5
1.4. Cromosomas en banda G tocándose. 7
1.5. Cromosomas en banda G solapados. 8
1.6. Diagrama de bloques de la propuesta de este trabajo. 9

2.1. Diagrama de bloques de la propuesta de pre-procesamiento. . 12
2.2. Ejemplo de subdivisión para calcular el umbral adaptado de Otsu. 14
2.3. Elemento estructurante de 3x3. 15
2.4. Resultados de aplicar umbral, eliminación de componentes espurias y relleno de agujeros. 18
2.5. Diferencia entre la combinación de umbrales propuesta y sólo el umbral de Otsu global en imagen con tinción en banda G. 19
2.6. Diferencia entre la combinación de umbrales propuesta y sólo el umbral de Otsu global en imagen con tinción en banda Q. 20
2.7. Resultado de la eliminación de objetos grandes. 21
2.8. Resultado de la eliminación de objetos en el borde. 21
2.9. Imagen con tinción en banda G usada como entrada. 22
2.10. Veinte de las imágenes de salida correspondientes a los objetos segmentados. 22

3.1. Diagrama de la arquitectura U-net. 25
3.2. Diagrama de bloques del proceso de generación de datos. . . 27
3.3. Ejemplo de un solapamiento sintético. 29
3.4. Diagrama del modelo propuesto por Hu. 30
3.5. Diagrama de la arquitectura VGG16. 32
3.6. Diagrama de la arquitectura PSPNet. 33
3.7. Predicciones de clusters de cromosomas reales que no requieren post-procesamiento. 40
3.8. Predicciones de clusters de cromosomas reales con resultado aceptable que requieren un post-procesamiento simple. . . . 41

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

3.9. Predicciones de clusters de cromosomas reales que requerirían un post-procesamiento más complejo. 42
3.10. Predicciones de cromosomas reales con buen resultado y que no requieren post-procesamiento. 43
3.11. Predicciones de cromosomas reales individuales erróneas y que requieren post-procesamiento. 44

4.1. Ejemplo del método de corrección por canales. 48
4.2. Predicciones de la red convolucional de corrección con artefactos en zonas correspondientes al fondo. 54
4.3. Ejemplos de predicciones realizadas sobre datos sintéticos luego del post-procesamiento. 58
4.4. Predicciones de clusters de cromosomas reales post-procesadas con resultado diferente al obtenido en la separación. 59
4.5. Predicciones de clusters de cromosomas reales post-procesadas con resultado similar al obtenido en la separación. 60
4.6. Predicciones imposibles de resolver para el procedimiento planteado. 61

5.1. Flujo de trabajo. 65
5.2. Funciones del módulo de pre-procesamiento. 66
5.3. Funciones del proyecto de generación de datos. 68
5.4. Scripts del proyecto de entrenamiento de la red convolucional. 71
5.5. Clase del módulo de separación. 73
5.6. Funciones del módulo de post-procesamiento. 75
5.7. Funciones desarrolladas para la integración de los módulos. . 76
5.8. Imagen microscópica utilizada como entrada de la herramienta y su correspondiente cariograma. 80
5.9. Imágenes de salida generadas por la herramienta. 80
5.10. Resultados sobre cluster de 8 cromosomas. 81

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Índice de tablas

2.1. Valores por defecto del módulo de pre-procesamiento. 17

3.1. Datasets generados. 28
3.2. Resultados con solapamientos de 2 cromosomas fijos. 34
3.3. Resultados con solapamientos de hasta 5 cromosomas de un mismo cariograma. 35
3.4. Resultados con solapamientos de hasta 5 cromosomas provenientes de 10 cariogramas. 36
3.5. Resultados con solapamientos de hasta 5 cromosomas provenientes de todos los cariogramas. 37

4.1. Resultados de la eliminación de pequeñas imperfecciones. . . 51
4.2. Resultados de aplicar umbrales para detectar clases confiables 51
4.3. Resultados de la corrección por canales según umbral para determinar las clases confiables 52
4.4. Resultados de k-NN mediante el uso de un umbral para determinar las clases confiables 53
4.5. Resultados de la red convolucional profunda de corrección. . . 54
4.6. Resultados de la red convolucional profunda de corrección aplicando la máscara de pre-procesamiento. 55
4.7. Resultados de la comparación de métodos. 56
4.8. Resultados de la evaluación del mejor método sobre cromosomas reales. 57

5.1. Resumen de parámetros de cada módulo. 79

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Capítulo 1

Introducción

En este capítulo se introduce al lector en la problemática de la segmentación de cromosomas y se explica cómo se afrontará en el resto del trabajo. La Sección 1.1 describe el problema abordado, la relevancia del mismo y los beneficios que puede proporcionar su resolución. Luego, en la Sección 1.2 se presentan los objetivos del proyecto y en la Sección 1.3 el alcance del mismo. A continuación, en la Sección 1.4 se realiza una descripción más detallada del problema y los métodos utilizados en la literatura para resolver cada parte. Por último, en la Sección 1.5 se expone la propuesta de solución abordada en este trabajo a la segmentación de cromosomas.

1.1. Justificación

La citogenética es la parte de la genética que se encarga del estudio de la estructura y de la función de los cromosomas [2]. La información del genoma humano está almacenada típicamente en 23 pares de cromosomas, de los cuales 22 son homólogos y se denominan autosómicos, mientras que el restante se denomina par sexual. Este par es responsable de determinar los rasgos masculinos y femeninos de los individuos [3]. Así, se da lugar a 24 tipos de cromosomas humanos: los 22 homólogos y los 2 del par sexual.

Los estudios sobre cromosomas son un importante procedimiento de diagnóstico médico en dictámenes prenatales, en pacientes con retrasos mentales y múltiples problemas de nacimiento, en pacientes con anormal desarrollo sexual y en algunos casos de infertilidad o múltiples abortos espontáneos. También es útil para el estudio y tratamiento de pacientes con neoplasias malignas y desórdenes hematológicos [3].

Una representación muy utilizada para el análisis de los cromosomas es el cariograma [3, 4]. Para realizarlo, primero se extrae una muestra de células del ser humano bajo estudio, se le aplica alguna técnica de tinción para realzar los cromosomas y luego se obtiene una imagen microscópica de ellos

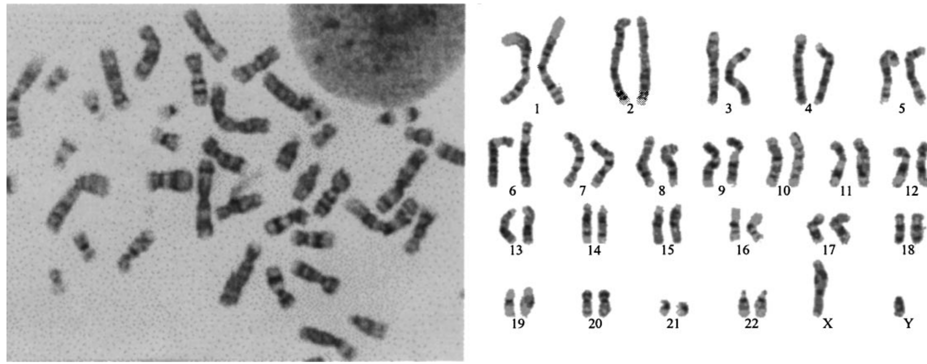


Figura 1.1: Cromosomas de un ser humano con tinción en banda G y su correspondiente cariograma [7].

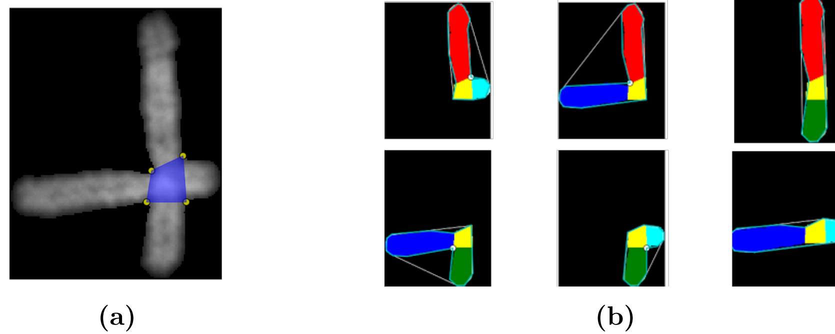
generalmente en metafase [1]. A partir de ésta, para obtener el cariograma debe pre-procesarse la imagen, segmentarse los cromosomas y luego clasificarse en los 24 tipos de cromosomas humanos existentes [5,6]. La Figura 1.1 presenta un ejemplo de cromosomas con tinción en banda G. Existen otras que logran distinguir cada tipo de cromosoma con distintos colores, como la llamada FISH, pero son más costosas y suelen utilizarse como información complementaria a la tinción en banda G [4].

Un paso fundamental en la obtención del cariograma es la segmentación de los cromosomas a partir de una imagen obtenida de la observación microscópica de las células extraídas del ser humano bajo estudio. En la misma, dos o más cromosomas pueden estar solapados y generar ambigüedades sobre cómo separarlos. La Figura 1.2a presenta un ejemplo típico de ésta situación. Como puede apreciarse en la figura 1.2b, existen tres formas diferentes de segmentar estos cromosomas, y cada caso puede conllevar una interpretación diferente. El problema real se complica aún más al ser los cromosomas estructuras no rígidas y al poder solaparse más de dos de ellos. Además, la segmentación manual es una tarea costosa en términos de tiempo y recursos. Por tal motivo, la automatización del proceso supondría una ventaja ya que se evitaría utilizar dichos recursos en la segmentación y permitiría concentrar el esfuerzo en el análisis de los cromosomas.

Existen productos comerciales que realizan todo el proceso de obtención del cariograma, pero tienen algunas desventajas. Primero, se venden como paquetes cerrados [9] [10] [11], por lo que no se sabe cómo funcionan realmente, no pueden adaptarse a necesidades específicas ni mejorarse con el desarrollo de nuevos métodos de segmentación. Algunos de ellos funcionan sólo con determinado equipamiento [10] o están atados a las técnicas de tinción más costosas [10] [11], limitando su aplicabilidad. También exis-

[1]Fase del ciclo celular en la que pueden apreciarse los patrones que forman las bandas de los cromosomas según la tinción utilizada.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.



(a) (b)

Figura 1.2: Posibles segmentaciones para un solapamiento de cromosomas [8]. (a) Dos cromosomas con la zona de solapamiento marcada en azul. (b) Por columna se ve cada una de las tres segmentaciones posibles con la zona de solapamiento en amarillo y las diferentes partes de cada cromosoma en verde, celeste, rojo y azul.

te software de código abierto pero son semi-automáticos [12] [13], es decir, requieren la intervención humana en algún punto.

Dado que en la actualidad no existe una herramienta de código abierto que realice la segmentación automática de cromosomas, en este proyecto se propone el desarrollo de una herramienta computacional de código abierto que, a partir de una imagen microscópica de cromosomas humanos con tinción en banda G, los segmente de forma totalmente automática y dé como resultado imágenes de dichos cromosomas separados, aún cuando éstos estén originalmente solapados. El hecho que sea de código abierto permite fundamentalmente dos cosas. Primero, una mejora continua y distribuida de la herramienta debido a que cualquier persona con los conocimientos suficientes puede inspeccionarla y optimizarla conforme avance la investigación sobre la segmentación de cromosomas. Segundo, la adaptación para aplicaciones específicas y la integración en otras herramientas más completas.

Esta herramienta, sumada a un hardware con modestas prestaciones y a un sistema de clasificación de cromosomas como el descripto en [14], podrían interactuar para constituir un sistema de cariotipado automático de acceso libre para la comunidad.

1.2. Objetivos

1.2.1. Objetivo general

Diseñar y desarrollar una herramienta que permita la segmentación automática de cromosomas a partir de una imagen microscópica.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

1.2.2. Objetivos específicos

- Analizar y comparar el desempeño de las técnicas de segmentación de cromosomas en metafase disponibles en la literatura.
- Definir el algoritmo de segmentación y el conjunto de parámetros configurables del mismo.
- Desarrollar un prototipo funcional y bien documentado de la aplicación.
- Analizar el desempeño del algoritmo con medidas objetivas empleando corpus de imágenes de libre disponibilidad.
- Obtener una interfaz que permita la abstracción de las funciones y opciones de la herramienta por sobre su desarrollo algorítmico.

1.3. Alcance

1.3.1. Requerimientos funcionales

- Pre-procesar las imágenes para facilitar su posterior segmentación.
- Segmentar y separar los cromosomas.
- Post-procesar los cromosomas para facilitar su posterior análisis o clasificación.

1.3.2. Requerimientos no funcionales

- Robustez frente a distintos tipos de imágenes de entrada.
- Se utilizará como lenguaje de programación Python ya que éste prioriza la legibilidad y facilitaría la reutilización de la herramienta.

1.3.3. Restricciones

- Se desarrollará una interfaz sólo por línea de comandos.
- Se utilizarán imágenes de cromosomas humanos con tinción en banda G.

1.4. Descripción del problema y estado del arte

Una forma de plantear el proceso de la segmentación es dividirla en las etapas que pueden observarse en el diagrama de bloques de la Figura 1.3. Se empieza extrayendo los objetos de la imagen para luego analizar si

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

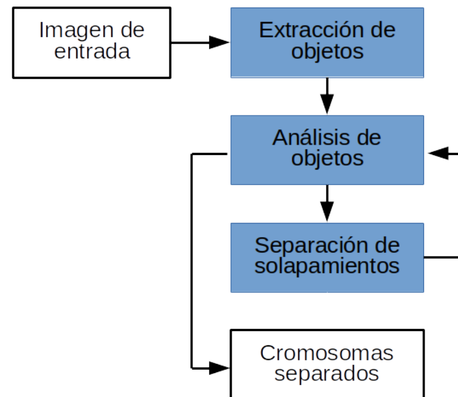


Figura 1.3: Diagrama de bloques del proceso de segmentación de cromosomas.

cada uno es un cromosoma individual o un cluster de ellos. Los que ya son identificados como individuales no requieren un procesamiento posterior, mientras que los restantes deben ser separados en cromosomas individuales verificando en cada paso si lo que se obtiene son efectivamente cromosomas individuales o clusters de menor tamaño.

A continuación se explica en más detalle los diferentes métodos propuestos para la resolución de cada bloque. Además, en la sección 1.4.4 se plantean soluciones para resolver solapamientos de más de dos cromosomas.

1.4.1. Extracción de objetos

El primer paso propuesto para la segmentación es la extracción de los objetos del fondo, generalmente mediante la binarización de la imagen. Los métodos usados se pueden agrupar en dos tipos: globales y locales. Los primeros se basan en las características globales de la imagen para la determinación de un único umbral, mientras que los segundos lo determinan según la posición en el espacio en que se encuentre cada píxel y utilizando información de su vecindad.

El método global del umbral de Otsu es el más utilizado para las imágenes con tinción en banda G [15–18]. La idea detrás del mismo es que, dada una imagen con dos clases de píxeles, se busque el valor que minimice la combinación de la varianza interna de cada clase [19].

En cuanto a los métodos locales, en [19] se reporta que son con los que se obtienen mejores resultados para cromosomas con tinción en banda Q. En ese caso, las imágenes se caracterizan por tener un fondo negro y las bandas brillantes, que son equivalentes a las bandas oscuras obtenidas con el bandeo G. En el mismo artículo, se muestra que dentro de este grupo el que obtiene resultados más cercanos a la segmentación perfecta realizada manualmente

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

es el umbral adaptado. Éste consta de dividir la imagen en cuadrados no solapados en los que se calcula en cada uno el umbral de Otsu. Así, se obtiene una matriz de valores que luego se lleva al tamaño original de la imagen interpolando dichos valores [20]. La lógica detrás de la interpolación es evitar cortar cromosomas que se encuentren en más de un cuadrado. Sin embargo, en [21] se utiliza para la segmentación de cromosomas con bandeo G.

Dentro del mismo grupo, en [19] se mencionan los métodos de re-umbralizado local y umbralizado adaptado multietapa. El primero consiste en aplicar el umbral de Otsu a la imagen entera y, en un segundo paso, aplicar nuevamente el umbral de Otsu a cada componente conexa obtenida anteriormente. Mientras que el segundo utiliza dos percentiles como umbrales globales para determinar un intervalo en el que fuera de él se clasifica el píxel como objeto o fondo y dentro de él se determina un umbral local basado en la media y la varianza de una vecindad definida.

Siguiendo con los métodos locales, otro enfoque son los fundamentados en conjuntos de nivel. Éstos se basan en detectar los cromosomas haciendo evolucionar curvas hasta que se ajusten a los contornos de los mismos. La evolución está guiada en la búsqueda de la minimización de la energía, que generalmente está definida según la región que contiene la curva [5, 19, 22].

Por último, en [19] se nombra un tercer grupo de métodos: los umbrales multinivel. Los utilizados en el trabajo son dos métodos de ajuste de histograma con una optimización basada en enjambre de partículas. Se obtienen los peores resultados de los tres grupos con la mayor carga computacional.

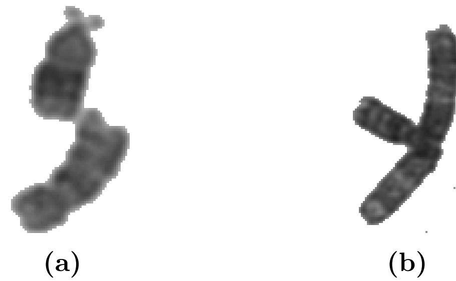
1.4.2. Análisis de objetos

Una vez que se tiene los objetos extraídos del fondo, el siguiente paso es determinar si cada uno es un cromosoma individual o un cluster de ellos. Para ello, existe un método basado en el análisis del cambio de la curvatura del eje medio del objeto combinado con la detección de puntos de acumulación que pueden ser asociados a los extremos de los cromosomas. En éste se asume que un cromosoma individual no puede tener cambios de curvatura bruscos (no puede tener forma de 'S') ni tener más de dos puntos de acumulación [20].

Otro método se basa en la comparación del área del objeto con el área de su convex hull, en la proporción del eje mayor y el eje menor de la menor elipse que lo contiene y en el recuento de puntos finales del esqueleto morfológico del mismo, usando umbrales empíricos y asumiendo que un cromosoma individual debe tener exactamente dos puntos finales [6]. En otros trabajos se utiliza directamente el recuento de puntos finales del esqueleto [17, 21].

Un último método se basa en los parámetros de área, circularidad (definida como la proporción entre el área del objeto y del mínimo círculo que lo contiene) y largo del eje medio del objeto [5].

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.



(a) **(b)**

Figura 1.4: Cromosomas en banda G tocándose. (a) Solapamiento que se resuelve mediante el método de pale path. (b) Solapamiento que se resuelve mediante el análisis de su contorno.

1.4.3. Separación de solapamientos

El paso posterior es separar los clusters de cromosomas en cromosomas individuales, que debe realizarse de forma diferente según como se dé el solapamiento. En la Figura 1.4 pueden verse ejemplos de cromosomas tocándose y en la Figura 1.5 cromosomas solapados.

En el caso de la Figura 1.4a, es posible separar los dos cromosomas buscando los dos puntos del contorno más cercanos que se unan por medio de una línea contenida dentro del cluster [22] o siguiendo el camino de los píxeles de menor densidad. En este último método, con tinción en banda G, estos píxeles son los más claros (pale path) [23–25].

Por el contrario, los mencionados métodos no pueden resolver la separación de los cromosomas de la Figura 1.4b. En este caso, se analiza el contorno del cluster en busca de puntos candidatos de corte, generalmente los de menor concavidad. Luego, se seleccionan los dos con menor distancia acumulada respecto a los demás puntos [6], o se explora exhaustivamente el espacio de las posibles líneas que estén dentro del cluster y generen como resultado lo más parecido a cromosomas individuales [20].

El último caso a considerar es el solapamiento de dos cromosomas como los de la Figura 1.5. Puede resolverse nuevamente buscando puntos candidatos con baja concavidad en el contorno y eligiendo los cuatro que determinan la zona de solapamiento de distintas formas. Una es seleccionar los más cercanos al nodo del esqueleto morfológico del cluster [17], es decir, los más cercanos al punto del esqueleto en el que se crucen más de dos caminos. Otra es buscar los cuatro puntos que formen lo más parecido a un romboide verificando cada cuadrilátero posible [20]. Un enfoque distinto es calcular la triangulación Delaunay de los puntos pertenecientes al contorno y buscar los dos triángulos de mayor área más cercanos entre sí, que determinan la zona de solapamiento [16, 26].

La mayoría de las propuestas para resolver solapamientos como los de la Figura 1.5 suponen que los cromosomas que se cruzan tienen baja curvatura,

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Figura 1.5: Cromosomas en banda G solapados.

por lo que se separan uniendo los segmentos opuestos entre sí respecto a la zona de solapamiento. Por ejemplo, en el caso de la Figura 1.2a se tomaría siempre la segmentación de la tercera columna de la Figura 1.2b. Sin embargo, en algunos trabajos se tiene en cuenta los patrones de bandeo para hacer esta asignación correctamente [7, 16]. En particular, en [16] se tiene en cuenta dicha información para saber cuál de los dos cromosomas estaba encima del otro utilizando una matriz de covarianza y le asigna al mismo la zona de solapamiento.

Los métodos mencionados buscan separar los cromosomas de los casos de las Figuras 1.4b y 1.5 con líneas rectas. Por el contrario, en [5] se propone separarlos partiendo de un punto candidato con baja concavidad y siguiendo el camino de mayor gradiente hasta llegar al contorno. La lógica es que en las zonas de superposición la densidad es mayor que en el resto del cluster y el cambio de intensidades es abrupto.

Con el auge de las redes convolucionales profundas, surge un enfoque totalmente diferente para la segmentación de imágenes basado en ellas [27]. En el ámbito de la segmentación de cromosomas, sólo en [8] se propone entrenar una red convolucional profunda para la separación de sólo dos cromosomas solapados con tinción M-FISH. Su idea a futuro es que podría separarse la imagen completa en bounding-boxes en los que se pueda aplicar esta herramienta de a dos cromosomas repetidamente.

1.4.4. Clusters de tres o más cromosomas

El último problema a resolver es cuando hay cluster de tres o más cromosomas. Para esto, deben aplicarse las herramientas anteriores repetidamente. La opción sencilla es tener un método para determinar si lo que se quiere resolver es un toque o un solapamiento, de forma de aplicar alguna de las herramientas anteriores y al resultado volver a analizar si es un cromosoma individual o un cluster, y continuar así hasta obtener todos cromosomas individuales [6].

Otra manera de hacer esto es crear un árbol en el que en cada nodo se apliquen las tres posibles operaciones: corte por pale paths, toques a través de corte geométrico y resolución de solapamiento. Luego, se analice en las hojas cuán parecidos a un cromosoma individual son y la cantidad de

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

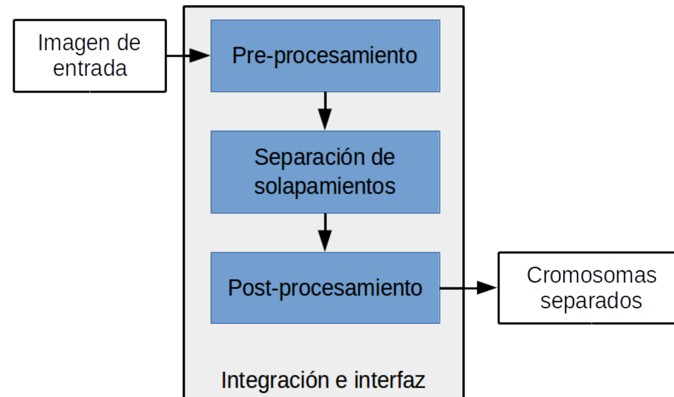


Figura 1.6: Diagrama de bloques de la propuesta para la segmentación de cromosomas.

cortes que fueron necesarios, para así elegir un único camino [20]. Ésta es más costosa computacionalmente pero asegura el tratamiento de todas las alternativas. En [21] se adopta un enfoque similar para tomar la decisión pero más simplificado ya que trata posibles solapamientos entre sólo dos cromosomas y tiene en cuenta sólo las posibles separaciones a través de pale paths y de resolución de overlaps.

1.5. Propuesta

En la Figura 1.6 se presenta un diagrama de bloques con la propuesta para la segmentación de cromosomas. Cada bloque corresponde a un módulo a desarrollar que se analiza en un capítulo dedicado. En el Capítulo 2 se aborda el pre-procesamiento que comprende la extracción de objetos del fondo mencionado en la Sección 1.4.1 y la eliminación de objetos ruidosos no correspondientes a cromosomas. Luego, en el Capítulo 3 se desarrolla el bloque de separación de solapamientos que combina el análisis de objetos explicado en la Sección 1.4.2, la separación de cromosomas solapados de la Sección 1.4.3 y el tratamiento de clusters de más de dos cromosomas referido en la Sección 1.4.4. En el Capítulo 4 se analizan métodos que puedan contribuir a mejorar el resultado obtenido en la separación de solapamientos. Posteriormente, en el Capítulo 5 se afronta la integración de los tres módulos mencionados y el desarrollo de la interfaz de usuario para la conformación de la herramienta de segmentación, también haciendo énfasis en los detalles de la codificación. Se finaliza con el Capítulo 6, en el que se exponen las conclusiones del trabajo y el trabajo a futuro. Además, en el Apéndice A puede encontrarse la documentación de la herramienta.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Capítulo 2

Pre-procesamiento

En este capítulo se aborda el problema del pre-procesamiento de la imagen microscópica con tinción en banda G. Se espera obtener sub-imágenes de la misma en las que haya únicamente un cluster de cromosomas. Con ese objetivo, se combinan métodos de procesamiento de imágenes tales como ecualización de histograma adaptativa con contraste limitado, umbralizado de Otsu, umbralizado adaptado y operaciones morfológicas así como detección y análisis de componentes conexas. Dichos métodos son aplicados sobre imágenes microscópicas de cromosomas para analizar los resultados que se obtienen.

El capítulo se organiza de la siguiente manera. Se comienza explicando el método propuesto para el pre-procesamiento en la Sección 2.1. Luego, en la Sección 2.2 se muestran los experimentos realizados con el fin de comprobar el correcto funcionamiento del módulo. Por último, en la Sección 2.3 se presentan las conclusiones del capítulo. Además, en el Apéndice A se encuentra la documentación del código desarrollado.

2.1. Método propuesto

Para el módulo de pre-procesamiento se propone el diagrama de bloques de la Figura 2.1. Este proceso consta de 6 etapas en las que se realiza un mejorado de la imagen, se extrae cada objeto del fondo mediante un umbralizado, se eliminan las componentes espúreas y se analiza cada componente conexa a fin de descartar las que correspondan a objetos ruidosos grandes. El detalle de cada etapa se describe en las secciones a continuación.

2.1.1. Análisis de fondo y realce de imagen

La imagen de entrada es una imagen micróscopica de los cromosomas a segmentar. Se consideran los casos en que la tinción de los mismos es banda G porque, como ya se mencionó, es la más utilizada por ser la técnica de

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

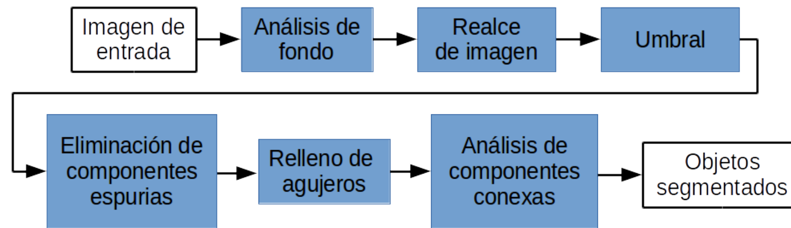


Figura 2.1: Diagrama de bloques de la propuesta de pre-procesamiento.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

tinción más accesible. Además, la célula debe estar en metafase, fase en que los patrones de bandeo son visibles. Aunque no sea información que se utilice en este módulo, es relevante durante el proceso de separación de solapamientos del Capítulo 3.

Para unificar el posterior tratamiento, se verifica que el fondo de la imagen I sea blanco. En caso de serlo no se modifica la imagen, mientras que, si es negro, se calcula la inversa de la imagen con la fórmula:

$$Inv(I)(x, y) = 255 - I(x, y) \tag{2.1}$$

Asumiendo que las imágenes de entrada tendrán los cromosomas centrados, se toman ROIs (regiones de interés) de las cuatro esquinas de la misma de tamaño 10x10 píxeles. Así, se promedian todas las intensidades de ellas para verificar si están más cercano al negro o al blanco y, con esto, saber si se debe modificar la imagen o no.

Se pretende mejorar el contraste de la imagen para facilitar la separación entre el fondo y los objetos. Una forma de hacerlo es el ecualizado del histograma (HE) [17] que realiza una transformación sobre la imagen basada en la función de distribución acumulada (CDF), pero tiene el inconveniente que no mejora mucho el contraste en zonas muy oscuras o muy claras respecto al resto de la imagen. Para solucionar esto, se puede usar la ecualización adaptada del histograma (AHE), en la que para cada píxel se calcula la CDF en una vecindad definida. El inconveniente de AHE es que, al igual que HE, produce malos resultados cuando el histograma de la imagen está muy localizado (hay modas muy marcadas), que es el caso ideal de fondo homogéneo. Para resolver esto, la alternativa es limitar la amplificación del contraste (CLAHE) recortando el histograma en cierto valor y redistribuyendo dichos los valores entre todas las intensidades por igual previamente al cálculo de la CDF [28]. Se propone la aplicación de CLAHE, siendo el tamaño de vecindad y el límite de contraste parámetros de este método.

Por último, se normaliza la imagen ya que se pretende utilizar todo el rango [0-255] para seguir con el objetivo de aumentar el contraste y así mejorar el desempeño de la binarización. La misma consiste en un desplazamiento de las intensidades de la imagen de forma que la mínima resulte en 0,

seguido por un escalado de las mismas a través de una constante lineal para llevar el rango de la imagen al rango [0-255]. De esta forma, el nuevo valor I_N de la imagen de cada píxel de la imagen I estará dado por la fórmula de la Ecuación (2.2).

$$I_N = 255 \, \frac{I - I_{Min}}{I_{Max} - I_{Min}}, \tag{2.2}$$

donde I_{Min} e I_{Max} son los valores mínimos y máximos valores de intensidad de la imagen I.

2.1.2. Umbral

En este paso se busca obtener una imagen binaria en la que la intensidad de los píxeles sea 255 en los que se detecte la presencia de un cromosoma y 0 en caso contrario. El método más usado para imágenes con tinción en banda G es el umbral de Otsu [15–18], pero el mismo tiene inconvenientes en imágenes con fondo variable y puede generar varios agujeros dentro de los cromosomas si tienen intensidades cercanas al fondo. Por ello, en este bloque se propone una combinación entre el umbral de Otsu y el umbral adaptado, adicionando los resultados. Así, la máscara binaria resultante será la unión entre las imágenes binarias obtenidas en cada método.

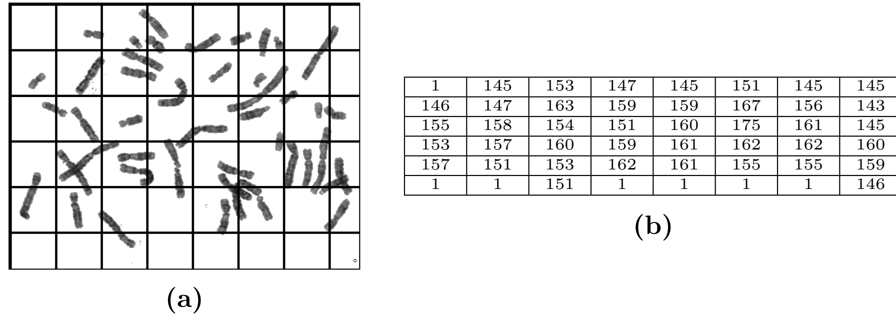
Para esto, a partir de la imagen resultante del realce, se calcula el umbral adaptado con regiones de tamaño según un parámetro dado en píxeles. El método se basa en separar la imagen en regiones no solapadas, como se observa en la Figura 2.2a, en las que se calcula localmente el valor del umbral de Otsu. Este valor es el que divide a los píxeles en dos clases minimizando la combinación de la varianza interna de cada una de ellas [29]. Luego, los valores se colocan en una matriz que será, si el tamaño de ventana es de 100x100 como suele serlo [20, 21], 1/100 veces más pequeña que la correspondiente a la imagen dada. Continuando el ejemplo, en la Figura 2.2b se observa la matriz de 6x8 obtenida. Entonces, para llevarla al tamaño de la imagen, se interpola bilinealmente y, de esta forma, se tiene un umbral para cada píxel.

Así, con el umbral adaptado se tiene en cuenta la información local que permite una mejor adaptación a la variación del fondo y a la detección de objetos pequeños con el riesgo de que, cuanto más pequeña sea la ventana, más probable es que en la máscara binara haya componentes espurias. Mientras que con el umbral de Otsu se tiene justamente una visión global que permite un mejor rendimiento en fondos más homogéneos y logra una mejor detección de objetos grandes.

2.1.3. Procesamiento morfológico

Para eliminar las componentes espurias que pueden ser deshechos del proceso de tinción o trozos muy pequeños de cromosomas desprendidos, en

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.



(a)

1	145	153	147	145	151	145	145
146	147	163	159	159	167	156	143
155	158	154	151	160	175	161	145
153	157	160	159	161	162	162	160
157	151	153	162	161	155	155	159
1	1	151	1	1	1	1	146

(b)

Figura 2.2: (a) Ejemplo de subdivisión de la imagen realizada para aplicar el umbral adaptado de Otsu. (b) Matriz resultante de calcular el umbral de Otsu para cada región de (a).

varios trabajos se propone realizar una operación morfológica de apertura [20, 21] a la imagen binaria B obtenida anteriormente. Una característica de la apertura morfológica es que suaviza los bordes y, en este caso, esa información no se quiere perder ya que podría ser útil para la separación de solapamientos si se afrontara con los métodos que utilizan el contorno del objeto mencionados en la Sección 1.4.3. Por esto, se propone realizar primero una erosión con un elemento estructurante cuadrado de un tamaño que será dado como parámetro con todos elementos iguales a 255 para la eliminación de las componentes espurias. Dicha erosión provoca la reducción del tamaño de los objetos de la máscara y, por ende, la desaparición de los que sean más pequeños que el elemento estructurante dado.

Luego, al resultado obtenido, se lo usa como imagen inicial F para una reconstrucción morfológica, que se basa en dilataciones geodésicas aplicadas repetidamente. La dilatación geodésica $D_G(F)$ es una operación binaria que expande los bordes de una imagen binaria inicial dada F y que utiliza otra de referencia G, con la que va haciendo la operación lógica AND bit a bit para limitar dicha expansión [30]. En este caso, se aplica para hacer crecer el resultado obtenido de la erosión hasta que no haya más cambios, de forma que se recuperen las máscaras de los cromosomas que se vieron afectadas por la mencionada erosión. Se utiliza el elemento estructurante de la Figura 2.3 y la siguiente fórmula con $G = B$.

$$R_G^D = D_G^{(k)}(F), \qquad \textit{hasta que } D_G^{(k)}(F) = D_G^{(k+1)}(F) \tag{2.3}$$

En la máscara binaria obtenida pueden quedar pequeños agujeros dentro de los cromosomas debido a imperfecciones en la captura de la imagen y en el umbralizado. Éstos deben corregirse porque pueden afectar el rendimiento de los métodos de separación de solapamientos que trabajan sobre las máscaras binarias, en especial los que utilizan los esqueletos morfológicos mencionados en la Sección 1.4.3. El procedimiento consiste en la aplicación

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

0	255	0
255	255	255
0	255	0

Figura 2.3: Elemento estructurante de 3x3 utilizado para la reconstrucción morfológica.

de una reconstrucción morfológica por dilatación geodésica sobre la imagen binaria B, empleando el mismo elemento estructurante de 3x3 de la Figura 2.3, considerando $G(x, y) = 255 - B(x, y)$ en la Ecuación (2.3). Dicha corrección se calcula como:

$$F(x,y) = \begin{cases} 255 - B(x,y) & si\ (x,y)\ pertenece\ al\ borde\ de\ B \\ 0 & para\ otros\ casos \end{cases} \quad (2.4)$$

Esto no sólo rellena los pequeños espacios que pueden quedar dentro de un cromosoma, sino que también lo hace con los agujeros que pueden formarse por varios cromosomas solapándose en una cadena cerrada. Por esto, se realiza una resta entre la imagen rellena obtenida con la Ecuación (2.3) y la imagen umbralizada previa, de forma de obtener los agujeros que rellenó el proceso. Luego, se analiza cada uno de ellos restaurando los que sean mayores a un parámetro dado en píxeles cuadrados, asumiendo que los agujeros rellenos externos a los cromosomas son más grandes que los internos que sí deben llenarse.

Una alternativa podría ser el llenado de los agujeros mediante un algoritmo de relleno por difusión [17], que es poco eficiente y requiere un punto inicial o semilla. Otra opción para rellenar pequeños agujeros podría ser simplemente una operación de cierre morfológica [18], pero ésta tiene las desventajas que suaviza los bordes y podría unir cromosomas cercanos. Por ello, se prefiere el método recién desarrollado para evitar estos problemas a pesar de que sea más complejo.

2.1.4. Análisis de componentes conexas

En este punto, cada componente conexa obtenida puede corresponder a un cromosoma, a un cluster de ellos o a núcleos de las células que son relativamente grandes. Para diferenciar éstos últimos, de la imagen binaria anterior se buscan las componentes conexas correspondientes a los distintos objetos de la imagen mediante un algoritmo de seguimiento de bordes para imágenes binarias implementado en OpenCV [31]. Entonces, se filtra a los objetos que tengan un área mayor a un umbral definido en píxeles cuadrados para descartar los grandes residuos. A éstos se le calcula la proporción entre su área y el área de su convex hull y se eliminan los objetos grandes cuya mencionada proporción sea mayor a un umbral dado. Así, se evita descartar clusters de varios cromosomas que podrían tener un área considerable.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Sigue habiendo inconveniente cuando estos objetos ruidosos grandes están cortados en la imagen y se ven parcialmente, por lo que su área no es considerable como para su descarte. Por ello, se propone eliminar los objetos que estén en los bordes de la imagen que tengan un segmento sobre dicho borde mayor a un parámetro μ dado en píxeles. Para lograrlo, se realiza una reconstrucción morfológica por dilatación geodésica mediante la ecuación (2.3) con el elemento estructurante de la Figura 2.3, $G = B$ y la Ecuación (2.5). Así, se evita descartar cromosomas que estén tocando el borde de la imagen y se obtienen los objetos de los bordes que se quieren descartar, los cuales se eliminan de B.

$$F(x,y) = \begin{cases} B(x,y) & si\ (x,y)\ \epsilon\ borde\ de\ B\ y\ segmento > \mu \\ 0 & para\ otros\ casos \end{cases} \tag{2.5}$$

Las componentes conexas resultantes se utilizan como máscaras para segmentar cada objeto de la imagen mediante operaciones lógicas AND. Así, se obtiene como salida una lista de imágenes correspondientes a los objetos segmentados en la imagen, sin distinción si son cromosomas individuales o clusters de ellos. A cada una de ellas se la normaliza mediante la ecuación (2.2) ya que anteriormente se aplicó de forma global, de forma que no se garantiza que cada objeto segmentado ocupe la totalidad del rango [0-255]. Esto es de utilidad ya que para la futura separación de los cromosomas solapados del Capítulo 3 se utiliza la información de las intensidades de los grises.

2.2. Resultados y discusión

Para analizar el rendimiento de este módulo, se procede a realizar varios experimentos sobre algunas imágenes tomadas al azar del corpus de imágenes usado en [32]. En cada caso, se realiza una inspección visual a fin de evaluar el correcto funcionamiento de los bloques bajo análisis. Para cada bloque, se utilizarán los valores por defecto del módulo listados en la Tabla 2.1, salvo que se especifique lo contrario. Éstos fueron determinados a partir de experimentos preliminares realizados.

Para todos los experimentos, se utiliza un tamaño de ventana de 8x8 y un límite de contraste de 40 en el bloque de realce de imagen explicado en la Sección 2.1.1. El mencionado bloque es muy dependiente de la imagen de entrada, por lo que con el dataset considerado sólo se utiliza con el fin de mejorar el contraste del bandeo puesto que el fondo es homogéneo. Pero si se usara en otras imágenes con fondo variable podría ayudar también a mejorar el contraste respecto al mismo. Aún en el mismo dataset los resultados pueden ser muy distintos según los parámetros que se elijan.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 2.1: Valores por defecto del módulo de pre-procesamiento.

Parámetro	Bloque	Valor por defecto
Tamaño de ventana	Realce de imagen	8x8
Límite de contraste	Realce de imagen	40
Vector de tamaños de ventana	Umbral	[0,100]
Tamaño de elemento estructurante	Eliminación de componentes espurias	7x7
Tamaño máximo de agujero interno	Relleno de agujeros	10
Umbral de área	Análisis de componentes conexas	4000
Umbral de proporción con área de convex hull	Análisis de componentes conexas	0.8
Tamaño máximo de segmento en borde	Análisis de componentes conexas	30

2.2.1. Umbral, eliminación de componentes espurias y relleno de agujeros

El objetivo de este experimento es analizar el rendimiento de los bloques de umbral explicado en la Sección 2.1.2 y el procesamiento morfológico de la Sección 2.1.3. Para ello, se colorea con cyan lo que se detecta como fondo luego de aplicar los mencionados procesos y a continuación se muestran algunas ROIs de los resultados obtenidos en la Figura 2.4.

Para analizar el rendimiento del bloque de umbral de la sección 2.1.2 puede verse principalmente en la Figura 2.4c la precisión que se obtiene en los bordes de los objetos, teniendo en cuenta lo irregulares que son en la imagen de entrada. Para ver los resultados del bloque de eliminación de componentes espurias, se observa con detalle en la Figura 2.4a como éstas son eliminadas. Mientras que en las Figuras 2.4a y 2.4b pueden verse los agujeros que se rellenaron mediante el bloque de relleno de agujeros en los píxeles blancos, puesto que no fueron pintados de color cyan y por ende no son considerado fondo, sino parte del objeto.

2.2.2. Umbral adaptado y umbral de Otsu

En esta sección se pretende comparar el resultado obtenido anteriormente utilizando la combinación de umbral de Otsu global y umbral adaptado en el bloque de la sección 2.1.2 con lo que se obtendría sólo aplicando el umbral de Otsu global (lo que allí se refiere como umbral con tamaño de ventana 0). Para ello, se aplican ambas opciones a una misma región de una imagen tomada al azar y se compara visualmente el resultado en la Figura

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

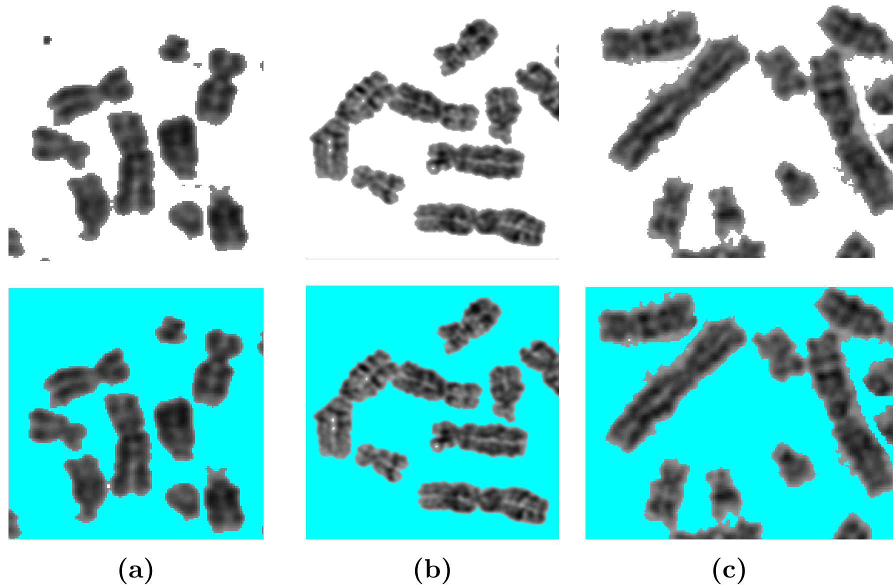


Figura 2.4: (a-c) En la primera fila ROIs de imágenes de entrada con tinción en banda G y en la segunda fila los resultados de aplicar umbral, eliminación de componentes espurias y relleno de agujeros. El color cyan corresponde al fondo de la imagen.

2.5. Luego, aunque en este trabajo se considere sólo la tinción en banda G, para mostrar la mejoría que supone el umbral adaptado se aplica el método a una ROI de una una imagen microscópica con tinción en banda Q [20] y se presenta el resultado en la Figura 2.6.

En la Figura 2.5 se ve que se obtiene un resultado similar con las dos opciones comparadas, lo que quiere decir que, en este caso, al tener las imágenes fondos completamente homogéneos, no es absolutamente necesario utilizar el umbral adaptado. Pero también prueba que dicho umbral adaptado no distorsiona significativamente el resultado correcto del Otsu global. De todas formas, en la Figura 2.5c se puede ver que hay una diferencia, aunque siga sin ser considerable, en la precisión de los bordes.

La imagen en tinción en banda Q de la Figura 2.6a se caracteriza por tener un fondo menos homogéneo respecto a las de banda G del dataset utilizado, por lo que en la Figura 2.6b puede verse el resultado que se obtiene combinando distintos tamaños de ventana (0 y 100), obviamente modificando también los parámetros del bloque de realce de imágenes. En la Figura 2.6c se muestra en color la diferencia que hay en la segmentación sólo utilizando el umbral de Otsu global respecto a la combinación de distintos tamaños de ventana. Es notable la mejoría utilizando el umbral adaptado, por eso es que se lo incluye como opción en la herramienta.

Un detalle adicional que surge de la observación de los resultados con

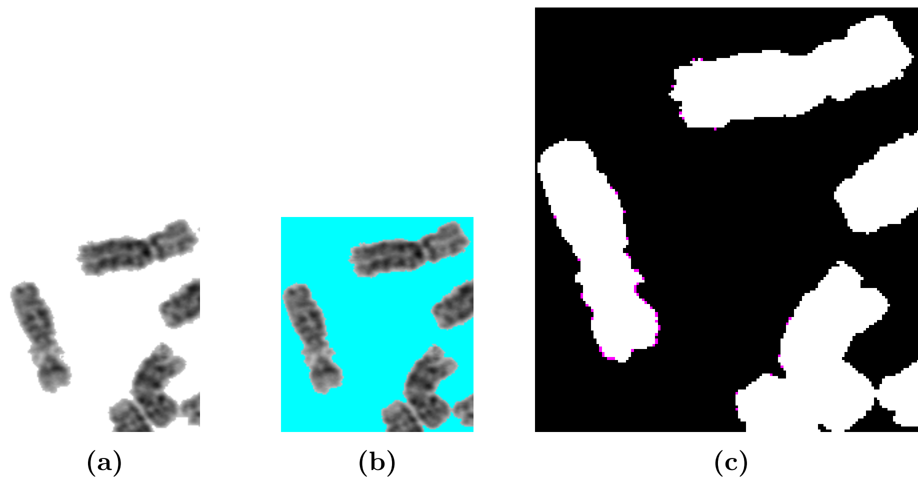


Figura 2.5: Diferencia entre aplicar la combinación de umbrales propuesta y sólo el umbral de Otsu global en una imagen con tinción en banda G. (a) ROI de imagen de entrada con tinción en banda G. (b) Resultado de aplicar la herramienta con la combinación de tamaños de ventana 0 y 100 en el bloque de umbral. (c) Ampliado para que se aprecie, marcado en color magenta, la diferencia entre el resultado de (b) y aplicar sólo Otsu global (ver bordes interiores). Lo que está en color es detectado también como objeto usando Otsu global.

la tinción en banda Q es el correcto funcionamiento del bloque de análisis de fondo ya que las mismas, a diferencia de las que están en banda G, poseen fondo negro y en la Figura 2.6 se prueba que se obtienen resultados satisfactorios.

2.2.3. Eliminación de objetos grandes

El objetivo de este experimento es examinar la función encargada de eliminar los objetos grandes indeseados de la sección 2.1.4. En el dataset considerado no se da este caso, por lo que se prueba con un bajo umbral de área (50) y con umbral de proporción de área del objeto con el área del convex hull igual a 0.8, el valor por defecto. Este proceso se aplica a una región de una imagen tomada al azar y el resultado se presenta en la Figura 2.7.

En la Figura 2.7 se ve que se eliminan los cromosomas individuales por tener su convex hull un área muy similar a la del propio cromosoma, mientras que los clusters no son descartados justamente por lo contrario. Sólo el pequeño cluster de dos cromosomas que se encuentra en la parte superior de la Figura 2.7a es eliminado, pero si el umbral de área fuera más alto como debería, no hubiese sido detectado como posible objeto a descartar. Así, se demuestra el correcto funcionamiento siempre que se elija un umbral

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

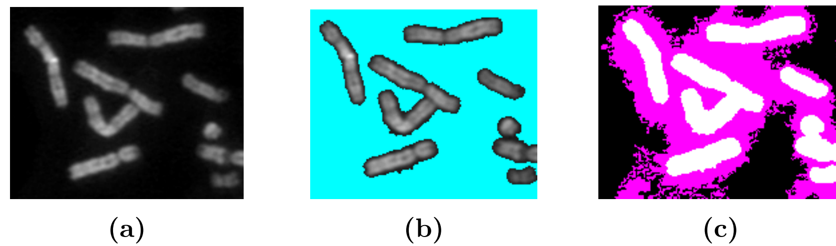


Figura 2.6: Diferencia entre aplicar la combinación de umbrales propuesta y sólo el umbral de Otsu global en una imagen con tinción en banda Q. (a) ROI de imagen de entrada con tinción en banda Q. (b) Resultado de aplicar la herramienta con la combinación de tamaños de ventana 0 y 100 en el bloque de umbral. (c) En color, se ve la diferencia entre el resultado de (b) y aplicar sólo Otsu global. Lo que está en color es detectado también como objeto usando Otsu global.

de área lo suficientemente grande para que no descarte los cromosomas individuales (como se vio en las imágenes de los experimentos anteriores) y lo suficientemente pequeño para que detecte los objetos grandes que se desean eliminar.

2.2.4. Eliminación de objetos en el borde

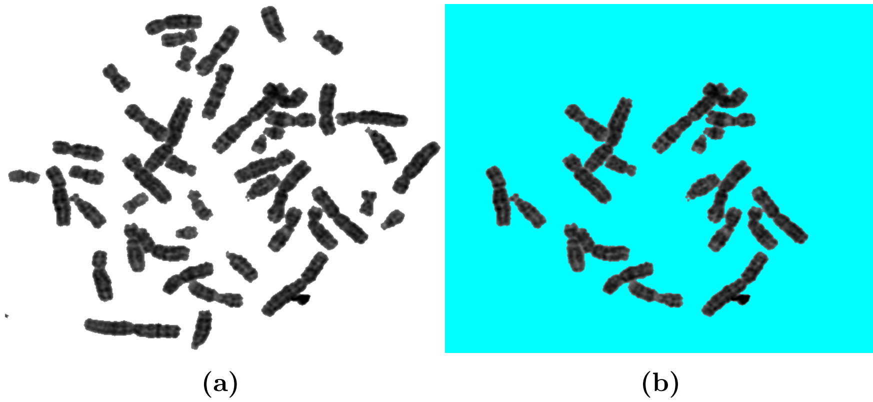
El objetivo del experimento es probar el funcionamiento de la eliminación de los objetos que están en el borde de la imagen. Como en el dataset considerado no se presenta esta situación, se recorta a una de ellas para que queden algunos cromosomas en el borde como se ve en la Figura 2.8a. En la Figura 2.8b puede observarse el resultado de la eliminación de dichos objetos.

Como se esperaba, se ve señalados en verde en la Figura 2.8b que los objetos que tienen poca superficie de contacto con el borde permanecen intactos. Mientras que los demás se eliminan por ser más probable que pertenezcan a objetos relativamente grandes, como sucede con el cromosoma inferior de la Figura 2.8a y señalado en rojo en la Figura 2.8b.

2.2.5. Integración

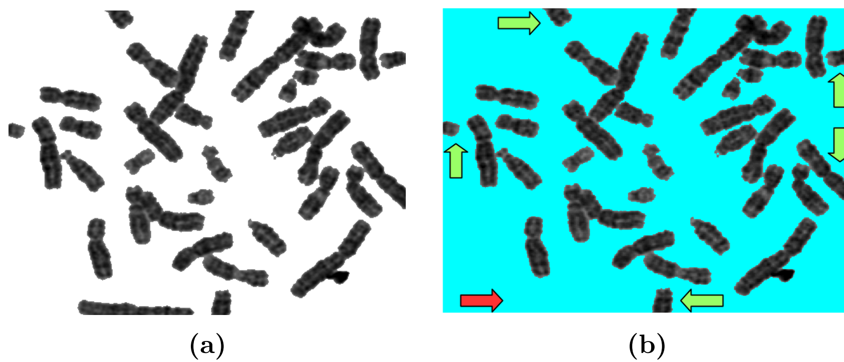
En este experimento se quiere exponer el correcto funcionamiento del módulo integrado, para lo que simplemente se muestra en la Figura 2.10 veinte de las imágenes de salida correspondiente a la imagen de entrada de la Figura 2.9. Se observa un acertado comportamiento tanto para los cromosomas individuales como para los clusters.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.



(a) (b)

Figura 2.7: Resultado aplicando un bajo umbral de área (50) y con umbral de proporción de área del objeto con el área del convex hull igual a 0.8 en el bloque de análisis de componentes conexas. (a) ROI de imagen de entrada con tinción en banda G. (b) Resultado de aplicar la herramienta.



(a) (b)

Figura 2.8: Resultado de la eliminación de objetos en el borde. (a) ROI de imagen con tinción en banda G. (b) En verde se señalan los cromosomas no eliminados y en rojo el eliminado.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Figura 2.9: Imagen con tinción en banda G usada como entrada en el experimento de integración.

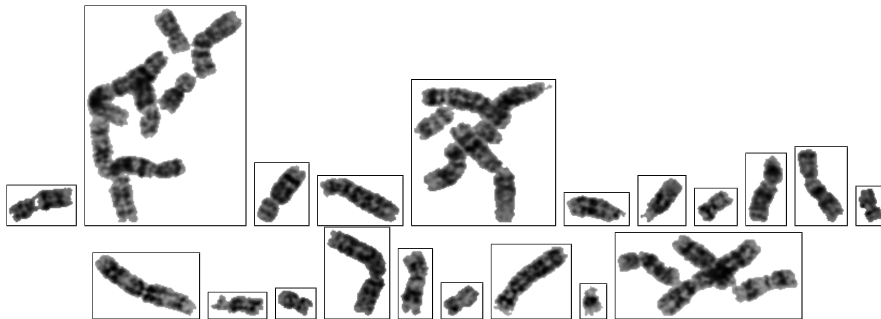


Figura 2.10: Veinte de las imágenes de salida correspondientes a los objetos segmentados.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

2.3. Conclusiones

Se propuso un método de pre-procesamiento basado en métodos de procesamiento de imágenes y se comprobó que los resultados fueran los esperados. Se observó que los parámetros en general dependen de las características particulares de la imagen de entrada, en especial los referidos al realce de la imagen. Sin embargo, el mencionado bloque tuvo un rendimiento correcto, destacando que no modificó el fondo homogéneo como hubiese hecho la ecualización de histograma simple. De forma similar, puede observarse que el umbral adaptado presenta resultados correctos para fondos homogéneos aunque fuese incluido por su rendimiento con fondos no homogéneos.

Respecto a la eliminación de objetos grandes y objetos de borde, se tuvo que cambiar los parámetros por defecto para probar con el dataset considerado ya que en el mismo no se da ninguno de estos casos. Sin embargo, se incluyen en la herramienta ya que por ejemplo en la Figura 1.1 se observa un objeto grande a la derecha que, en caso de no ser detectado como tal, debería ser detectado como objeto de borde. Además, se observa que tener en cuenta el área del convex hull de los objetos grandes antes de eliminarlos es una opción factible para evitar descartar clusters de cromosomas con área relativamente grande. Adicionalmente, se ve que la operación morfológica de erosión es útil para la eliminación de pequeños objetos ruidosos.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Capítulo 3

Separación de solapamientos

En este capítulo se afronta el problema de la separación de cromosomas solapados, a partir de una imagen que contiene sólo un cluster de cromosomas. Se pretende obtener máscaras binarias que, aplicadas a dicha imagen, den como resultado los cromosomas individuales involucrados en el cluster. Para ello, a pesar que en la literatura predominen los métodos geométricos, se utiliza un método basado en redes convolucionales por su mayor flexibilidad. Dado que los corpus de imágenes de cromosomas disponibles consisten de relativamente pocas imágenes y casos de solapamiento, un problema adicional que debe ser abordado es la generación sintética de datos. Luego, se realizan diversos experimentos con numerosas arquitecturas en busca de la que provee mejores resultados, variando la complejidad de una a otra y la forma en que éstas combinan la información. Para poder comparar el rendimiento entre ellas se calculan las medidas de recall y coeficiente de Jaccard. Además, se analizan los resultados obtenidos con cromosomas reales utilizando el modelo que da mejores resultados con solapamientos sintéticos.

El capítulo se organiza de la siguiente manera. Se comienza mencionando algunos antecedentes en la separación de cromosomas solapados en la Sección 3.1 y se define el enfoque que se utilizará. En la Sección 3.2 se explica el procedimiento para generar solapamientos sintéticos y los diferentes datasets generados. Luego, en la Sección 3.3 se exponen las arquitecturas de redes convolucionales consideradas. En la Sección 3.4 se muestran los experimentos realizados con distintas combinaciones de datasets y arquitecturas. Por último, en la Sección 3.5 se presentan las conclusiones del capítulo. Además, en los Apéndices B y C se encuentra la documentación del código para la generación de datos y para la separación de solapamientos, respectivamente.

3.1. Enfoque utilizado

El enfoque más utilizado en la literatura para la separación de cromosomas solapados es el geométrico, que generalmente se basa en el análisis de

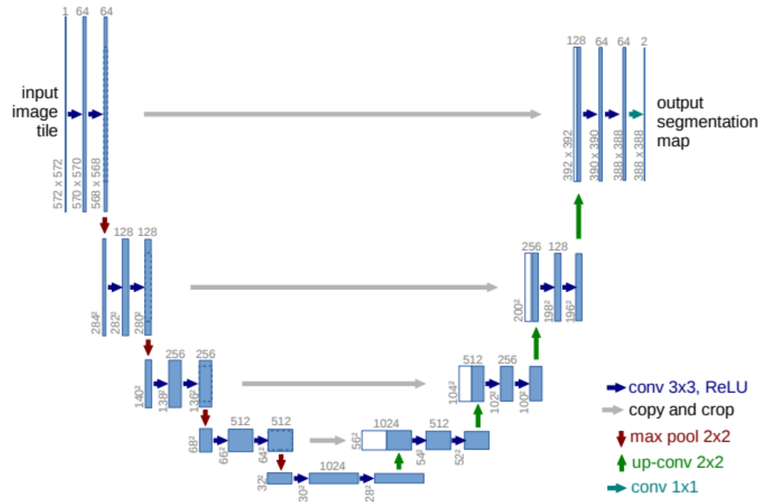


Figura 3.1: Diagrama de la arquitectura U-net [1].

los contornos. El método más frecuente es la búsqueda de puntos de menor concavidad para trazar una línea de separación [6, 17, 20], aunque también hay otros métodos que se basan en la triangulación Delaunay [16, 26] de los píxeles del contorno y en el seguimiento de los píxeles de mayor intensidad [23–25].

Una técnica poco explorada en la literatura para esta tarea son las redes neuronales. En particular, las redes convolucionales han demostrado un gran desempeño en tareas de segmentación de imágenes [27], en las que se utilizaron diversas arquitecturas para distintos problemas. Entre ellas, se puede mencionar la arquitectura U-net de la Figura 3.1 que se distingue analizar y combinar características a distintas escalas y se utilizó para tareas de segmentación en imágenes biomédicas [1].

Otra arquitectura es la FRRN (Full-Resolution Residual Networks), que combina un flujo de información con las imágenes submuestreadas con otro en el que se analiza la imagen completa, usada para la clasificación de objetos en escenas urbanas [33]. La arquitectura PSPNet combina flujos de información con distintos tamaños de filtros y fue utilizada para segmentación semántica en escenas urbanas y cotidianas [34]. SegNet está inspirada en la U-net pero presenta la novedad de guardar los índices de submuestreo para utilizarlos al momento de sobremuestreo, logrando una optimización de velocidad y uso de memoria en las tareas de segmentar semánticamente escenas urbanas y cotidianas [35].

En el ámbito de la segmentación de cromosomas, sólo Hu [8] propone entrenar una red convolucional profunda inspirada en la antes nombrada U-net para la separación de dos cromosomas solapados con tinción M-FISH. Su idea a futuro es que podría separarse la imagen completa en bounding-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

boxes en los que se pueda aplicar esta herramienta de a dos cromosomas repetidamente.

Teniendo en cuenta estos antecedentes, en este capítulo se aborda el problema de la separación de cromosomas solapados utilizando redes convolucionales profundas. Éstas tienen la ventaja, respecto a los métodos geométricos, de ser más flexibles y adaptables a las diferentes combinaciones de solapamientos de cromosomas que pueden darse. Es decir, en los métodos geométricos debe analizarse cada posible caso de solapamiento que pueda darse, mientras que una red correctamente entrenada no debería tener problemas para separar casos no vistos previamente.

El costo de ello es la gran cantidad de tiempo y recursos computacionales que necesita una red convolucional profunda para entrenarse. Además, otro inconveniente que surge en la utilización de redes convolucionales es la necesidad de gran cantidad de datos con su correcta separación (llamado ground truth) para su entrenamiento. Como no se dispone del ground truth de las imágenes, además de la elección de la arquitectura utilizada para la de los cromosomas mediante su segmentación semántica, en este trabajo se afronta la generación de datos para el mencionado entrenamiento.

3.2. Generación de datos

Como ya se mencionó, el entrenamiento de una red convolucional profunda requiere de una gran cantidad de datos. En este caso, si se quiere que la misma aprenda a separar cromosomas, debe tener imágenes de solapamientos con su respectiva solución. La forma utilizada para que la red pueda distinguir uno de otros es según su tipo. Entonces, para cada solapamiento, se debe tener una máscara etiquetada con la clase de cada cromosoma involucrado en el mismo.

3.2.1. Método de generación

En esta subsección se explica el proceso por el cual se crean datos sintéticos de cromosomas solapados con su correspondiente máscara de correcta separación de forma que se pueda entrenar a las redes con ellos. El diagrama de bloques de la Figura 3.2 presenta una descripción del método de generación de datos. En el Apéndice B se encuentra la documentación de la herramienta desarrollada para tal fin.

El primer paso es extraer los cromosomas a partir de los cariogramas manualmente producidos provistos por el dataset [32, 36]. Se extrajeron los cromosomas empleando OpenCV [31], aplicando umbralizado y un algoritmo de detección de componentes conexas. Luego, se guardan en orden de izquierda a derecha y de arriba a abajo, con el objetivo de que luego se pueda etiquetar a cada uno según su ubicación en el cariograma. Se utilizaron

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

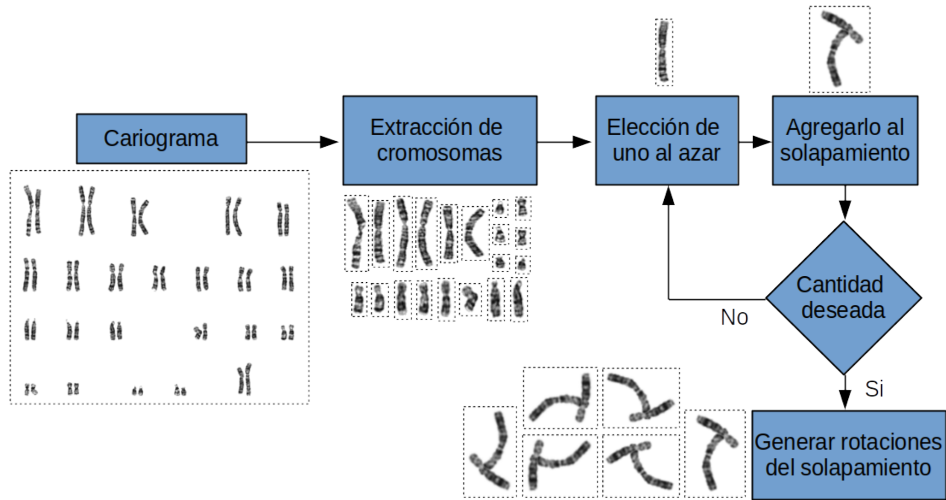


Figura 3.2: Diagrama de bloques del proceso de generación de datos sintéticos.

los 536 cariogramas que contenían 46 cromosomas para evitar problemas de etiquetado de ellos.

Cada imagen se construyó adicionando sobre un fondo blanco el número de cromosomas que se desea solapar, cada uno en una ubicación al azar y con una rotación aleatoria. Para esto, se verifica que al agregar cada uno ellos haya una intersección del cromosoma agregado con al menos uno de los que ya haya en el solapamiento y se mide el porcentaje de solapamiento con respecto al cromosoma agregado, de forma que se evite cromosomas totalmente ocultos o cromosomas sin solapamientos. En otras palabras, para cada cromosoma agregado, se verifica que el porcentaje de solapamiento esté entre un mínimo y un máximo definidos. También se comprueba que no se agreguen cromosomas repetidos ni del mismo tipo, lo que constituye una limitación para la herramienta.

Por último, para cada solapamiento obtenido, se generan varias versiones de la imagen rotada para aumentar la cantidad de datos y para que la red presente la invarianza a la rotación.

Desde la extracción hasta generar las rotaciones del solapamiento, debe hacerse un seguimiento de las máscaras de cada cromosoma a fin que se puedan almacenar como solución de dicho solapamiento. Además, estas máscaras son utilizadas para la verificación de los porcentajes de solapamientos antes mencionados y para evitar que más de dos cromosomas se solapen en una misma zona.

Cabe aclarar que el solapamiento se genera con cromosomas de un mismo cariograma, ya que éstos varían de uno a otro en aspectos tales como tamaño, brillo, fase, etc. Además, los datos se balancean en cuanto al número de veces

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 3.1: Datasets generados.

Nombre	**Cantidad de cromosomas solapados**	**Cariogramas distintos utilizados**
GenMismos2	2	1
Gen5unicoKaryo	2 a 5	1
Gen5variosKaryo	2 a 5	10
Gen5todosKaryo	2 a 5	536
Gen5todosKaryoNorm	2 a 5	536
Gen5todosKaryoNormVar	2 a 5	536

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

que aparece cada clase de cromosoma.

3.2.2. Datasets generados

En la Tabla 3.1 se listan los datasets generados a partir del método explicado. Debe aclararse que en el dataset GenMismos2 se utilizan sólo dos cromosomas fijos de un mismo cariograma, es decir, se generan distintos solapamientos con sólo dos cromosomas preestablecidos. Otro detalle tenido en cuenta en el caso de los tres últimos datasets es que se reservaron 50 cariogramas para la generación exclusiva de datos de test, de forma que se pueda medir el rendimiento de cada red con datos que ninguna haya visto anteriormente.

La diferencia entre los tres últimos radica que en Gen5todosKaryoNorm se normaliza el histograma de cada solapamiento generado asignándole la forma de una función gaussiana con una media y desvío preestablecidos (proceso conocido como histogram matching). Mientras que en el dataset Gen5todosKaryoNormVar se busca agregar variabilidad en los solapamientos, de manera que en el paso de la Figura 3.2 en que se generan distintas rotaciones de un mismo solapamiento, se aplica histogram matching al grupo de píxeles no pertenecientes al fondo con una media de 128, un desvío aleatorio en el rango [50, 70], y un filtro gaussiano de 3x3 con desvío al azar en el rango [0, 0.7]. Si bien se utiliza una media fija de 128, el procedimiento hace que la misma no sea exactamente 128, por lo que se tiene variabilidad también en esa característica. Un ejemplo del proceso puede verse en la Figura 3.3. A simple vista puede que no se distinga variación alguna entre las imágenes, sin embargo la variación numérica que se genera sí es detectada por la red convolucional.

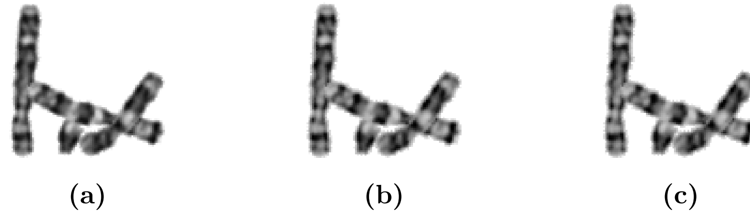


Figura 3.3: Ejemplo de un solapamiento sintético del dataset Gen5todosKaryoNormVar. (a) Cromosomas solapados obtenidos del proceso de la Figura 3.2. (b) Imagen luego de aplicar histogram matching. (c) Resultado de aplicar un filtro gaussiano.

3.3. Arquitecturas consideradas

En esta sección se explican las arquitecturas de redes convolucionales consideradas, dando mayor énfasis en la primera subsección a la elegida finalmente.

3.3.1. Modelo inspirado en U-net

La arquitectura elegida es una variación de la propuesta por Hu [8] e inspirada en U-net [1], que puede observarse gráficamente en la Figura 3.4. La primera modificación que se le hizo al modelo es que la salida sea de 24 canales en lugar de 4, ya que en el trabajo de Hu sólo había 4 clases (fondo, solapamiento y los 2 cromosomas diferentes solapados), mientras que en este trabajo hay 24 categorías (el fondo y los 23 distintos tipos de cromosomas). Estas 23 clases surgen de los 22 pares de cromosomas homólogos y de otra adicional que agrupa al par sexual X e Y, ya que como el objetivo es que a partir de la segmentación se pueda generar el cariograma, no es necesaria la distinción entre ellos. Cabe recalcar que con esto no sólo se generan las máscaras para separar los cromosomas, sino que también se pretende realizar la clasificación de los mismos. El segundo cambio es que a la salida no se le aplica una función softmax, sino que se utiliza la salida lineal debido a que la función de pérdida utilizada en el entrenamiento ya tiene embebida una función similar a la softmax. Sumado a lo anterior, al realizar inferencias simplemente puede tomarse como salida para cada píxel la clase de mayor valor en los 24 canales sin modificar el resultado.

En este tipo de arquitecturas, suele denominarse como codificador a la primera mitad de la red que busca asignar distintas características a la imagen de entrada. En otras palabras, busca representarla con esas características en lugar de los píxeles dados. Por el contrario, la segunda mitad suele llamarse decodificador ya que su objetivo es, a partir de dichas características, determinar a qué categoría pertenece cada píxel. La lógica tras la arquitectura es combinar información local o 'fina' obtenida de las pri-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

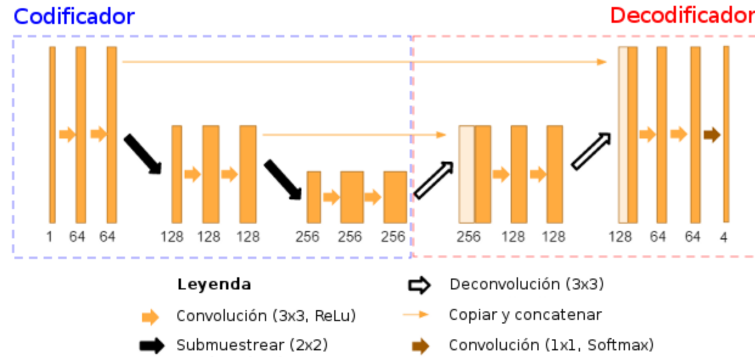


Figura 3.4: Diagrama del modelo propuesto por Hu (adaptado de [8]).

meras capas del codificador con información global o 'gruesa', obtenida al final del mismo. Es decir, a medida que una imagen avanza por el codificador la información se va haciendo cada vez más general, teniendo en cuenta cada vez más sus vecindades. Por lo tanto, en esta arquitectura se busca que el decodificador complemente la información 'gruesa' conseguida al final del codificador con resultados intermedios que se fueron alcanzando en el mismo, a fin de tener en cuenta también la información 'fina'.

Empleando como base el modelo propuesto por Hu (arquitectura original), se realizaron diversas modificaciones tendientes a proporcionar mayor capacidad al modelo neuronal. Se exploraron 4 variantes de esta red:

- Hu: Arquitectura original.
- HuV2: Consiste en una variante del modelo original, al que se le adiciona un nivel de profundidad. Así, al codificador se le adiciona una capa de submuestreo y dos capas de 512 filtros de 3x3, mientras que al decodificador se le añade una capa de sobremuestreo y dos capas de 256 filtros de 3x3 para mantener la simetría.
- HuV3: A HuV2 se le añade nuevamente un nivel de profundidad con dos capas de 512 filtros de 3x3, de forma que quede similar a la arquitectura de la U-net de la Figura 3.1.
- HuV3BN: Se adiciona a HuV3 una capa de normalización luego de cada capa de convolución, lo que se conoce como Batch Normalization.
- HuV3BNN: Idéntica a HuV3BN con el añadido de normalizar los datos de entrada con una media de 0.5 y desvío de 0.5. Es la arquitectura finalmente elegida.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

3.3.2. Otras arquitecturas consideradas

Modelo inspirado en SegNet

La arquitectura propuesta está inspirada en SegNet [37] y corresponde a una simplificación de la misma. Este modelo reducido, denominado mini-SegNet, posee un número menor de filtros en cada capa que la red original (32) y para el sobremuestreo utiliza simplemente una interpolación en lugar de una convolución transpuesta.

Su complejidad es menor a la red Hu de la sección anterior, puesto que posee 47736 parámetros entrenables contra los 1863192 de la red Hu original, por lo que puede utilizarse para comparar si con un modelo más simple puede resolverse el mismo problema.

VGG16

En la Figura 3.5 puede observarse la arquitectura VGG16 propuesta originalmente [38]. La idea tras la misma es utilizar sucesivas capas convolucionales y de submuestreo para obtener características generales de la imagen. Luego, las mismas son conectadas a tres capas de redes neuronales completamente conectadas para su clasificación. Esto hace que esté orientada a la clasificación de la imagen como un todo en una categoría específica, por lo que se reemplazan los bloques completamente conectados para obtener una categoría por cada píxel.

A medida que se fue avanzando con los experimentos, se fueron logrando distintas versiones para realizar la transferencia de aprendizaje planteando diferentes propuestas para reemplazar la capa completamente conectada que se ve en la Figura 3.5 (bloques etiquetados como fully connected+ReLU):

- VGG16v1: se propone conectar a la salida convolucional de la red VGG16 [38] una red completamente conectada similar a la de Figura 3.5 con 1000 neuronas en la capa oculta.

- VGG16v2: se plantea conectar a la última salida convolucional de la VGG16 original un decodificador convolucional que vaya sobremuestreando el mapa de características obtenido de la VGG16 con capas convolucionales que tengan una cantidad de filtros similares a los del decodificador de Hu. El sobremuestreo se realiza mediante convoluciones transpuestas.

- VGG16v3: se modifica el decodificador convolucional de la VGG16v2 para que tenga en cuenta resultados intermedios de la VGG16 original, buscando así combinar, al igual que en el modelo de Hu, información global y local. Se concatenan a la correspondiente capa del decodificador los resultados previos al tercer submuestreo.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

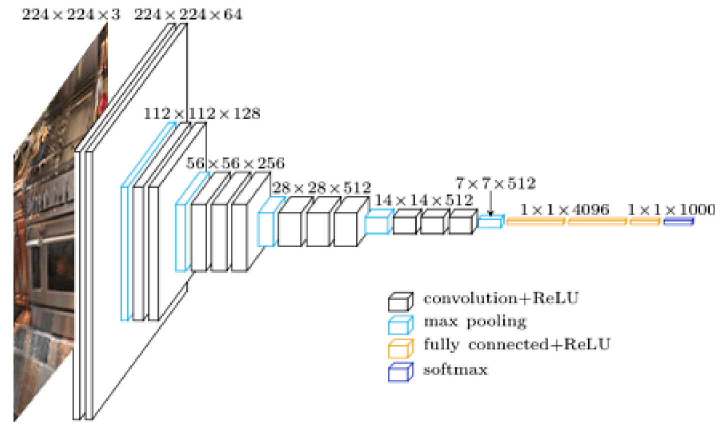


Figura 3.5: Diagrama de la arquitectura VGG16 [39].

- VGG16v4: se modifica nuevamente el decodificador para que utilice todos los resultados obtenidos inmediatamente antes de cada submuestreo, de forma similar a la U-net de la Figura 3.1.

Modelo inspirado en PSPNet

La última arquitectura considerada es la PSPNet [34] que se muestra en la Figura 3.6. En la misma se utiliza un modelo previamente entrenado como extractor de características, en este caso la red VGG16 de la Figura 3.5 justo antes del tercer submuestreo, para que luego un decodificador pueda clasificar los píxeles a partir de las mismas. Dicho decodificador combina filtros con kernels de distinto tamaño con el objetivo de analizar las características obtenidas con VGG16 a diferentes escalas. Esto puede verse también como una combinación de información local (kernels más chicos) con información más global (kernels más grandes). En este trabajo, se utilizan kernels de tamaño 1x1, 3x3 y 5x5, cada uno con 128 canales. Luego, se sobremuestrean y se concatenan para ser analizadas con una capa convolutiva de 3x3 con 128 canales para después pasar por otra de 1x1 con salida de 24 canales con salida lineal.

3.4. Resultados y discusión

La comparación de los diferentes modelos neuronales se realizó empleando los datasets descriptos en la Sección 3.2.2. En todos los casos los datos fueron particionados en 3 grupos: train, validation y test. El entrenamiento se llevó a cabo empleando mini-batch de entre 10 y 25 patrones/imágenes sobre los datos de train (según la capacidad de GPU disponible de acuerdo al tamaño de la red convolucional), mientras que validation fue empleado

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

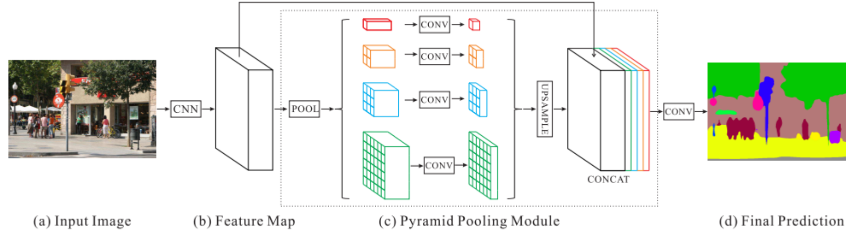


Figura 3.6: Diagrama de la arquitectura PSPNet [34].

para analizar la capacidad de generalización y detener el entrenamiento. En todos los experimentos se utilizó el optimizador Adam [40] con tasa de aprendizaje 10^{-4} y la función de pérdida de entropía cruzada (EC). Esta función convierte las salidas de la red en probabilidades y calcula el error cometido debido a la asignación de la clase incorrecta. Ésta se calcula como:

$$EC(x_i, c_i) = -log\left(\frac{e^{x_i[c_i]}}{\sum_j e^{x_i[j]}}\right), \tag{3.1}$$

donde x_i corresponde al vector de salidas lineales de la red para el i-esimo píxel, y c_i es la clase correcta para el mismo. En base a esto, el error global cometido durante la clasificación de los píxeles de una imagen se determina como el promedio de los valores de entropía cruzada para cada píxel de la imagen.

Luego, la pérdida total de un mini-batch es el promedio de las imágenes que lo componen y ese es el valor utilizado para la actualización de los parámetros de la red convolucional. Al finalizar una época, las pérdidas obtenidas por mini-batch se promedian nuevamente para medir el progreso del entrenamiento de la red.

El análisis del desempeño de la red para la tarea de segmentación se realizó mediante el cálculo del recall y el coeficiente de Jaccard. El primero mide la capacidad que tiene el modelo de distinguir los píxeles correctamente de cada clase, teniendo en cuenta sólo los píxeles que son efectivamente de cada una de ellas. Mientras tanto, el coeficiente de Jaccard incluye en el cálculo para cada clase todos los píxeles de las demás y, por lo tanto, penaliza también por los que estén clasificados como pertenecientes a dicha clase pero que efectivamente no sea así. Los mismos se calculan para cada clase de cada predicción mediante las siguientes fórmulas:

$$recall = \frac{VP}{VP+FN} \tag{3.2}$$

$$jaccard = \frac{VP}{VP+FN+FP} \tag{3.3}$$

donde VP (verdaderos positivos) corresponde a la cantidad de píxeles identificados por el modelo como pertenecientes a la clase en cuestión y que

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 3.2: Resultados con solapamientos de 2 cromosomas fijos.

Modelo	Épocas	Recall	Jaccard
Hu	5	**99,31 %**	**98,65 %**
miniSegNet	20	97,74 %	95,66 %
VGG16v1	50	23,42 %	22,98 %

efectivamente corresponden a la misma, FP (falsos positivos) indica la cantidad de píxeles identificados por el modelo como pertenecientes a la clase en cuestión y que no corresponden a la misma, y FN (falsos negativos) es la cantidad de píxeles identificados por el modelo como no pertenecientes a la clase en cuestión y que sí corresponden a la misma. Para cada mini-batch, los valores de recall y Jaccard se calculan como promedio sobre las mediciones individuales para cada patrón y luego, éstos nuevamente se promedian para cada época de entrenamiento. Los resultados presentados en cada experimento corresponden a la evaluación de las redes sobre el conjunto de test.

3.4.1. Solapamientos de 2 cromosomas fijos

En este experimento se utiliza el dataset más simple llamado GenMismos2, a fin de evaluar qué tan bien funcionan las arquitecturas Hu, miniSegNet y VGG16v1 con un problema sencillo en el que hay solapamientos generados sólo con 2 cromosomas diferentes de un mismo cariograma. Los resultados obtenidos con cada una de las redes consideradas se presentan en la Tabla 3.2. Esta tabla presenta el número de épocas de entrenamiento, el recall y el coeficiente Jaccard para cada modelo analizado. Claramente, el modelo Hu presenta los mejores resultados logrando el mayor recall y Jaccard empleando el menor número de épocas. Por su parte, mientras que miniSegNet alcanza valores de recall y Jaccard cercanos a Hu, VGG16v1 no supera 25 % para ambas medidas y requiere 10 veces más épocas de entrenamiento que Hu. En base a estos resultados, se descartó el modelo VGG16v1 para futuros experimentos.

Los resultados alcanzados por el modelo de Hu podrían explicarse considerando que su arquitectura posee mayor cantidad de parámetros entrenables (1863192) respecto a miniSegNet (47736), lo que se traduce en una mayor capacidad para aprender características de los cromosomas. Mientras que la diferencia entre estas dos respecto a VGG16v1 puede deberse a la causa opuesta: las capas completamente conectadas introducen una gran cantidad de parámetros entrenables (136301072), por lo que probablemente la cantidad de datos de entrenamiento utilizados sean insuficientes para lograr estabilizarlos.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 3.3: Resultados con solapamientos de hasta 5 cromosomas de un mismo cariograma.

Modelo	Épocas	Recall	Jaccard
Hu	10	**98,18 %**	**97,50 %**
miniSegNet	110	36,63 %	26,98 %
VGG16v2	25	29,47 %	24,43 %

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

3.4.2. Solapamientos de hasta 5 cromosomas de un mismo cariograma

En base a los resultados obtenidos en la sección anterior, se realizó un nuevo experimento empleando el dataset Gen5unicoKaryo. El mismo consiste en segmentar imágenes compuestas por hasta 5 cromosomas solapados. Además de las redes Hu y miniSegNet, en este experimento se utiliza la VGG16v2 que incorpora un decodificador convolucional en lugar de capas completamente conectadas. Esto es debido a que las capas completamente conectadas introducen muchos parámetros a entrenar en comparación a las capas convolucionales.

La Tabla 3.3 presenta el número de épocas de entrenamiento, el recall y Jaccard para cada modelo considerado. Claramente, el modelo Hu presenta los mejores resultados logrando el mayor recall y Jaccard empleando el menor número de épocas. Por su parte, miniSegNet y VGG16v2 emplean un mayor número de épocas de entrenamiento, logrando un recall menor a 40 % y un Jaccard menor a 30 %. Por ello, ambas son descartadas para los próximos experimentos.

La red VGG16v2 no tiene la capacidad de reconstruir las máscaras a partir de las características que extrae. Por ende, el problema puede ser que no tenga la información suficiente para hacerlo ya que utiliza toda la información global (obtenida al final de la extracción de características) pero no incorpora información local (obtenida en los pasos intermedios, más cercana a la imagen de entrada) como lo hacen las otras dos. La red miniSegNet es muy pequeña (47736 parámetros entrenables) respecto a la de Hu (1863192 parámetros), por lo que el problema es más complejo de lo que puede resolver. Por más que entrenó por más de 100 épocas, aprendió a resolver algunas clases de cromosomas pero no todas. En resumen, la red Hu da mejores resultados por combinar la información mejor que VGG16v2 y por tener mayor complejidad que miniSegNet.

3.4.3. Solapamientos de hasta 5 cromosomas provenientes de 10 cariogramas

En este experimento se incorporan solapamientos de 2, 3, 4 y 5 cromosomas tomados de 10 cariogramas elegidos al azar, correspondientes al dataset

Tabla 3.4: Resultados con solapamientos de hasta 5 cromosomas provenientes de 10 cariogramas.

Modelo	Épocas	Recall	Jaccard
Hu	100	**93,18 %**	**90,38 %**
PSPNet	20	25,82 %	20,97 %
VGG16v3	30	39,52 %	34,78 %

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

llamado Gen5variosKaryo. El hecho que se mezclen solapamientos generados con diferentes cariogramas hace que el problema sea más complejo, ya que la red debe adaptarse a las variaciones de brillo, tamaño y curvaturas que hay de un cariograma a otro.

Una vez más, se modifica la red VGG16 para llegar a la VGG16v3 (ver Sección 3.3.2). A partir del análisis del experimento anterior, se incorpora a la mencionada red resultados intermedios obtenidos previo al tercer submuestreo de la red VGG16 original de la Figura 3.5. Además, se incorpora el modelo PSPNet debido a que se observa que hasta el momento los mejores resultados obtenidos son gracias a la combinación de información global y local, tal como hace el modelo de Hu. Este modelo realiza algo similar pero con un enfoque distinto ya que lo que hace es analizar las características a distintas escalas, es decir, con diferentes tamaños de kernels.

En la Tabla 3.4 se muestran los resultados obtenidos para cada modelo junto con la cantidad de épocas que fueron entrenados. Nuevamente, se observa que el mejor desempeño tanto en recall como en Jaccard es obtenido con la red Hu. Aunque fue entrenada durante 100 épocas, cabe destacar que a las 40 épocas ya había alcanzado un recall y un Jaccard de aproximadamente 90 % de validación en entrenamiento, pero se continuó con el mismo hasta las 100 épocas para verificar qué tanto podía mejorar. Respecto a las redes PSPNet y VGG16v3, aunque empleen un menor número de épocas, no superan el 40 % en recall ni el 35 % en Jaccard. Debido a ello, ambas se descartan para los próximos experimentos.

Probablemente la forma en que la red PSPNet combina la información global y local, con distintos tamaños de kernels en lugar de reutilizar resultados previos como Hu y VGG16v3, no sea la más adecuada para este problema. La red VGG16v3 puede justificar su mal rendimiento a que la información intermedia concatenada en la decodificación no sea suficiente, a diferencia de la red Hu que reutiliza todos los resultados intermedios previos a un submuestreo. De esta forma, se concluye que la forma en que Hu conecta las salidas intermedias es la más acertada.

En este punto, se realiza una prueba con imágenes de solapamientos de cromosomas provenientes de otros cariogramas de los utilizados para entrenar, a fin de comprobar qué tan bien generaliza la red. En otras palabras, podría ser que los 10 cariogramas tomados sean suficientemente representati-

Tabla 3.5: Resultados con solapamientos de hasta 5 cromosomas provenientes de todos los cariogramas.

Modelo	Dataset	Épocas	Recall	Jaccard
Hu	Gen5todosKaryo	30	71,61 %	60.24 %
Hu	Gen5todosKaryoNorm	24	71,24 %	59,85 %
Hu	Gen5todosKaryoNormVar	20	72,48 %	65,58 %
VGG16v4	Gen5todosKaryoNormVar	13	58,31 %	49,84 %
HuV2	Gen5todosKaryoNormVar	17	84,16 %	78,87 %
HuV3BN	Gen5todosKaryoNormVar	9	86,91 %	80,70 %
HuV3BNN	Gen5todosKaryoNormVar	12	**88,71 %**	**83,47 %**

vos de los datos como para que la red sepa resolver solapamientos generados a partir de otros cariogramas. Utilizando los datos de test de Gen5todosKaryo se obtuvo 33,15 % de recall y 25,84 % de Jaccard con la red Hu entrenada en este experimento. Por lo tanto, se comprueba que la red convolucional aprendió características demasiado específicas de los 10 cariogramas tomados y no es capaz de generalizar a otros casos.

3.4.4. Solapamientos de hasta 5 cromosomas provenientes de todos los cariogramas

Del experimento anterior se desprende que la red no resuelve solapamientos generados a partir de otros cariogramas de los considerados para entrenar la misma. Por lo tanto, en este experimento se prueba sobre los datasets que utilizan todos los cariogramas para generar solapamientos a fin de comprobar si una mayor variedad de cromosomas se traduce en una mejor capacidad de generalización de la red. En un principio, se mantuvo el modelo de Hu original y se varió la forma en que se generaron los datos. Luego, se mantuvo fijo el dataset que mejores resultados dio y se exploraron arquitecturas cada vez más complejas.

Los resultados obtenidos con cada una de los datasets y redes considerados se presentan en la Tabla 3.5. Se observa en las tres primeras filas de la tabla que se obtienen mejores resultados tanto en recall como en Jaccard y en una menor cantidad de épocas con el dataset Gen5todosKaryoNormVar. Sin embargo, los resultados obtenidos en recall y coeficiente de Jaccard son relativamente bajos así que se experimentó con otras arquitecturas. La red VGG16v4 tiene un menor desempeño en ambas medidas, mientras que la red HuV2 mejora más de un 10 % tanto el recall como el Jaccard. Esta última lo logra añadiendo un nivel de profundidad respecto a Hu, por lo que el paso siguiente es añadir otro más para alcanzar HuV3BN y HuV3BNN, que incluyen Batch Normalization con el objetivo de acelerar el entrenamiento. Se observa que los mejores resultados en recall y Jaccard son obtenidos con

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

HuV3BNN, con un aumento de 2 % en ambas medidas respecto a HuV3BN a un costo de 3 épocas de entrenamiento extra.

El dataset Gen5todosKaryoNormVar se distingue de Gen5todosKaryo y Gen5todosKaryoNorm por hacer variar las características de los solapamientos sintéticos, lo que hace que la red aprenda características más generales de los mismos y consecuentemente mejore el desempeño de la misma. Por otro lado, las redes HuV3BN y HuV3BNN obtienen mejores resultados por tener una mayor cantidad de parámetros entrenables (aproximadamente 20 millones) respecto a HuV2 (8 millones), Hu (casi 2 millones) y VGG16v4 (14 millones). Esta última también puede justificar su mal desempeño en que fue entrenada con un dataset de otro problema y utilizada mediante transfer learning en este trabajo, por lo que probablemente las características que extraiga de las imágenes no sean las más adecuadas. Por último, comparando HuV3BN y HuV3BNN se concluye que la capa de normalización de las imágenes de entrada provoca un mejor rendimiento de la red, debido a que el hecho de que todos los datos estén bajo un mismo rango probablemente logra que el modelo pueda aprender mejor las diferencias entre clases.

3.4.5. Evaluación del mejor método con cromosomas reales

En este experimento se busca evaluar el funcionamiento de la red convolucional HuV3BNN entrenada con solapamientos sintéticos en cromosomas reales, para lo que se hará una inspección visual sobre las predicciones que hace la red sobre dichos cromosomas reales ya que no se tiene el ground truth de los mismos. Para ello, se toman 60 clusters de cromosomas al azar que fueron segmentados con el módulo de pre-procesamiento explicado en el Capítulo 2. A continuación se toma cada clúster y se le aplica el mismo proceso de histogram matching mencionado en la generación de datos con media 128 y desvío 60, para luego pasarle un filtro gaussiano de 3x3 con desvío 0,7. El objetivo de ello es que los cromosomas reales crudos se parezcan más a los cromosomas provenientes de los cariogramas utilizados para el entrenamiento de la red.

En 42 de los clusters, la segmentación se realiza correctamente y no se necesitaría ningún tipo de post-procesamiento. En la Figura 3.7 pueden observarse alguno de estos casos. En las filas impares pueden verse los clusters de cromosomas que se utilizaron de entrada a la red y en las filas pares la separación que dio como salida dicha red con distintos colores acordes a la clase que pertenezcan, que se van a mantener a lo largo de las distintas figuras.

Luego, en la Figura 3.8 se observan los 9 casos en que la predicción no es tan precisa pero que podría aplicarse algún método de post-procesamiento, como por ejemplo la detección de pequeñas zonas de píxeles erróneas y su corrección según los píxeles vecinos a dichas zonas. Este enfoque de post-procesamiento no funcionaría para las dos primeras imágenes de la primer

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

fila de la Figura 3.8 debido a que los píxeles clasificados incorrectamente son vecinos a píxeles del mismo tipo bien etiquetados. Sin embargo, el error es pequeño y por lo tanto aceptable.

En las Figura 3.9 se muestran los 9 clusters de cromosomas reales que la red no fue capaz de separar correctamente. En la primera fila se ven clusters de 4 o 5 cromosomas y se observa que el modelo no puede resolver sólo uno o dos cromosomas, por lo que al menos podría rescatarse el resultado parcialmente correcto. En la tercera fila se ven casos en que la red asigna incorrectamente las clases a un lado y otro de la zona de solapamiento, por lo que tal vez podría aplicarse algún tipo de post-procesamiento más complejo para solucionarlo. Por último, en la quinta fila se muestran dos casos similares a los anteriores aunque un poco más difíciles de resolver por tener más cantidad de píxeles érroneos y en la tercera imagen se ve un caso que la red no puede resolver: cuando hay un solapamiento de dos cromosomas de una misma clase.

También se probó la red con 20 cromosomas reales individuales elegidos al azar y los resultados fueron muy buenos. En la Figura 3.10 se ven 12 de los 18 casos resueltos correctamente por la red sin necesidad de post-procesamiento, mientras que en la Figura 3.11 se ven los 2 que no fue capaz de hacerlo completamente. Aunque, además de ser pocos los casos érroneos, se podría aplicar un post-procesamiento que los corrija de forma similar al ya mencionado sobre el análisis de píxeles vecinos.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

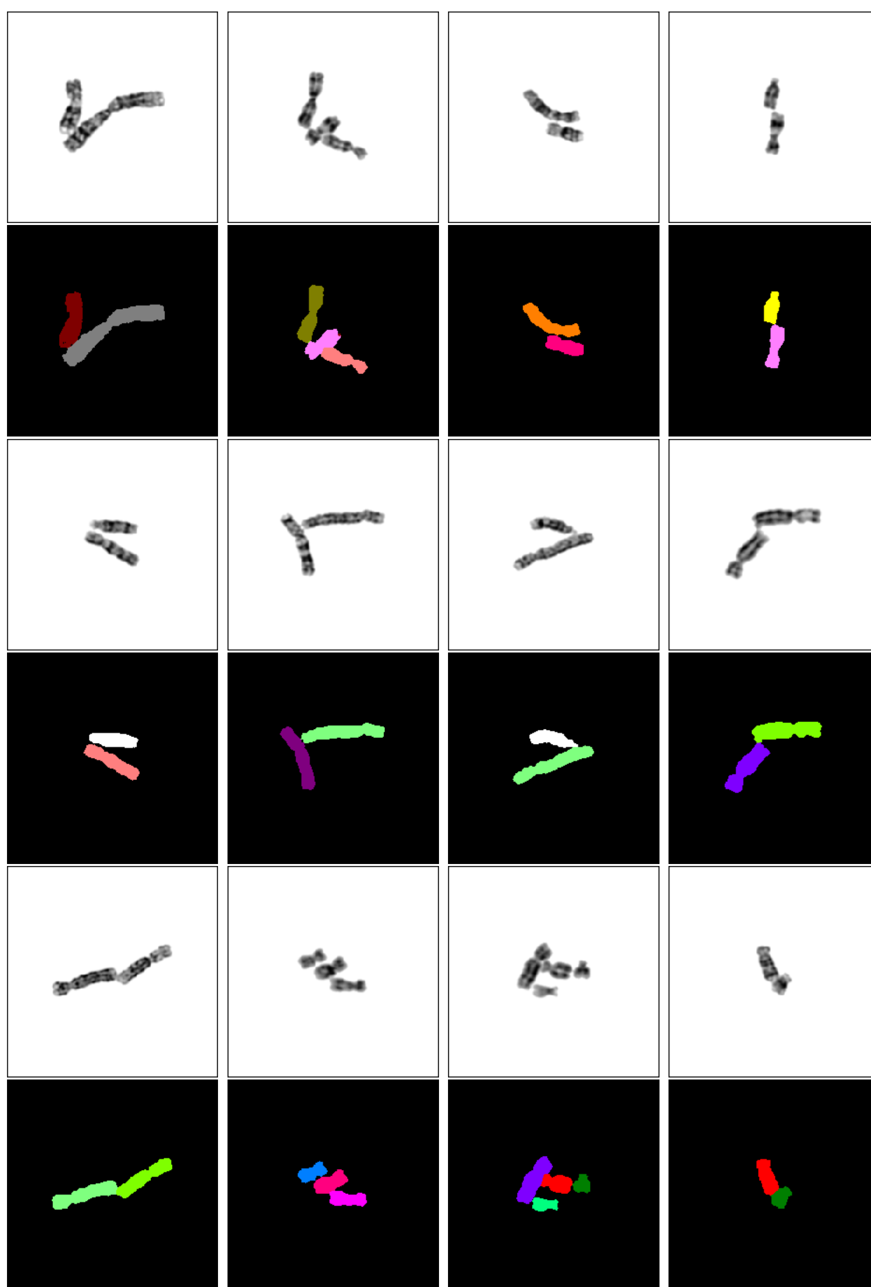


Figura 3.7: Predicciones de clusters de cromosomas reales que no requieren post-procesamiento.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

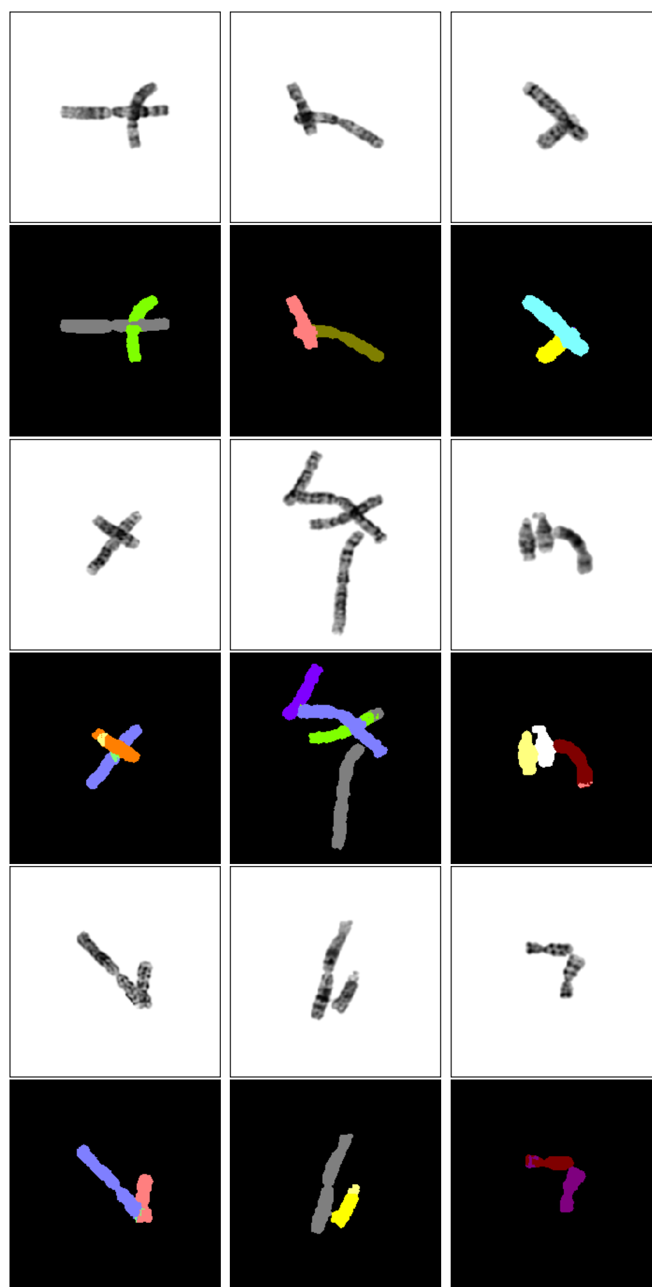


Figura 3.8: Predicciones de clusters de cromosomas reales con resultado aceptable que requieren un post-procesamiento simple.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

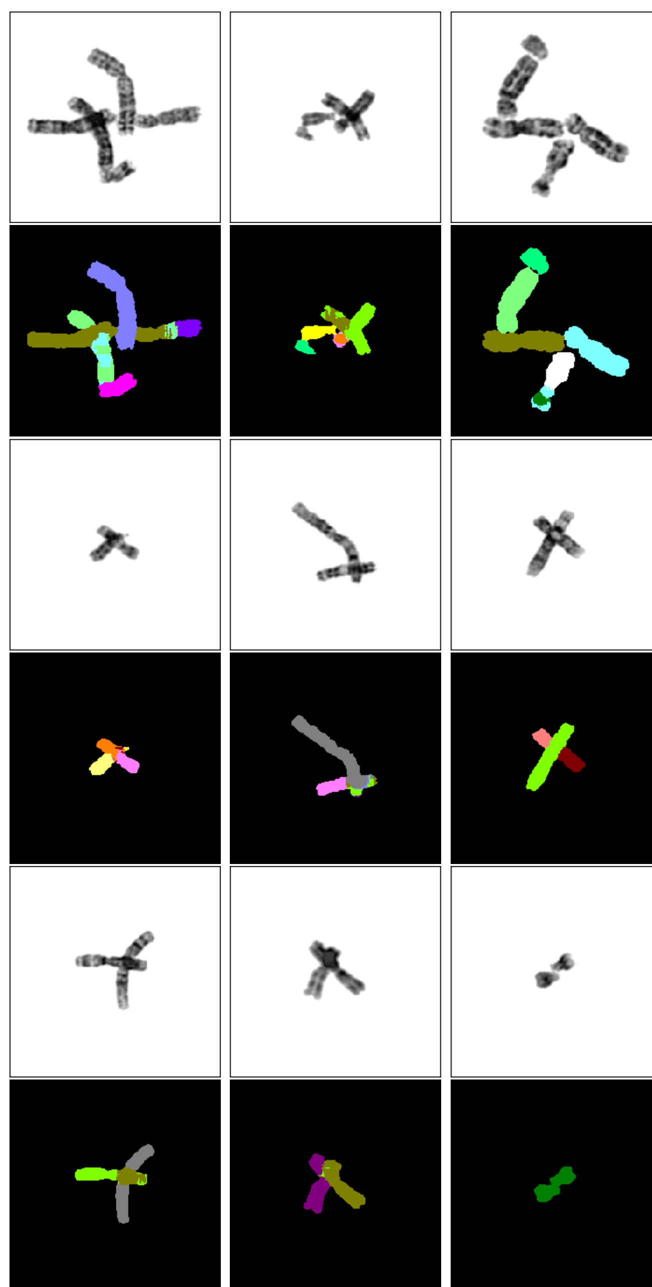


Figura 3.9: Predicciones de clusters de cromosomas reales que requerirían un post-procesamiento más complejo.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

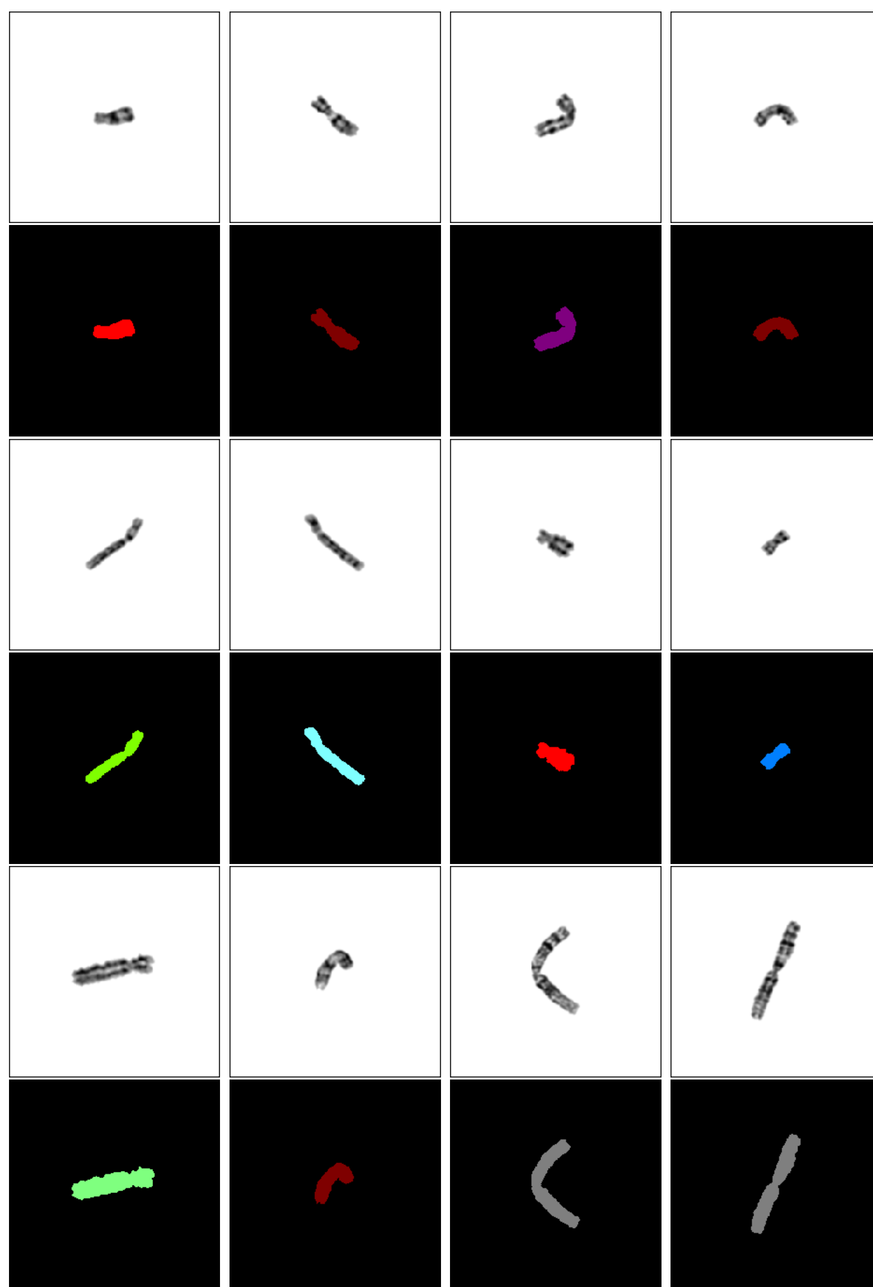


Figura 3.10: Predicciones de cromosomas reales con buen resultado y que no requieren post-procesamiento.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

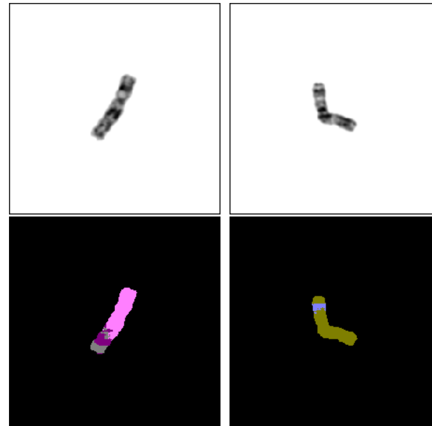


Figura 3.11: Predicciones de cromosomas reales individuales erróneas y que requieren post-procesamiento.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

3.5. Conclusiones

En este capítulo se abordó el problema de la separación de cromosomas solapados con el uso de una red convolucional entrenada con datos generados sintéticamente. Se varió la forma en que se generaron los mismos y se demostró que los mejores resultados se obtienen cuanto más variabilidad se le agregan a los datos. Así, la red se centra en el aprendizaje de características más generales de los cromosomas para su separación. Además, se demostró que se obtienen buenos resultados con cromosomas reales teniendo en cuenta que el entrenamiento de la red se hizo con datos sintéticos y que éstos difieren entre sí. Esto es porque a los cromosomas reales se les aplicó un procesamiento desconocido para llevarlos a su correspondiente cariograma.

En los experimentos se fue aumentando la complejidad del problema a resolver, agregando más cromosomas a los solapamientos y teniendo en cuenta una mayor cantidad de cariogramas. Se exploraron varias arquitecturas diferentes que varían la forma en que combinan la información, la cantidad de parámetros entrenables y la profundidad. Para la comparación de resultados, se utilizaron medidas objetivas tales como el recall y el coeficiente de Jaccard. Los mejores resultados se obtuvieron con la arquitectura HuV3BNN que es similar a U-net, la que se basa en guardar los pasos intermedios del codificador para luego al decodificar poder combinar la información global obtenida al submuestrear con la información local obtenida en los pasos anteriores. Además, se incorporó al modelo capas de normalización por batch en la salida de cada capa convolucional con el objetivo de acelerar el entrenamiento de la misma.

Resulta interesante destacar que la red convolucional se entrenó exclusivamente con cromosomas solapados, es decir, que al menos había dos cromosomas. Sin embargo, se observó que ésta también clasifica de forma correcta imágenes de cromosomas individuales. Estos resultados indicarían que no es necesario el paso de diferenciación entre cluster y cromosoma individual mencionado en 1.4.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Capítulo 4

Post-procesamiento

A partir de los resultados obtenidos en la separación de solapamientos, se observó que la aplicación de algún método de post-procesamiento podría mejorar de forma importante la calidad de las segmentaciones. Por lo tanto, en este capítulo se aborda el problema de corregir las máscaras obtenidas con la red convolucional de separación. Los métodos utilizados en este capítulo se pueden clasificar en dos tipos: los que no involucran el re-entrenamiento de una red convolucional y los que sí lo hacen. Dentro de los primeros se encuentran la eliminación de pequeñas imperfecciones analizando las vecindades, la corrección basada en las probabilidades de los canales de salida de la red convolucional de separación, y la corrección mediante el algoritmo k-NN. En los del segundo tipo, se propone acoplar a la salida de la red de separación otra red con una arquitectura similar, variando el modo de entrenarla. Los resultados se analizan considerando las mismas medidas que en experimentos previos y una nueva denominada *confiabilidad*, la cual indica la probabilidad de que el número de cromosomas segmentados sea efectivamente el correcto.

El capítulo se organiza de la siguiente manera. La Sección 4.1 explica los diferentes métodos que se usan para el post-procesamiento y la lógica tras ellos. Luego, en la Sección 4.2 se realizan experimentos para evaluar el resultado de aplicar los mencionados métodos combinados de distintas maneras y variando sus parámetros. Finalmente, en la Sección 4.3 se presentan las conclusiones del capítulo. Además, en el Apéndice A se encuentra la documentación del código desarrollado.

4.1. Métodos propuestos

A continuación se explicará cada uno de los métodos utilizados. El primero de ellos se aplica una vez que se le asignó a cada píxel la clase de mayor probabilidad. Los siguientes tres usan la información de probabilidad de los 24 canales de salida de la red convolucional previo a la asignación de una

clase a cada píxel. Él último de ellos utiliza otra red convolucional, por lo que es el único que requiere re-entrenamiento.

4.1.1. Eliminación de pequeñas imperfecciones

El método más sencillo consiste en corregir, dentro de un conjunto de píxeles de una misma clase, pequeñas zonas de píxeles de una clase diferente que no tienen el tamaño suficiente para ser un cromosoma. Como estas imperfecciones pueden ser consecuencia de errores de la predicción de la red o de la imagen de entrada, la propuesta consiste en analizar la vecindad de las zonas de píxeles desechadas para asignarle una clase.

Para la determinación del umbral para el área, se analizan los cromosomas del dataset de clase 21 por ser los de menor tamaño [32]. Se observa que los mismos tienen una media de aproximadamente 391 píxeles, siendo su menor tamaño 213 píxeles. Por lo tanto, se determina que un umbral de 100 sería adecuado para no descartar este tipo de cromosomas con este método. Luego, para definir a que clase pertenecen las zonas a corregir, se observa cuál es la clase mayoritaria en los píxeles inmediatamente vecinos.

4.1.2. Corrección por canales según umbral para la determinación de clases confiables

El objetivo del método es descartar información de salida de la red convolucional que puede considerarse como ruidosa. Las clases confiables se definen como aquellas para las que al menos un píxel posee una probabilidad de pertenencia a dicha clase mayor que un umbral establecido. Estableciendo un umbral suficientemente elevado se puede lograr reducir el error de aparición de clases incorrectas. Para cada píxel, las clases con probabilidades inferiores al umbral son descartadas (probabilidad seteada como 0) y se recalculan las probabilidades para las clases restantes.

La Figura 4.1 presenta un ejemplo de este método de corrección. Por simplicidad, cada píxel de la imagen (indicados con números del 1 al 4) puede ser asignado a una de 10 clases disponibles (indicadas en letras desde la A a la J). En la parte superior de la Figura 4.1 se resalta en verde la clase a la que sería asignado cada píxel si no se aplicara una corrección. Suponiendo un umbral de 0,8, sólo las clases A, G y H en los píxeles de la figura superior poseen probabilidades superiores a dicho umbral. También puede observarse que para el tercer píxel, ninguna de las clases supera dicho umbral. Al aplicar el método propuesto, sólo las probabilidades asociadas a las clases A, G y H son conservadas para cada píxel (se hace 0 el resto), y sus valores son normalizados para que la suma de probabilidades para cada uno vuelva a sumar la unidad. Así, para el primer píxel puede observarse que la probabilidad para la clase A es 1 y que la asignación de clase para el píxel es la misma que antes de aplicar el método. Mientras que para el tercer píxel, se

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

	A	B	C	D	E	F	G	H	I	J
1	0,9	0	0	0,05	0	0,05	0	0	0	0
2	0,05	0	0	0	0,01	0	0,9	0	0	0,04
3	0	0	0,05	0,05	0	0,4	0,3	0,1	0	0
4	0,05	0,05	0	0	0	0	0,05	0,85	0	0

	A	B	C	D	E	F	G	H	I	J
1	1	0	0	0	0	0	0	0	0	0
2	0,05	0	0	0	0	0	0,95	0	0	0
3	0	0	0	0	0	0	0,75	0,25	0	0
4	0,05	0	0	0	0	0	0,05	0,9	0	0

Figura 4.1: Ejemplo del método de corrección por canales con un umbral de 0,8. En verde se indica la clase elegida para cada píxel.

ve que al aplicar el procedimiento explicado se hace 0 la probabilidad que era mayor (clase F) y se asigna la clase G. En este último caso, se logra reducir el número de clases de la imagen eliminando las probabilidades relativamente bajas del tercer píxel que no pertenecen a las clases confiables y así, reducir el ruido presente en la predicción.

4.1.3. *k*-NN mediante el uso de un umbral para determinar los píxeles confiables

Este método también se basa en el uso de un umbral para determinar los píxeles de clases confiables al igual que el de corrección por canales de la Sección 4.1.2. Los mencionados píxeles se etiquetan como referentes de las clase confiable a la cual pertenecen. Por ejemplo, tomando como umbral 0,8, en la parte superior de la Figura 4.1, el píxel 1 sería referente de la clase A, el píxel 2 de la clase G y el píxel 4 de la clase H. Luego, se clasifica el resto de los píxeles de acuerdo con su parecido a los píxeles confiables tomados como referencia. Siguiendo con el ejemplo de la parte superior de la Figura 4.1, al píxel 3 debería asignársele una clase según su parecido a los píxeles 1, 2 y 4.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Para la asignación de clases se utiliza el algoritmo de los k vecinos más cercanos (k-NN, k-Nearest Neighbors) [41], que consiste en definir objetos representativos de cada clase y luego, para cada objeto a clasificar, se le asigna una clase según la clase de los k vecinos más cercanos en el espacio n-dimensional. En este caso, el espacio es de 24 dimensiones correspondiente a la cantidad de canales para cada píxel, $k = 5$ y la distancia utilizada es la euclidiana, dada por la fórmula:

$$d(x_i, x_j) : \sqrt{\sum_{r=1}^{24} (x_i^r - x_j^r)^2}, \tag{4.1}$$

donde x_i y x_j son dos píxeles y el superíndice r denota cada uno de sus valores para las 24 dimensiones. El algoritmo se entrena almacenando los 24 canales de los píxeles confiables, que luego se usan para predecir la clase de los no confiables según la clase predominante en los k vecinos más cercanos.

4.1.4. Red convolucional W

A la salida de la red HuV3BNN utilizada para la separación de solapamientos en el Capítulo 3, se acopla otra instancia de la misma red que toma como entrada la salida de los 24 canales de la primera, previamente procesados para transformarlos en probabilidades. El conjunto se llamará Red W y se referirá como primer y segunda U a cada instancia de la red HuV3BNN, respectivamente.

La propuesta consiste en que la segunda U aprenda a corregir los errores de la primera, a partir del entrenamiento con datos sintéticos en las mismas condiciones que los experimentos del Capítulo 3. En un primer caso se prueba entrenando sólo la segunda U, manteniendo la primera fija según el entrenamiento realizado en la separación y luego se experimenta entrenándola completamente.

4.2. Resultados y discusión

En los resultados presentados en la Sección 3.4, las medidas primero se promediaron por mini-batch y luego para la totalidad de los datos (suponiendo un mini-batch igual a 10, primero se promedian las medidas de esas 10 imágenes para después calcular la media de dichos promedios). En esta sección, se calculan como el promedio de la totalidad de los datos (equivalente a que el mini-batch sea igual a 1). Además, para mejorar el análisis se determina el intervalo de confianza (IC) con 95 % de confianza de las medias de las métricas [42]. Nuevamente se emplean las medidas de recall y coeficiente de Jaccard para analizar las mejoras obtenidas con cada método o combinación de ellos. También se incorpora una medida denominada confiabilidad,

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

que indica la probabilidad de que el número de cromosomas obtenidos luego de la segmentación sea efectivamente el esperado. Así, una confiabilidad de 98 % para el caso de segmentaciones que producen 2 cromosomas indica que en el 98 % de los casos los clusters que generen 2 cromosomas efectivamente estaban compuestos por dos de ellos. Para ello, en el caso de dos cromosomas recién descripto, se calcula mediante la siguiente fórmula:

$$confiabilidad = \frac{\text{cantidad de solapamientos predichos con dos cromosomas} \atop \text{y que efectivamente era así}}{\text{cantidad de solapamientos predichos con dos cromosomas}}$$

De la misma forma se calcula para tres, cuatro y cinco cromosomas solapados y para el total de ellos.

Para reducir los tiempos de ejecución de los experimentos, los primeros cuatro de ellos se realizan utilizando un grupo de 2688 imágenes del dataset Gen5todosKaryoNormVar (que corresponde a los datos usados como test en la Sección 3.4) luego de pasar por la red HuV3BNN, ambos explicados en el Capítulo 3 sobre la separación de solapamientos. En el quinto experimento de la Sección 4.2.5 se varía la red convolucional y los datos de entrenamiento pero el dataset utilizado para test es el recién mencionado. Para el experimento de la Sección 4.2.6, se genera un nuevo dataset de 2688 imágenes bajo las mismas condiciones que Gen5todosKaryoNormVar con el fin de conocer el desempeño de los métodos en datos no vistos anteriormente y así poder comparar su rendimiento.

4.2.1. Eliminación de pequeñas imperfecciones

En este experimento se prueba el método de eliminación de pequeñas imperfecciones (EPI) explicado en la Sección 4.1.1. Con ese objetivo, se calculan las medidas previamente explicadas, antes y después de la aplicación del mismo. Los resultados obtenidos se presentan en la Tabla 4.1, discriminando la confiabilidad obtenida según la cantidad de cromosomas presente.

Claramente, la principal ventaja de esta corrección se observa en los valores de confiabilidad. Puede verse que mientras los valores de recall y Jaccard no sufren modificaciones, la confiabilidad en la predicción del número de clases se incrementa notablemente. Hay sólo una pequeña reducción en la confiabilidad en el caso de dos cromosomas solapados, pero este pequeño costo es compensado con creces con el aumento de la misma en los casos con más cromosomas.

4.2.2. Umbrales para detectar clases confiables

El objetivo de este experimento es determinar el mejor umbral para la determinación de las clases confiables, es decir, las clases que contengan píxeles con probabilidad mayor a un umbral. Para ello, se calcula la confiabilidad de cada predicción para los distintos umbrales.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 4.1: Resultados de la eliminación de pequeñas imperfecciones.

Método	**Ninguno**	**EPI**
Recall	$88,8 \pm 0,6\,\%$	$\mathbf{89,1 \pm 0,6\,\%}$
Jaccard	$87,1 \pm 0,6\,\%$	$87,1 \pm 0,6\,\%$
Confiabilidad		
2 cromosomas	$\mathbf{98,9 \pm 1,1\,\%}$	$97,8 \pm 1,3\,\%$
3 cromosomas	$64,7 \pm 4,8\,\%$	$\mathbf{80,2 \pm 3,2\,\%}$
4 cromosomas	$39,3 \pm 4,6\,\%$	$\mathbf{69,4 \pm 3,6\,\%}$
5 cromosomas	$26,7 \pm 4,2\,\%$	$\mathbf{63,3 \pm 3,8\,\%}$
Total	$55,5 \pm 2,4\,\%$	$\mathbf{77,0 \pm 1,7\,\%}$

Tabla 4.2: Resultados de aplicar umbrales para detectar clases confiables

Umbral	**0,80**	**0,85**	**0,90**	**0,95**	**0,98**
2 cromos.	$\mathbf{96,7 \pm 1,3\,\%}$	$\mathbf{96,7 \pm 1,3\,\%}$	$95,0 \pm 1,6\,\%$	$90,7 \pm 2,2\,\%$	$83,1 \pm 2,8\,\%$
3 cromos.	$76,2 \pm 3,2\,\%$	$77,5 \pm 3,2\,\%$	$\mathbf{78,3 \pm 3,1\,\%}$	$77,2 \pm 3,2\,\%$	$75,0 \pm 3,3\,\%$
4 cromos.	$63,4 \pm 3,6\,\%$	$67,5 \pm 3,5\,\%$	$70,3 \pm 3,5\,\%$	$\mathbf{73,2 \pm 3,3\,\%}$	$70,3 \pm 3,5\,\%$
5 cromos.	$58,5 \pm 3,7\,\%$	$63,0 \pm 3,7\,\%$	$71,3 \pm 3,4\,\%$	$79,7 \pm 3,0\,\%$	$\mathbf{87,6 \pm 2,5\,\%}$
Total	$73,0 \pm 0,3\,\%$	$75,5 \pm 0,3\,\%$	$78,5 \pm 0,3\,\%$	$\mathbf{80,1 \pm 0,3\,\%}$	$78,3 \pm 0,3\,\%$

En la Tabla 4.2 se presentan los cálculos de confiabilidad para diferentes umbrales y diferenciados de acuerdo a la cantidad de cromosomas involucrados en el solapamiento. A medida que aumenta el número de cromosomas, se observa un incremento de la confiabilidad con el crecimiento del umbral. Mientras que en el total, se nota que el umbral que provee una mayor confiabilidad total es el $0,95$ con $1,5\,\%$ de diferencia con $0,9$. Sin embargo, se concluye que es más conveniente el umbral $0,9$ debido a que provee un mejor rendimiento en los casos de 2 y 3 cromosomas respecto al de $0,95$, los cuales se dan mucho más en situaciones reales. A pesar de esto, en los siguientes experimentos también se prueba con otros umbrales dependiendo de cada situación.

4.2.3. Corrección por canales según umbral para determinar las clases confiables

En este experimento se evalúa el método de corregir los canales (CC) de la salida de la red convolucional explicado en la sección 4.1.2, nuevamente variando los umbrales. Luego, como ya se explicó, se predice la clase para cada píxel según la clase que tenga mayor probabilidad. Al resultado se le aplica el método EPI de la sección 4.1.1 puesto que ya se demostró su buen desempeño en la Sección 4.2.1.

En la Tabla 4.3 se presentan los resultados obtenidos, mostrando en la parte superior el recall y Jaccard para diferentes umbrales. En la parte inferior se observa la confiabilidad discriminada por el número de cromosomas presente en el solapamiento y la confiabilidad total. Como puede observarse,

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 4.3: Resultados de la corrección por canales según umbral para determinar las clases confiables. En superíndice se indica en porcentaje el umbral utilizado.

Método	**sólo EPI**	**CC^{70}**	**CC^{80}**	**CC^{90}**	**CC^{95}**
Recall	$89,1 \pm 0,6\,\%$	$89,2 \pm 0,6\,\%$	$89,2 \pm 0,6\,\%$	**$90,0 \pm 0,6\,\%$**	$88,5 \pm 0,6\,\%$
Jaccard	$87,1 \pm 0,6\,\%$	**$87,2 \pm 0,6\,\%$**	$87,1 \pm 0,6\,\%$	$86,6 \pm 0,7\,\%$	$85,5 \pm 0,7\,\%$
Confiabilidad por cantidad de cromosomas					
2	**$97,8 \pm 1,3\,\%$**	$97,6 \pm 1,3\,\%$	$97,0 \pm 1,4\,\%$	$95,0 \pm 1,8\,\%$	$90,8 \pm 2,3\,\%$
3	$80,2 \pm 3,2\,\%$	$80,3 \pm 3,2\,\%$	**$80,8 \pm 3,1\,\%$**	$80,4 \pm 3,0\,\%$	$77,8 \pm 3,1\,\%$
4	$69,4 \pm 3,6\,\%$	$70,4 \pm 3,6\,\%$	$72,5 \pm 3,5\,\%$	$73,8 \pm 3,4\,\%$	**$74,5 \pm 3,3\,\%$**
5	$63,3 \pm 3,8\,\%$	$66,0 \pm 3,8\,\%$	$69,6 \pm 3,7\,\%$	$76,0 \pm 3,5\,\%$	**$82,5 \pm 3,2\,\%$**
Total	$77,0 \pm 1,7\,\%$	$78,0 \pm 1,7\,\%$	$79,6 \pm 1,6\,\%$	$81,1 \pm 1,5\,\%$	**$81,2 \pm 1,5\,\%$**

no hay mejoras significativas en el recall con el aumento del umbral. Incluso, por encima de 0,9 el Jaccard disminuye significativamente. Por el lado de la confiabilidad, los mejores resultados para clusters de 4 y 5 cromosomas se obtienen empleando umbrales altos (0,95). Sin embargo, los resultados muestran que para clusters con menos cromosomas la situación es opuesta, siendo los umbrales más bajos la opción que mejora la confiabilidad. Esto hace pensar que la red convolucional se confunde más frecuentemente de clases cuanto mayor sea la cantidad de cromosomas presentes en el cluster, necesitando umbrales mayores para corregirlo. Estos resultados refuerzan la conclusión obtenida en la Sección 4.2.2: el umbral más conveniente es el 0,9 ya que mejora la confiabilidad para 4 y 5 cromosomas con una disminución relativamente baja en 2 y 3 y evitando la caída abrupta del Jaccard.

4.2.4. *k*-NN mediante el uso de un umbral para determinar las clases confiables

El objetivo de este experimento es comprobar el desempeño del método k-NN explicado en la Sección 4.1.3. La evaluación se realiza utilizando un umbral de 0,9, dado que los mejores resultados fueron obtenidos con este valor en experimentos previos. Luego, se evalúa el rendimiento del k-NN en combinación con el método de corrección por canales. La lógica de la propuesta es que CC elimine el ruido presente en los canales y que después k-NN mejore la calidad de la segmentación asignando los píxeles poco confiables a una clase confiable, como se explico en la Sección 4.1.3. Se varían los umbrales tanto de CC como de k-NN ya que la modificación en uno de los métodos podría hacer que el otro tenga mejor rendimiento. En todos los casos, se aplica como última instancia EPI puesto que se demostró en 4.2.1 su buen desempeño.

En la Tabla 4.4 se presentan los resultados más representativos obtenidos, mostrando de forma similar a los experimentos anteriores el recall, el Jaccard y la confiabilidad para distintos umbrales. En cuanto a la calidad de la segmentación, en todos los casos se observa que el recall y el Jaccard casi no varían respecto a lo medido inicialmente sin aplicar ningún post-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 4.4: Resultados de k-NN mediante el uso de un umbral para determinar las clases confiables

Método	**kNN^{90}**	**CC^{70}+ k-NN^{90}**	**CC^{80}+ k-NN^{90}**	**CC^{90}+ k-NN^{90}**	**CC^{80}+ k-NN^{95}**
Recall	$88,9 \pm 0,6\,\%$	$89,1 \pm 0,6\,\%$	$89,2 \pm 0,6\,\%$	$89,0 \pm 0,6\,\%$	$\mathbf{89,3 \pm 0,6\,\%}$
Jaccard	$86,4 \pm 0,7\,\%$	$87,1 \pm 0,6\,\%$	$\mathbf{87,2 \pm 0,6\,\%}$	$86,6 \pm 0,7\,\%$	$87,0 \pm 0,6\,\%$
Confiabilidad por cantidad de cromosomas					
2	$94,5 \pm 1,8\,\%$	$\mathbf{97,0 \pm 1,4\,\%}$	$96,7 \pm 1,5\,\%$	$95,0 \pm 1,8\,\%$	$95,7 \pm 1,6\,\%$
3	$79,9 \pm 3,0\,\%$	$81,2 \pm 3,0\,\%$	$\mathbf{81,9 \pm 3,0\,\%}$	$80,7 \pm 3,0\,\%$	$81,3 \pm 3,0\,\%$
4	$73,7 \pm 3,4\,\%$	$73,9 \pm 3,4\,\%$	$74,8 \pm 3,4\,\%$	$73,7 \pm 3,4\,\%$	$\mathbf{75,1 \pm 3,3\,\%}$
5	$\mathbf{77,5 \pm 3,4\,\%}$	$70,7 \pm 3,6\,\%$	$72,7 \pm 3,6\,\%$	$76,4 \pm 3,5\,\%$	$77,3 \pm 3,4\,\%$
Total	$81,1 \pm 1,5\,\%$	$80,4 \pm 1,6\,\%$	$81,3 \pm 1,5\,\%$	$81,3 \pm 1,5\,\%$	$\mathbf{82,2 \pm 1,5\,\%}$

procesamiento ($88,8 \pm 0,6$ y $87,1 \pm 0,6$, respectivamente), por lo que la calidad de la segmentación prácticamente no se ve modificada. Mientras que en la confiabilidad no se detectó una tendencia esperable según el número de cromosomas al variar los parámetros. Además, la confiabilidad total también varía ligeramente, mostrando mejores resultados con CC^{80}+ k-NN^{95}, a una distancia menor de 1 % de los casos con otros umbrales. Se probó también con otros valores de umbrales para los métodos CC y k-NN, pero en todos los casos logrando un rendimiento inferior de confiabilidad.

Resulta interesante destacar que el tiempo de procesamiento por imagen del método CC^{90} +EPI del experimento de la Sección 4.2.3 es de 0,11 segundos, mientras que al añadir k-NN el mismo se incrementa a 3,8 segundos. En el primer caso se obtuvo una confiabilidad de $81,1 \pm 01,5$ mientras que en el segundo $82,2 \pm 1,5$, sin variar significativamente la calidad en la segmentación. Claramente, este incremento de confiabilidad de 1 % a expensas de un aumento de tiempo promedio del 3454,54 % no es provechoso. Por lo tanto, se concluye que es conveniente utilizar la combinación de métodos CC^{90}+EPI ya que se obtienen resultados similares a un costo significativamente menor de recursos.

4.2.5. Red convolucional profunda de corrección

En esta sección se analiza el desempeño del método explicado en la Sección 4.1.4, que consiste en utilizar una red Hu similar a la usada en la separación para el post-procesamiento. Primero se entrenó sólo la segunda red manteniendo la primera fija según lo entrenado para la separación, logrando un rendimiento inferior al de partida. Segundo, se probó entrenando la red completa (Red W) con más de 60 mil imágenes y los resultados obtenidos prácticamente no variaron. Luego, se determinó que el problema podía ser la cantidad de datos, por lo que se re-entrenó la red W con aproximadamente 135 mil imágenes. Adicionalmente, estos últimos resultados fueron procesados empleando el método CC^{90}+EPI por su buen desempeño en 4.2.3. La Tabla 4.5 muestra un resumen de los resultados obtenidos.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 4.5: Resultados de la red convolucional profunda de corrección.

Método	**Sólo 2da. U**	**Red W**	**Red W con más datos**	**Red W+ CC^{90}+ EPI**
Recall	$82,9 \pm 0,5\,\%$	$82,4 \pm 0,7\,\%$	$\mathbf{92,8 \pm 0,4\,\%}$	$\mathbf{92,8 \pm 0,5\,\%}$
Jaccard	$81,5 \pm 0,6\,\%$	$78,0 \pm 0,8\,\%$	$\mathbf{91,0 \pm 0,5\,\%}$	$90,2 \pm 0,6\,\%$
Confiabilidad por cantidad de cromosomas				
2	$98,8 \pm 1,0\,\%$	$99,2 \pm 1,1\,\%$	**100 %**	$95,3 \pm 1,7\,\%$
3	$69,9 \pm 4,2\,\%$	$45,6 \pm 6,2\,\%$	$10,9 \pm 3,1\,\%$	$\mathbf{84,1 \pm 2,8\,\%}$
4	$48,8 \pm 4,6\,\%$	$21,3 \pm 5,1\,\%$	$9,2 \pm 2,7\,\%$	$\mathbf{78,5 \pm 3,1\,\%}$
5	$38,2 \pm 4,4\,\%$	$12,3 \pm 3,6\,\%$	$9,5 \pm 2,7\,\%$	$\mathbf{80,2 \pm 3,2\,\%}$
Total	$63,1 \pm 2,2\,\%$	$42,6 \pm 3,0\,\%$	$13,7 \pm 1,8\,\%$	$\mathbf{84,5 \pm 1,4\,\%}$

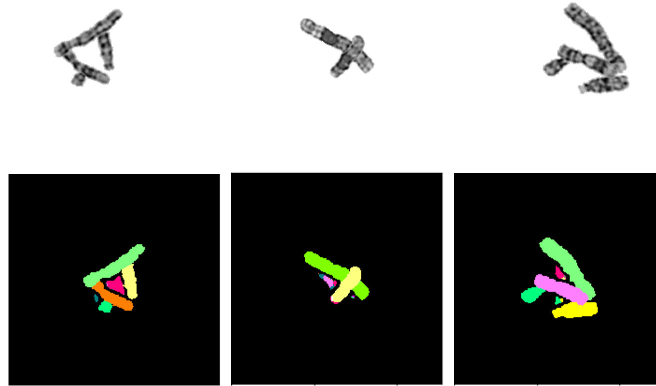


Figura 4.2: Predicciones de la red convolucional de corrección con artefactos en zonas correspondientes al fondo. En la primer fila los clusters de cromosomas y en la segunda el resultado que devuelve la red.

Se observa un aumento notable del recall y el coeficiente de Jaccard respecto a los métodos que no requerían re-entrenar una red convolucional. Sin embargo, llama la atención la baja confiabilidad de la Red W entrenada con más datos. De la inspección visual de las imágenes predichas por la red W se ve que se introducen artefactos en zonas correspondientes al fondo, como se muestra en la Figura 4.2. Esto provoca que se detecten clases adicionales en zonas que ni siquiera hay cromosomas presentes y, por lo tanto, la confiabilidad dé resultados muy bajos. Cabe destacar que la excepción es el caso de los 2 cromosomas en el que la confiabilidad es perfecta, pero esto es debido a que los artefactos recién mencionados provocan que muy pocos clusters sean detectados con dicho número de cromosomas (sólo el 8,6 %) y, en esos casos, es correcto.

Como se obtiene una gran mejora en la calidad de la segmentación con

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 4.6: Resultados de la red convolucional profunda de corrección aplicando la máscara de pre-procesamiento.

Método	**Sólo 2da. U**	**Red W**	**Red W con más datos**	**Red W+ CC^{90}+ EPI**
Recall	$82,9 \pm 0,5\,\%$	$82,4 \pm 0,7\,\%$	$\mathbf{92,8 \pm 0,4\,\%}$	$\mathbf{92,8 \pm 0,5\,\%}$
Jaccard	$81,5 \pm 0,6\,\%$	$80,4 \pm 0,5\,\%$	$\mathbf{91,5 \pm 0,5\,\%}$	$90,9 \pm 0,6\,\%$
Confiabilidad por cantidad de cromosomas				
2	$98,8 \pm 1,0\,\%$	$99,2 \pm 1,1\,\%$	$\mathbf{99,6 \pm 0,6\,\%}$	$95,0 \pm 1,7\,\%$
3	$69,9 \pm 4,2\,\%$	$45,6 \pm 6,2\,\%$	$75,2 \pm 3,8\,\%$	$\mathbf{85,7 \pm 2,7\,\%}$
4	$48,8 \pm 4,6\,\%$	$21,3 \pm 5,1\,\%$	$57,9 \pm 4,4\,\%$	$\mathbf{81,7 \pm 3,0\,\%}$
5	$38,2 \pm 4,5\,\%$	$12,3 \pm 3,6\,\%$	$44,7 \pm 4,5\,\%$	$\mathbf{86,5 \pm 2,8\,\%}$
Total	$63,1 \pm 2,2\,\%$	$42,6 \pm 3,0\,\%$	$69,0 \pm 2,1\,\%$	$\mathbf{87,1 \pm 1,3\,\%}$

la red W, se utilizará la máscara obtenida en el pre-procesamiento de la imagen para eliminar los artefactos que ésta genera. En este caso, como los solapamientos son sintéticos y el pre-procesamiento se realiza en la imagen global, se utilizará la máscara ya conocida del ground truth. En la Tabla 4.6 se muestran los resultados obtenidos con los mismos métodos de la Tabla 4.5 pero previamente quitando los artefactos según se explicó. Se observan mejores resultados con el método de la red W + CC^{90} + EPI. Aunque el mismo tenga una pequeña disminución en el coeficiente de Jaccard respecto al método de la red W sin aplicar CC^{90} + EPI, el recall se manteniene y aumenta la confiabilidad en casi un 20 %. En este sentido, un ligero detrimento en la segmentación de un pequeño grupo de imágenes (leve reducción en el coeficiente de Jaccard) tiene como beneficio el incremento en la confiabilidad del número de cromosomas predichos.

Para entender la causa de este comportamiento en la red convolucional, se procede a realizar una inspección visual de predicciones tomadas al azar utilizando el método de la red W + CC^{90}+ EPI. Analizando la Figura 4.3a, en la primer y segunda columna se observan casos en los que se asigna incorrectamente una clase, pero la separación de los cromosomas se realiza correctamente y por lo tanto cumple igualmente con el objetivo del proyecto. Esto provoca que se vean disminuidas las medidas de recall y coeficiente de Jaccard aunque la separación pueda realizarse correctamente. No es así el caso de la tercera columna, en la que la equivocación se da con una clase ya presente en la predicción y por lo tanto no sólo se afectan las métricas recién mencionadas, sino también la confiablidad y la correcta separación. Sin embargo, en la Figura 4.3b se observan imágenes que el método mencionado no fue capaz de resolver adecuadamente la segmentación. Como se ve en la primer columna, la red convolucional tiene deficiencias para separar correctamente un cromosoma que está parcialmente oculto en una zona central a él y que, por lo tanto, tiene dos componentes conexas. En estos casos, puede pasar que se asigne diferentes clases a cada componente. En

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 4.7: Resultados de la comparación de métodos.

Método	**Ninguno**	**CC^{90} + EPI**	**CC^{80}+ k-NN^{95}**	**Red W + CC^{90}+EPI**
Recall	$88,9 \pm 0,6\,\%$	$89,1 \pm 0,6\,\%$	$89,2 \pm 0,6\,\%$	$\mathbf{93,3 \pm 0,5\,\%}$
Jaccard	$87,2 \pm 0,6\,\%$	$86,4 \pm 0,7\,\%$	$86,7 \pm 0,7\,\%$	$\mathbf{91,5 \pm 0,5\,\%}$
Confiabilidad por cantidad de cromosomas				
2	**100 %**	$92,8 \pm 2,0\,\%$	$95,6 \pm 1,6\,\%$	$96,3 \pm 1,5\,\%$
3	$67,2 \pm 4,7\,\%$	$81,7 \pm 3,0\,\%$	$82,8 \pm 2,9\,\%$	$\mathbf{89,4 \pm 2,3\,\%}$
4	$43,8 \pm 4,8\,\%$	$74,6 \pm 3,4\,\%$	$76,6 \pm 3,2\,\%$	$\mathbf{85,5 \pm 2,7\,\%}$
5	$30,2 \pm 4,5\,\%$	$78,6 \pm 3,4\,\%$	$81,1 \pm 3,2\,\%$	$\mathbf{88,8 \pm 2,5\,\%}$
Total	$59,9 \pm 2,4\,\%$	$81,9 \pm 1,5\,\%$	$83,9 \pm 1,4\,\%$	$\mathbf{90,0 \pm 1,1\,\%}$

las predicciones de la segunda y tercer columna, la red convolucional no es capaz de distinguir correctamente los cromosomas. Esto puede ser debido a deficiencias en la generación de datos ya que los cromosomas están muy apretados en poco espacio o que simplemente la red convolucional no sea capaz de diferenciar clases de cromosomas similares.

Se concluye que con arquitecturas de redes convolucionales más complejas y optimizando la generación de datos se podrían mejorar aún más los resultados.

4.2.6. Comparación de los mejores métodos

El objetivo de esta sección es comparar los métodos con los que se obtuvieron mejores resultados en cada uno de los experimentos precedentes. Para ello, como ya se explicó, se generó un nuevo dataset de imágenes a fin de determinar el desempeño de cada uno de ellos con datos nunca vistos.

De forma similar a las secciones anteriores, se presentan las medidas de recall, Jaccard y confiabilidad para cada uno de ellos en la Tabla 4.7. Tanto en el recall como en Jaccard se ve una amplía mejoría del método Red W + CC^{90}+EPI, en ambos casos sacando una diferencia mayor de 4 % a los demás. Dicho método también es superior en cuanto a la confiabilidad, siendo ampliamente mayor que los demás en el total y para todos las cantidades de cromosomas. La única excepción es el caso de la confiabilidad para solapamientos de dos cromosomas, en la que resultó superior la red convolucional de separación sin ningún post-procesamiento con un 100 %. Sin embargo, el método Red W + CC^{90}+EPI obtuvo un 96,3 %, disminución que está justificada con la notable mejoría que se obtienen en los otros casos.

Se reafirma la conclusión obtenida en el experimento de la Sección 4.2.5: con la exploración de arquitecturas de redes convolucionales más complejas podrían mejorarse los resultados obtenidos.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 4.8: Resultados de la evaluación del mejor método sobre cromosomas reales.

Método	Separación	Post-procesamiento
Clusters correctos	**42**	40
Clusters aceptables	9	13
Clusters incorrectos	9	**7**
Individuales correctos	18	**20**
Individuales correctos	2	**0**

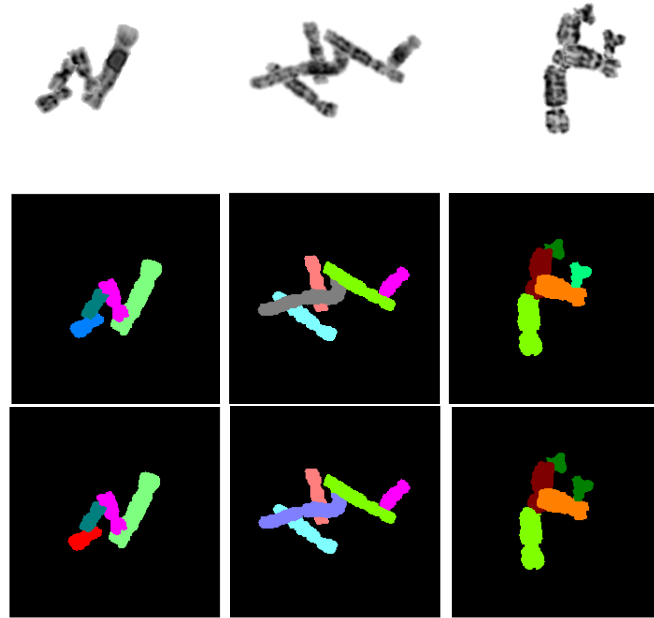
sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

4.2.7. Evaluación del mejor método sobre cromosomas reales

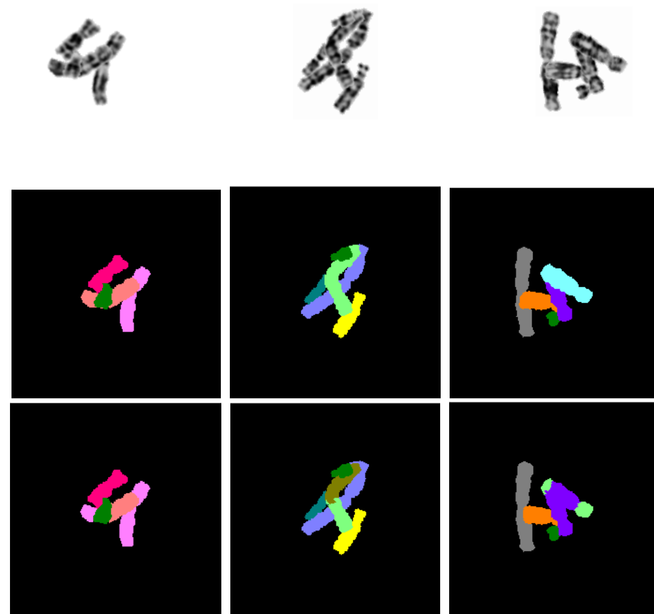
El objetivo de este experimento es evaluar el desempeño del método combinado Red W + CC^{90} + EPI en imágenes de cromosomas reales y analizar si se produce alguna mejora respecto a los resultados obtenidos en la Sección 3.4.5. Para ello, se aplica el mencionado método a los mismos 80 clusters de cromosomas reales que habían sido elegidos al azar en la sección mencionada, de los cuales 60 son clusters propiamente dichos y 20 son cromosomas individuales. Luego, se procede a realizar una inspección visual sobre las predicciones y se las clasifica según si la segmentación es incorrecta (Clusters incorrectos), si la segmentación tiene sólo pequeñas zonas incorrectas (Clusters aceptables) o si es correcta (Clusters correctos). No se tiene en cuenta si la clase asignada es correcta o no, sino si los cromosomas están bien separados. Además, los cromosomas individuales se clasifican simplemente en correctos e incorrectos.

En la Figura 4.4a se muestran casos en que el post-procesamiento mejora respecto a la separación y en la Figura 4.4b casos en los que empeora. También, en la Figura 4.5 se muestran ejemplos en los que el rendimiento es similar. Además, en la Figura 4.6 hay casos que no son posibles de resolver por tener dos cromosomas reconocidos como de una misma clase y que fueron contados para los resultados presentados.

En la Tabla 4.8 se presentan los resultados obtenidos en comparación con lo obtenido en la Sección 3.4.5. Se podría concluir que hay una leve mejoría respecto a la separación ya que, aunque disminuyó la cantidad de clusters correctos, aumentó la cantidad de individuales correctos y disminuyó el número de clusters incorrectos. Sin embargo, la diferencia no es significativa como lo es en el caso de los solapamientos sintéticos debido a que los datos sintéticos no son iguales a las imágenes de los cromosomas reales. Los resultados sobre los cromosomas reales podrían mejorar si se tuvieran imágenes microscópicas reales con su correcta segmentación para poder entrenar la red convolucional que ya se demostró que funciona.



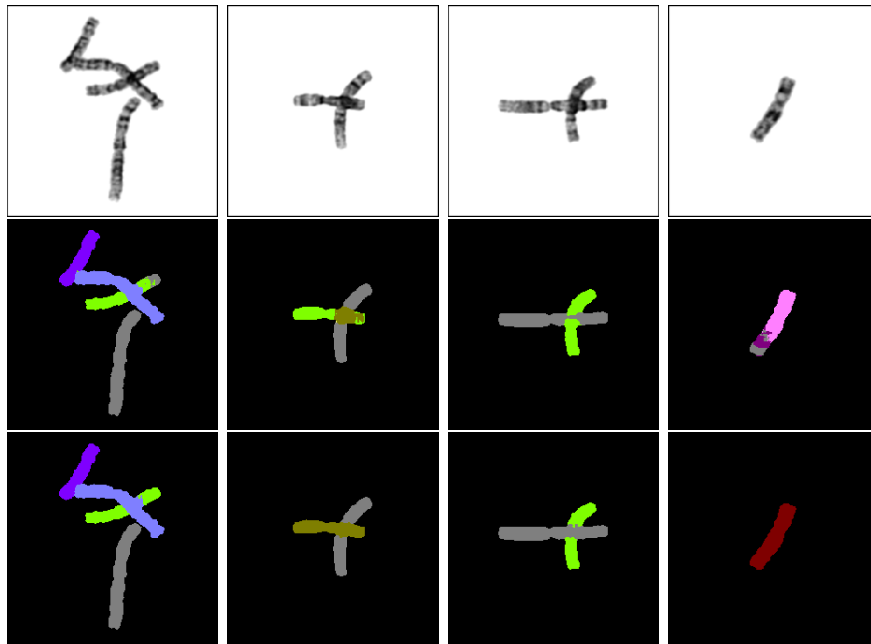
(a)



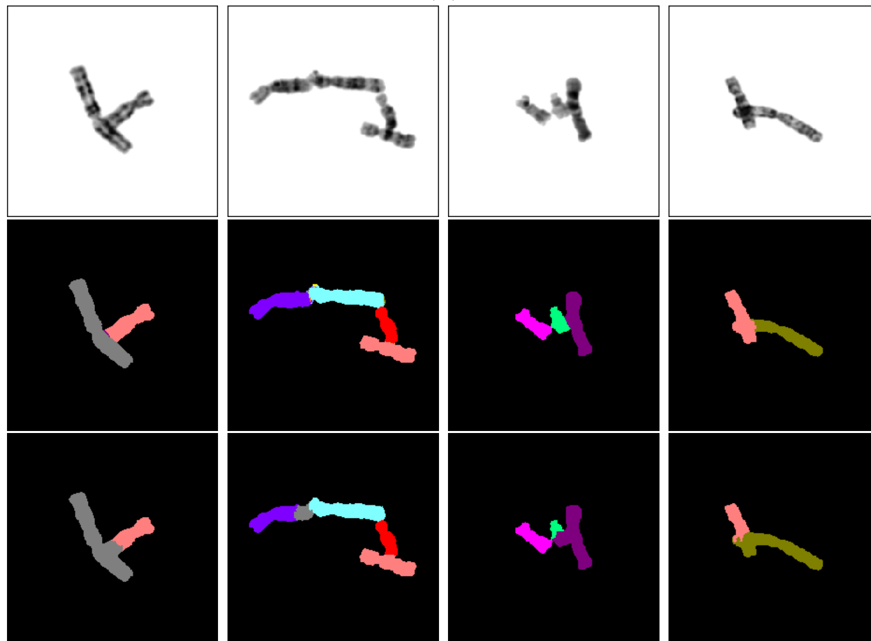
(b)

Figura 4.3: La primer fila son los clusters a segmentar, la segunda los ground truth de los mismos y la tercera las predicciones realizadas por el modelo luego del post-procesamiento. (a) Imágenes de experimento 5 mal clasificadas pero bien separadas. (b) Imágenes de experimento 5 separadas de forma incorrecta.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.



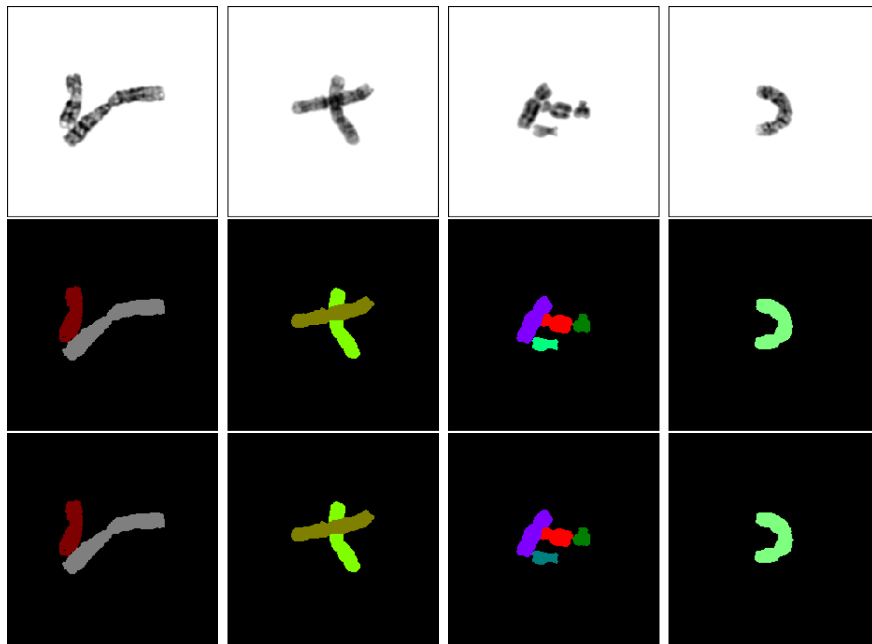
(a)



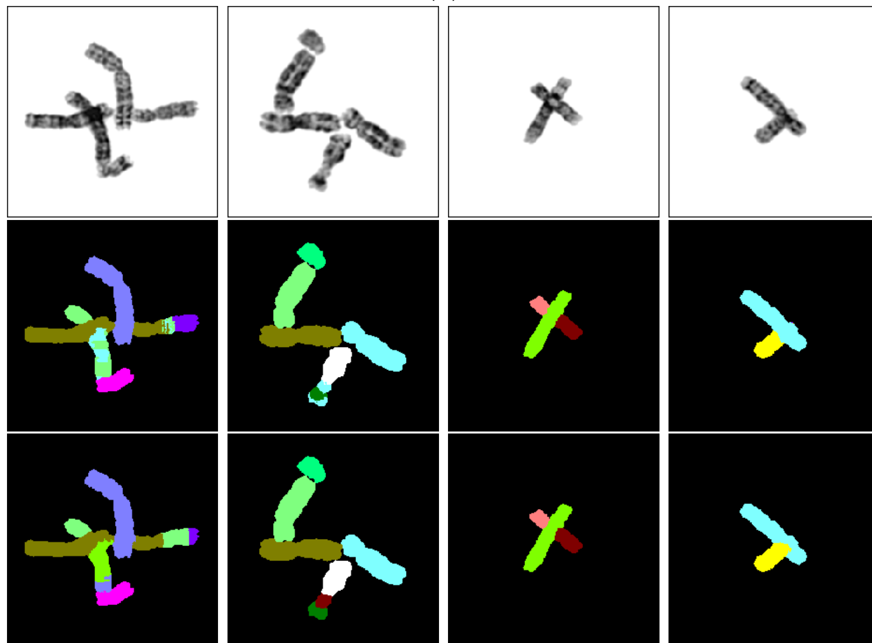
(b)

Figura 4.4: En la primer fila se ven los clusters a segmentar, en la segunda la predicción realizada por el modelo de separación del Capítulo 3 y en la tercera la predicción luego del post-procesamiento. (a) Predicciones de post-procesamiento con mejora respecto a lo obtenido en la separación. (b) Predicciones de post-procesamiento con menor precisión a lo obtenido en la separación.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.



(a)



(b)

Figura 4.5: En la primer fila se ven los clusters a segmentar, en la segunda las predicciones realizadas por el modelo de separación del Capítulo 3 y en la tercera las predicciones luego del post-procesamiento. (a)(b) Predicciones de post-procesamiento con rendimiento similar a lo obtenido en la separación.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

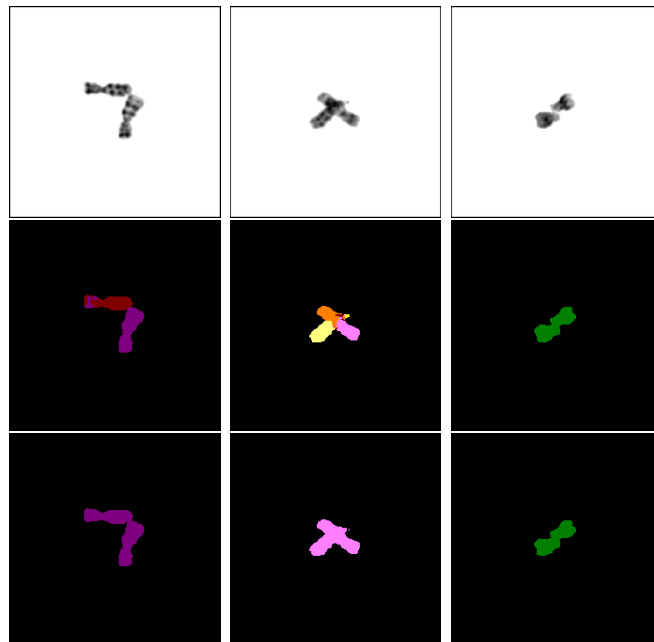


Figura 4.6: Predicciones imposibles de resolver para el procedimiento planteado. En la primer fila se ven los clusters a segmentar, en la segunda la predicción realizada por el modelo de separación del Capítulo 3 y en la tercera la predicción luego del post-procesamiento.

4.3. Conclusiones

Se abordó la corrección de las máscaras obtenidas en el capítulo correspondiente a la separación de solapamientos. Para la comparación del desempeño de los diferentes métodos se utilizó nuevamente el recall y el Jaccard, y además se incorporó una nueva medida denominada confiabilidad. La misma indica la probabilidad de que el número de cromosomas segmentados sea efectivamente el correcto.

Primero se probaron métodos que no involucran redes convolucionales tales como la corrección de pequeñas imperfecciones (EPI), la corrección por canales (CC) y k-NN. Se observó que éstos no modifican significativamente la calidad de la segmentación ya que prácticamente no hacen variar al recall y al Jaccard, pero provocan mejoras en la medida de confiabilidad. Se ve que los mejores resultados se obtienen al combinar los métodos CC^{90} y EPI, que lograron que la confiabilidad que se tenía desde la separación del 55 % supere el 80 %.

Luego, se utilizaron métodos que requirieron el entrenamiento de redes convolucionales. Se adicionó a la red convolucional usada en la separación otra instancia de la misma con el objetivo de que aprenda a corregir los errores de la primera, que en conjunto se llamó Red W. Se encontró que dicha red añade artefactos en regiones correspondientes al fondo, por lo que se eliminan mediante el uso de la máscara obtenida en el pre-procesamiento. Los mejores resultados en cuanto a recall y coeficiente de Jaccard se obtuvieron re-entrenando la Red W de principio a fin y aumentando la cantidad de datos de entrenamiento. Éstos subieron de 88,8 % y 87,1 % que se tenía de la separación a 92,8 % y 91,5 %. Sin embargo, la confiabilidad seguía siendo relativamente baja (no llegaba a 70 %), por lo que se le acopló al método la combinación de CC^{90} + EPI, logrando que la misma llegue a 87 %.

Por último, se generó un nuevo dataset con el que probar los mejores métodos de forma que los desempeños fueran comparables. Con el método Red W + CC^{90} + EPI se obtuvo un recall de 93 %, un Jaccard de 91 % y una confiabilidad de 90 %, superando ampliamente a los demás. Por la magnitud de las mejoras obtenidas con los dos tipos de métodos, se concluye que con arquitecturas de redes convolucionales más complejas se podrían mejorar aún más los resultados. Otro detalle que mejoraría los resultados sería la optimización de la generación de datos ya que, por ejemplo, se observa que la red convolucional tiene problemas para separar casos en que los cromosomas están muy apretados en poco espacio. Estos casos, si bien pueden darse en la práctica real, suelen ser descartados ya que a mayor cantidad de solapamientos, mayor es la información de los cromosomas que se pierde. Una alternativa superadora a lo anterior sería contar con la correcta segmentación de clusters de cromosomas reales.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Capítulo 5

Integración

En este capítulo se presenta una descripción detallada de la codificación realizada para la obtención de la herramienta de segmentación de cromosomas. El flujo general de trabajo, así como las principales librerías empleadas en cada módulo de la herramienta se encuentran sumarizadas en la Figura 5.1. Como puede observarse, la herramienta cuenta con un módulo de pre-procesamiento, otro de separación de solapamientos y un último de post-procesamiento, desarrollados en ese orden de forma independiente para luego conectarlos y establecer una interfaz de uso. Además, se observa que el módulo de separación requirió abordar dos proyectos complementarios: la generación de datos y el entrenamiento de la red convolucional.

El capítulo se organiza de la siguiente manera. Primero se describen los recursos utilizados en la Sección 5.1. Luego, hay una sección dedicada a cada bloque de la Figura 5.1. Finalmente, en la Sección 5.8 se presentan resultados obtenidos con el uso de la herramienta integrada.

5.1. Materiales

Para el desarrollo de los módulos, se utilizaron librerías de código abierto. Las librerías OpenCV [43] y scikit-image (skimage) [44] son usadas para dar soporte a los métodos de procesamiento de imágenes, la primera por el conocimiento previo que se tenía sobre ella y la segunda por poseer una cantidad mayor de algoritmos implementados. Ambas trabajan sobre NumPy [45], que es utilizada en computación científica para dar soporte a operaciones vectoriales y matriciales. Ésta también es usada directamente en el desarrollo de los módulos. Para la definición de las arquitecturas de las redes convolucionales y su posterior entrenamiento, se utiliza la librería PyTorch [46]. La misma provee soporte para realizar el mencionado entrenamiento e inferencias en una GPU usando la librería propietaria CUDA [47]. Además se utilizó la librería matplotlib [48] para crear los gráficos de colores que muestran las distintas clases que se detectan en un cluster de cromoso-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

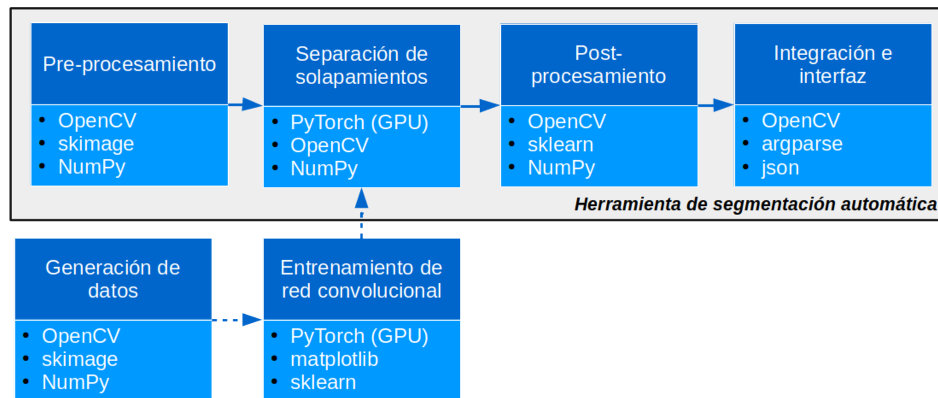


Figura 5.1: Flujo de trabajo.

mas, como se ve en los experimentos de las Secciones 3.4 y 4.2. También se usa scikit-learn (sklearn) [49] para el cálculo de las medidas de calidad de segmentación y el algoritmo k-NN. Por último, las librerías argparse [50] y json [51], incluidas por defecto en python, se utilizan para el pasaje de los parámetros en la ejecución de la herramienta y para formatearlos antes de almacenarlos (o después de leerlos), respectivamente.

Respecto a los recursos físicos empleados, todos los experimentos que no involucraban a una red convolucional se llevaron a cabo en la computadora propiedad del alumno. La misma también se utilizó en los casos que se necesita la red convolucional ya entrenada para separar clusters de cromosomas, pero para su entrenamiento se usaron otros recursos puesto que es una tarea más demandante. El principal de ellos es el Cluster Pirayú de la institución CIMEC [52], complementado en algunos casos por la herramienta CoLab de Google [53]. Esta última tiene la limitación de tiempo de ejecución de 12 horas, por lo que los experimentos en esta plataforma debieron realizarse por etapas.

En cuanto al corpus de imágenes, se tiene un dataset que consta de 612 imágenes, cada una correspondiente a distintas células en metafase con tinción en banda G. Del total, 536 imágenes contienen la cantidad normal de 46 cromosomas mientras que las restantes 76 poseen algún cromosoma extra o bien tienen el faltante de alguno de ellos. En total contiene 28148 cromosomas en 21089 componentes conexas. Además, el dataset trae consigo el cariograma de cada célula producido manualmente por citogenetistas [32, 36].

5.2. Pre-procesamiento

El objetivo del pre-procesamiento es la extracción de los clusters de cromosomas del fondo y la eliminación de objetos ruidosos. Como se muestra

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

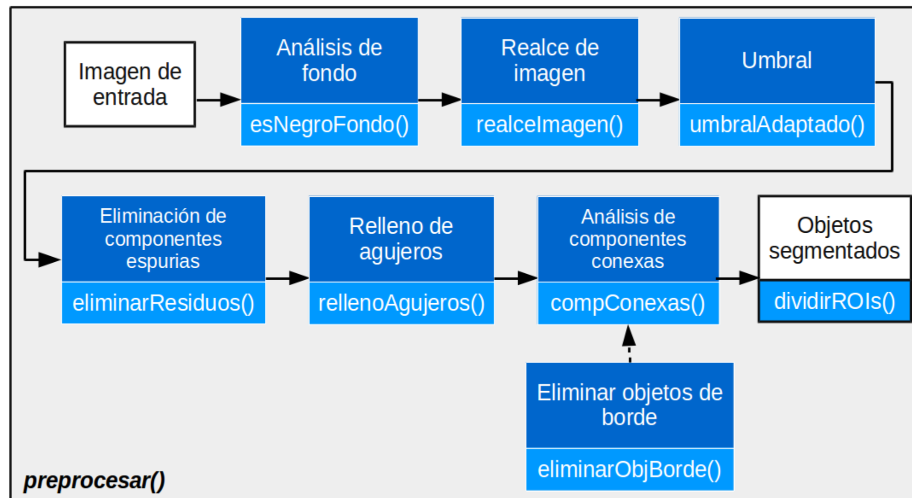


Figura 5.2: Funciones desarrolladas para cada bloque del módulo de pre-procesamiento.

en la Figura 5.2, se desarrolla una función para cada bloque del diagrama, excepto para el primero que corresponde a la imagen que se recibe como parámetro. Se aplican en la función integradora *preprocesar* que recibe la imagen microscópica a segmentar y los parámetros del módulo, para ir llamando las funciones en el orden descripto en la imagen. La misma devuelve una lista de imágenes que contienen sólo un cluster de cromosomas. A continuación se explica el funcionamiento del módulo, indicando entre corchetes los nombres de los parámetros de las funciones. Para más detalles se puede consultar el Apéndice A.

Se comienza con la imagen en forma de arreglo de NumPy de entero de 8 bits. Ésta es pasada a la función $esNegroFondo()$ junto con el tamaño del cuadrado que se utiliza para sacar de las muestras de píxeles de las esquinas [*tamCuadFondo*]. La misma, mediante la librería NumPy, calcula la intensidad promedio de los píxeles y devuelve Verdadero si el valor es más cercano a negro [0] o Falso si es más cercano a blanco [255], comparando si dicho promedio es menor a 128. Si el fondo es más cercano a negro, se calcula la inversa de la imagen simplemente haciendo píxel a píxel 255-i, siendo i la intensidad. Si *tamCuadFondo* es igual a 0, no se realiza ningún análisis sobre el fondo.

Luego, la imagen resultante [*img*] es pasada a la función $realceImagen()$, que le aplica la operación de ecualización adaptativa de histograma con contraste limitado (CLAHE) implementada en OpenCV. Los parámetros del realce son el tamaño de las ventanas [*TilesGridSize*] y el límite de contraste [*ClipLimit*]. Si alguno de ellos es 0, CLAHE no se aplica. A continuación, devuelve como resultado la imagen normalizada en el rango [0,255] mediante

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

la función *normalize()* de OpenCV. El mismo es pasado a la función *umbralAdaptado()* junto con una lista que indique los tamaños de ventana a utilizar [*tamCuadUmbral*]. Para cada tamaño indicado en la lista, se calcula el umbral de Otsu en ventanas no solapadas, se coloca el resultado en una matriz que luego se interpola mediante una función de OpenCV para llevar al tamaño de *img* y se aplica el umbral píxel a píxel. Cabe aclarar que el tamaño de ventana 0 corresponde al umbral global de Otsu. Por último, se devuelve la máscara acumulada aplicando operaciones lógicas OR mediante la librería OpenCV.

A partir de la máscara binaria recién obtenida [*img*] y un tamaño de elemento estructurante [*eeSize*], la función *eliminarResiduos()* aplica una erosión morfológica mediante una función provista por OpenCV. Luego, devuelve la reconstrucción morfológica por dilatación geodésica realizada a través de una función de skimage. La máscara binaria resultante [*img*] y un umbral de área que indica el tamaño máximo de agujero que se rellena [*menoresA*] son pasados a la función *rellenoAgujeros()*. La misma reconstruye morfológicamente por dilatación geodésica nuevamente con la función de skimage la inversa de la máscara desde los bordes tal como se explicó en la Sección 2.1.3, lo que se resta a *img* para obtener los agujeros. A continuación, detecta cada agujero mediante el algoritmo de detección de componentes conexas de OpenCV y, si su área, obtenida también mediante la librería OpenCV con la función *contourArea()*, es menor a *menoresA* rellena el agujero. Si *menoresA* fuera 0, rellena todos los agujeros sin ningún análisis.

A partir de la máscara binaria recién obtenida [*img*], en la función *compConexas()* primero se eliminan los objetos que están en contacto con el borde con un segmento menor al parámetro dado [*umbralSegm*] mediante una función desarrollada para tal fin [*eliminarObjBorde()*] que utiliza la reconstrucción morfológica de skimage. Seguidamente, para las componentes conexas restantes se verifica que su área sea menor a un umbral dado como parámetro [*umbralArea*]. De lo contrario, se calcula el área del convex hull de la componente mediante la ayuda de OpenCV para calcular la circularidad descripta en la Sección 2.1.4 y, si la misma es mayor al umbral dado [*umbralCH*], se elimina. Por último, se devuelve en una lista las componentes conexas resultantes.

Finalmente, la función *dividirROIs()* recibe la lista de componentes conexas recién descripta [*contours*] y, para cada una, aplica la operación lógica AND de OpenCV con la imagen pasada como parámetro [*data*]. En cada caso, la imagen se recorta con una función de OpenCV que determina el menor rectángulo que incluya a la componente conexa en cuestión, con el agregado de un margen de 2 píxeles. Las subimágenes resultantes se devuelven en una lista.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

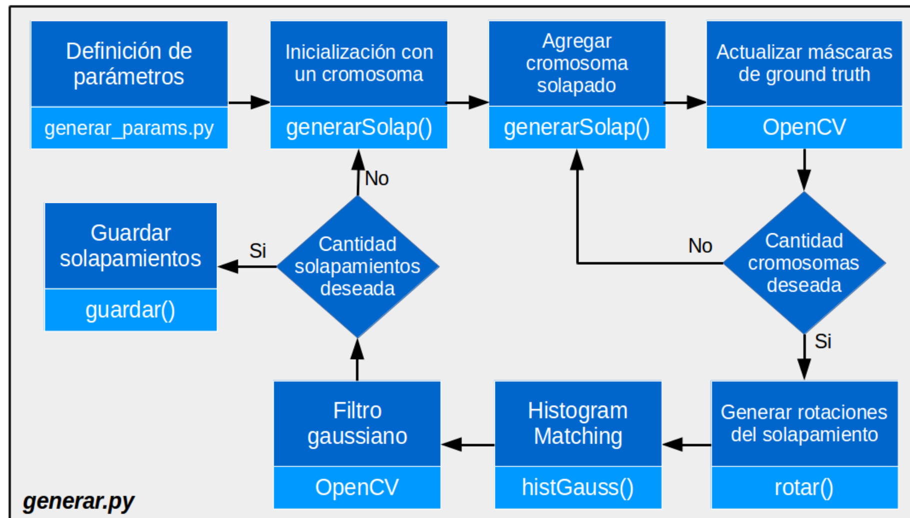


Figura 5.3: Funciones desarrolladas para cada bloque del proyecto de generación de datos.

5.3. Generación de datos

Como se explicó en el Capítulo 3, se abordó la generación de solapamientos sintéticos con su correspondiente ground truth para poder entrenar la red convolucional de separación. El mismo se puede ver como un proyecto complementario [1] ya que no forma parte de la herramienta de segmentación de cromosomas pero sustenta a la mencionada red convolucional. En la Figura 5.3 se muestran las funciones desarrolladas para la creación de las imágenes y su interrelación. Todas ellas se encuentran definidas en el archivo *generar_aux.py* para disminuir la longitud del archivo principal *generar.py* que las utiliza. A continuación se explica el funcionamiento del proyecto, indicando entre corchetes los nombres de los parámetros de las funciones. Para más detalles se puede consultar el Apéndice B.

Los cromosomas individuales en imágenes separadas se obtienen mediante la función *procesarCariogramas()* definida en el archivo [*separarKaryo.py*]. Ésta toma como parámetro un directorio y aplica el siguiente proceso a cada imagen que encuentre en él. Mediante OpenCV se lee la imagen y se utiliza un umbral de 254 para obtener una máscara que indique dónde estan los cromosomas. Luego, reutilizando las funciones explicadas en la Sección 5.2 se rellenan los agujeros de las máscaras, se detectan las componentes conexas para poder usarlas para extraer cada cromosoma y se almacenan en un directorio dado. La única diferencia con dicha sección es que las componentes detectadas se ordenan de arriba hacia abajo y de izquierda a derecha, de

[1] https://github.com/sfenoglio/GenData

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

forma que las imágenes extraídas se puedan guardar según su ubicación en el cariograma (que determina su clase). Por ello, se omiten cariogramas que contengan un número distinto de 46 cromosomas ya que provocaría un mal etiquetado de los mismos.

En el archivo *generar_params.py* se definen los parámetros con los que se crearán los solapamientos sintéticos, tales como el tamaño de la imagen resultante [*tamImagen*], la cantidad de veces que se rotará el solapamiento generado [*cant_angles*], la cantidad de solapamientos por cariograma [*train_cant y test_cant*] y la cantidad de cromosomas por solapamiento [*cant_crom_min*] y [*cant_crom_max*]. Otros de ellos son el directorio [*directorio*] en que se encuentran los cromosomas separados mediante el script *separarKaryo.py*, la cantidad de archivos de datos de salida de train [*cuantos_raw*], la cantidad de cariogramas que se utilizan para lograr cada uno de ellos [*intervalo*] y la cantidad total de cariogramas [*total_cario*] que se utiliza para calcular los sobrantes de train para generar los de test. Además se definen los porcentajes mínimo [*porcMinSolap*] y máximo [*porcMaxSolap*] de solapamiento y la mínima cantidad de end points nuevos [*minNewEP*] por cada cromosoma agregado. Por último, se determinan las cantidades de intentos fallidos máximos de agregar un cromosoma a un solapamiento antes de probar con otro [*maxIntentos*] y de seguir agregando cromosomas a un cluster antes de descartarlo para volver a comenzar [*maxIntentosAux*].

El proceso de generación de un solapamiento comienza con la creación de una imagen en blanco con un cromosoma e inicializando la lista que contendrá el ground truth de la correcta separación del solapamiento. Posteriormente, se entra en un bucle en el que se desplaza al azar el solapamiento que se tiene hasta el momento con *desplazarResize()* antes de agregar un cromosoma al azar al mismo con *generarSolap()*. Esta función adiciona el mencionado cromosoma al solapamiento existente en posiciones con rotaciones al azar generadas por la combinación de las funciones *desplazarResize()* y *rotar()*. La elección aleatoria de los cromosomas es según su clase sin reposición, de forma que en los solapamientos generados el número de aparición de cada una de dichas clases esté balanceadas. Luego, se actualizan las máscaras de ground truth teniendo en cuenta qué cromosomas son visibles mediante el uso de operaciones lógicas entre ellas de OpenCV. Se sale del mencionado bucle cuando se logra un solapamiento con la cantidad buscada de cromosomas o se re-comienza por algún error. Cabe aclarar que en *generarSolap()* se controla que no haya mas de dos cromosomas solapados en un mismo píxel, además de los porcentajes de solapamientos y la cantidad de end points nuevos calculados mediante *cantEP()*.

Una vez que se tiene un solapamiento, el mismo rota un ángulo al azar la cantidad de veces indicada en [*cant_angles*] mediante la función *rotar()*. La misma lo hace mediante una función de skimage, para luego actualizar las máscaras de ground truth correspondientes. Luego, a cada uno se le aplica el proceso de histogram matching explicado en la Sección 3.2.2 con un desvío

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

al azar entre [*minStd*] y [*maxStd*] con el uso de la librería NumPy. Seguido a esto, cada imagen se filtra con un kernel gaussiano de 3x3 con una media al azar entre [*minBlur*] y [*maxBlur*]. Por último, los solapamientos se centran en una imagen de tamaño [*tamImagen*] tomando como referencia el centro del mínimo cuadrado que contiene los cromosomas obtenido mediante una función de OpenCV.

El proceso se repite hasta obtener la cantidad de solapamientos deseada para las cantidades de cromosomas configuradas. Éstos se guardan mediante la función *guardar()*, que lo hace utilizando librería NumPy y el formato *.npz* que ella define. Dicho formato comprime las matrices sparse (que tienen pocos datos en comparación de los ceros que contiene), por lo que se almacena la inversa de los solapamientos de forma que el fondo sea negro y tenga una gran cantidad de ceros. Las mencionadas matrices están contenidas en un diccionario que bajo la clave 'data' guarda los solapamientos generados en una matriz de cuatro dimensiones y bajo 'maskLineal' el ground truth de cada uno en una matriz de tres dimensiones que guarda la información de a qué clase pertenece cada píxel con un número en el rango [0,23]. En ambos casos, la primer dimensión corresponde a la cantidad de datos y las dos últimas al alto y al ancho de las imágenes. En 'data' se agrega una segunda dimensión que indica la cantidad de canales de la imagen, que en este caso es 1 y sería innecesario, pero es información que se utiliza para el entrenamiento de la red convolucional de la Sección 5.4.

5.4. Entrenamiento de red convolucional

En esta sección se explican los distintos scripts y funciones desarrollados para el entrenamiento de la red convolucional que tiene como finalidad la separación de clusters de cromosomas solapados. El mismo se puede ver como un proyecto complementario [2] ya que no forma parte de la herramienta de segmentación de cromosomas pero es el que provee los parámetros entrenados de la red convolucional. En la Figura 5.4 se muestran los bloques desarrollados con los correspondientes archivos en que se hizo. Todos ellos son utilizados por el script *trainModel.py* que es el que integra los bloques para el entrenamiento. A continuación se explica el funcionamiento del proyecto, indicando entre corchetes los nombres de los parámetros del proceso. Para más detalles se puede consultar el Apéndice C.

Los parámetros del proyecto se definen en el archivo *parameters.py*, entre los que se encuentran la arquitectura utilizada [*nombreArq*], los directorios de las imágenes predichas [*PREDICT_PATH*], del modelo entrenado previo (si hubiera, sino se deja vacío) [*pathAnterior*], de almacenamiento del modelo [*MODEL_STORE_PATH*], de los datos de entrada [*DATA_PATH*] y de almacenamiento de predicciones sobre cromosomas reales [*REALES_-*

[2] https://github.com/sfenoglio/EntrenamientoRed

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

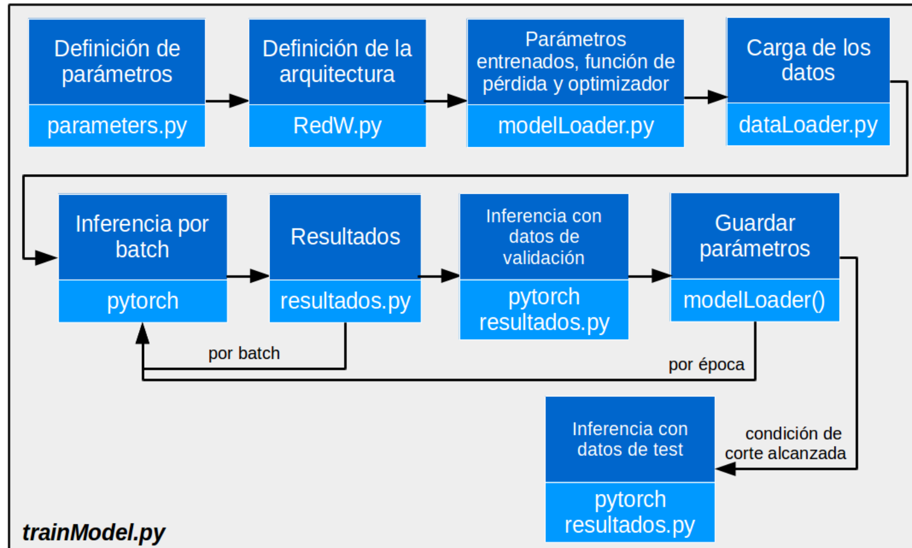


Figura 5.4: Scripts desarrollados para el entrenamiento de la red convolucional de separación.

PATH]. Para estos últimos, también se define la cantidad de módulos de entrenamiento a cargar [*cant_train*] y el porcentaje de datos que se utilizará para validación [*val_split*]. También se define la cantidad épocas que se entrena [*num_epochs*], el tamaño del batch [*batch_size*], la función de pérdida [*tipoLoss*], el optimizador a usar [*tipoOptim*], la tasa de aprendizaje [*learning_rate*] y el momentum [*momentum*] que se utiliza sólo cuando la función de pérdida es el gradiente descendiente. Por último, se define si se quiere ver las salidas online [*verGraficas*], cada cuantas épocas guardar las gráficas generadas [*cadaCuantas*] y cada cuantos batches guardar las predicciones de la red convolucional en el entrenamiento [*cadaCuantos*], en la validación [*cadaCuantosVal*] y en el test [*cadaCuantosTest*].

La arquitectura de la red convolucional utilizada se define en *RedW.py*, donde se especifica una clase con el mismo nombre que hereda de la clase *nn.Module* perteneciente a PyTorch. En la misma, se inicializa declarando las capas del modelo y luego se sobrecarga la función *forward()* en la que se especifica la forma en que dichas capas interactúan. Dicho modelo se envía a la GPU (si estuviera disponible) mediante la función *to()* de PyTorch. Luego, se utilizan las funciones definidas en *modelLoader.py* para obtener la función de pérdida *getLossFunc()*, obtener el optimizador *getOptimizer()* y para cargar parámetros ya entrenados para el modelo (si los hubiera) *loadModel()*. También se inicializa el logger en el que se guardará datos referidos a cada batch y la clase *graficas* definida en *resultados.py*.

Posteriormente, se cargan los datos de entrenamiento mediante la clase *genDataset()* definida en *dataLoader.py*. Ésta hereda de la clase *Dataset* de

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

PyTorch e incorpora al mismo los archivos de la forma 'train$k$.npz', donde k es un número desde 1 hasta *cant_train*. Además, se sobrecarga el método *__getitem__()* para transformar los datos con una media de 0,5 y un desvío de 0,5 y para que se pasen a formato *tensor* de PyTorch en tiempo de ejecución, debido a que almacenarlos como tal en memoria ocupa 8 veces más que hacerlo en formato entero de 8 bits. Luego, con la función *splitAndLoader()* también definida en *dataLoader.py*, se divide el dataset en una partición de entrenamiento y otra de validación. Ambas se transforman ahí mismo en un *DataLoader* definido por PyTorch, que permite iterar fácilmente por batch sobre cada partición. Un procedimiento similar se realiza con las imágenes de cromosomas reales (pero sin dividir en particiones), de forma que se pueda ver en cada época cómo funciona la red con clusteres reales.

A continuación, se define un bucle en el que se recorren los datos de entrenamiento de a batches, se envían a la GPU si estuviera disponible mediante *to()*, se infiere sobre las imágenes del mismo con la función *forward()* mencionada anteriormente, se calcula la pérdida con la función *backward()*, se actualizan los parámetros con la función *step()* del optimizador y se guardan los resultados obtenidos mediante la clase *graficas* definida en *resultados.py*. Esta última posee una función *cambiarPredicted()* que utiliza la librería matplotlib para mostrar una figura con dos imágenes: la predicha por la red a la izquierda y el ground truth a la derecha. Además, en caso de que se muestren los resultados online, la función se encarga de actualizar la figura que lo muestra. También se acumulan en listas las medidas de recall y coeficiente de Jaccard obtenidos mediante la función *medidas()* definida en *resultados.py*, que utiliza sklearn para calcularlos. Sumado a lo anterior, con la función *log()* definida en *resultados.py* se actualiza en OUT_PATH un archivo 'log.txt' con las medidas en cada batch. Una vez que se recorrieron todos los datos, se hace un procedimiento similar con los datos reservados para la validación y con las imágenes de cromosomas reales. Luego, se acumula en otras listas las medidas anteriores promediadas por época y mediante la función *graficar()* de *graficas()* se genera una figura que muestra gráficas del progreso de las medidas mencionadas en entrenamiento y validación durante las épocas.

Antes de proseguir con la siguiente iteración, se guardan los parámetros entrenados si es que la pérdida promedio en validación es menor a la mejor pérdida promedio obtenida hasta el momento. Esto se hace mediante la función *saveModel()* de *modelLoader.py* que los guarda en un diccionario mediante la clave *'state_dict'*. En el diccionario también se incluye el número de época con la clave *'epoch'* y el estado del optimizador con *'optim_dict'*. Luego, se repite el proceso desde el recorrido de los datos de entrenamiento hasta alcanzar la cantidad de épocas [*num_epochs*], que es la única condición de corte del entrenamiento. Una vez finalizado, se realiza un procedimiento similar con las imágenes de test.

Finalmente, se desarrolló un script *inferir.py* que es una versión sim-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

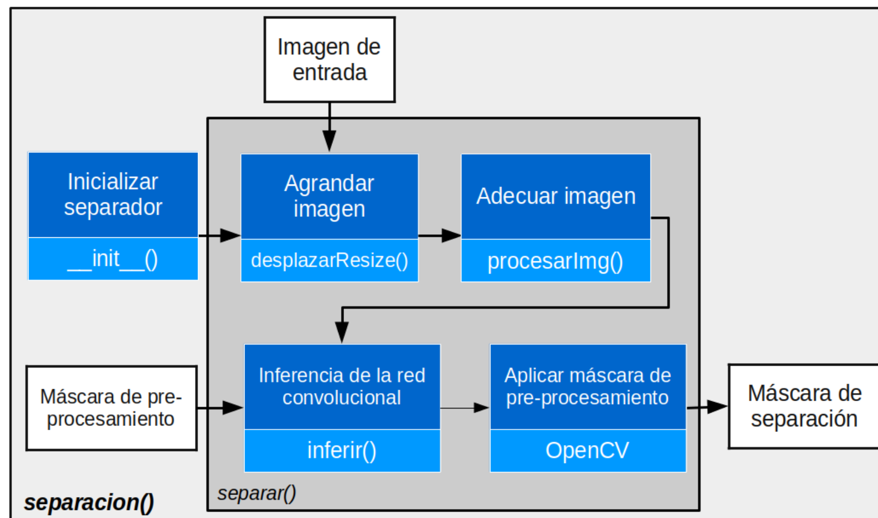


Figura 5.5: Clase desarrollada para la inferencia mediante una red convolucional.

plificada de *trainModel.py* ya que simplemente calcula la salida de la red convolucional y las medidas de calidad de segmentación, sin actualizar los parámetros de la red ni realizar las gráficas por época. Además, tiene la opción de que no se cuente con el ground truth y por ello no pueda calcular ninguna medida. Los parámetros del mismo se encuentran en *inferir_parameters.py* y son similares a *parameters.py*, exceptuando los que no son necesarios como la tasa de aprendizaje, momentum y el optimizador y agregando un booleano que indica si hay ground truth o no [*hayLabels*].

5.5. Separación

El módulo de separación incluye las porciones del proyecto de entrenamiento de la red convolucional de la Sección 5.4 necesarias para realizar una predicción a partir de un modelo entrenado. También abarca el proceso realizado a las imágenes de cromosomas reales explicado en la Sección 3.4.5 previo a la inferencia. Como se muestra en la Figura 5.5, se desarrolla una función para cada bloque del diagrama dentro de la clase *separacion()*. A diferencia de los otros módulos, se define una clase ya que éste conlleva una inicialización más lenta y luego se utiliza para separar todas los clusters de cromosomas que sea necesario mediante la función integradora *separar()*. A continuación se explica el funcionamiento del módulo, indicando entre corchetes los nombres de los parámetros de las funciones. Para más detalles se puede consultar el Apéndice A.

Se comienza definiendo la inicialización de la clase en la función __init__().

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

La misma consta de cargar los parámetros entrenados de un modelo dado con la función de PyTorch *load_state_dict()* y de mover la arquitectura a la GPU si así se le indicase y fuera posible. Luego, la función integradora *separar()* recibe las imágenes a separar [*imgs*], las procesa si así se indicase y después las envía a la red convolucional para determinar su máscara de separación, previo haberlas enviado a la GPU si allí estuviese el modelo. Antes del pasaje de la imagen a la red convolucional, la imagen que contiene el cluster debe ampliarse a un tamaño en el que la misma pueda submuestrearlo sin problemas, preferentemente que sea varias veces múltiplo de 2. Para ello, se aplica la función *desplazarResize()* que crea una imagen del tamaño deseado [*tamImagen*] con el color de fondo especificado [*fondo*] y coloca en el centro el cluster. Dicho fondo se elige según [*invertirImg*]: si éste fuera verdadero, significa que el fondo es blanco ya que el módulo fue preparado para trabajar con fondo negro. Si el tamaño del cluster fuera mayor a *tamImagen*, éste último se duplica.

La adecuación de la imagen realizada en la función *procesarImg()* primero calcula la inversa de la imagen si así se lo especificase en [*invertirImg*]. Esto es para permitir la opción de que se pueda pre-procesar la imagen con un color de fondo blanco y luego separarla con fondo negro. Luego, le aplica el método de histogram matching con una gaussiana con media 128 y un desvío dado [*std*] y al resultado lo convoluciona con un filtro gaussiano de 3x3 con un desvío pasado como parámetro [*blur*]. El objetivo de ello es lograr que los cromosomas reales se parezcan más a los solapamientos generados sintéticamente. Si alguno de estos dos parámetros fuera 0, la correspondiente operación no se realiza.

A continuación, la inferencia realizada por la función *inferir()* transforma la imagen de formato arreglo de NumPy a tensor de PyTorch, lo lleva del rango $[0, 255]$ al rango $[0, 1]$, lo envía a la GPU si fuera necesario para pasarlo por la red convolucional y al resultado lo tranforma nuevamente en arreglo de NumPy. Si se utilizara el modelo que viene por defecto *RedW*, éste incorpora en la función *forward()* la normalización de los datos a la media 0,5 y desvío 0,5 que en la Sección 5.4 se realiza en la carga de datos. Además, si se pasan como parámetros las máscaras obtenidas en el pre-procesamiento [*imgs_mask*], éstas se aplican a las imágenes posteriormente al pasaje por la mencionada red. Esto es para eliminar los artefactos que añadía el modelo utilizado en el post-procesamiento de la Sección 4.2.5. Si la lista *imgs_mask* tiene una longitud diferente a *imgs*, esto no se aplica.

5.6. Post-procesamiento

El objetivo del post-procesamiento es mejorar la máscara obtenida en la separación. Los métodos que involucran el entrenamiento de una red convolucional fueron probados mediante el proyecto de la Sección 5.4 y luego

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

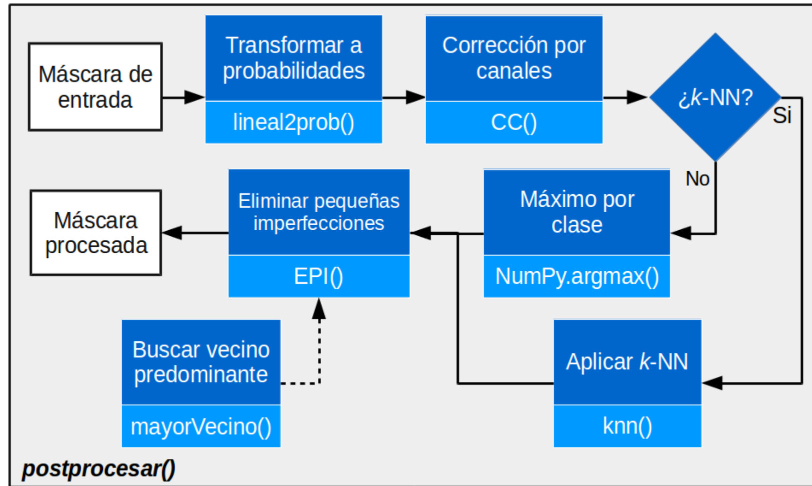


Figura 5.6: Funciones desarrolladas para cada bloque del módulo de post-procesamiento.

simplemente se actualiza la arquitectura del módulo de separación de la Sección 5.5. Por lo tanto, en esta sección se explican los métodos que no requieren el entrenamiento de una red. Como se muestra en la Figura 5.2, se desarrolla una función para cada bloque del diagrama, excepto para la elección del máximo por clase que se utiliza una perteneciente a NumPy. Se aplican en la función integradora *postprocesar()* que recibe la máscara de separación y los parámetros del módulo, para ir llamando las funciones en el orden descripto en la imagen. La misma devuelve la máscara corregida. A continuación se explica el funcionamiento del módulo, indicando entre corchetes los nombres de los parámetros de las funciones. Para más detalles se puede consultar el Apéndice A.

La salida de la red convolucional es lineal, por lo que se transforma a probabilidades para acotarlas entre 0 y 1 mediante la función *lineal2prob()*. Esto se hace mediante la función *Softmax2d()* de PyTorch y luego se transforma a formato arreglo de NumPy. Luego, se aplica la función *CC()* que desarrolla el método explicado en la Sección 4.1.2. La misma recibe las probabilidades recién mencionadas [*dataNP*] y un umbral de probabilidad [*umbralCC*] con el que determina las clases confiables fijándose que tengan al menos un píxel mayor a él. Si dicho umbral fuera 0, el método no se aplica. Posteriormente, hace 0 los canales de las clases no confiables y corrige las probabilidades para que sumen 1. Todo ello lo realiza mediante funciones de la librería NumPy.

El método de k-NN desarrollado en la función *knn()* también recibe un umbral de probabilidad [*umbralKNN*] para determinar los píxeles confiables mediante funciones de NumPy. Estos se utilizan para entrenar la clase *neighbors* de sklearn, para luego iterar sobre los píxeles no confiables y cla-

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

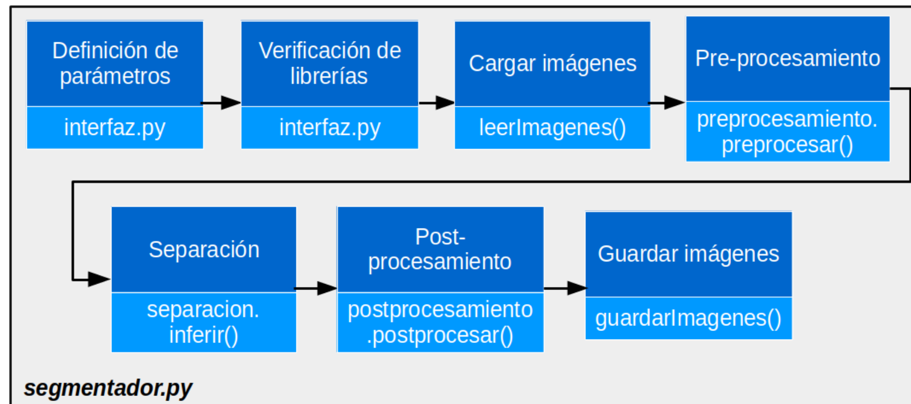


Figura 5.7: Funciones desarrolladas para la integración de los módulos.

sificarlos. Este proceso le asigna una clase a cada píxel, por lo que en caso de no aplicarse (cuando el umbral es 0) se asigna una clase a cada píxel eligiendo la que tenga mayor probabilidad mediante la función *argmax()* de NumPy.

Por último se aplica el método de eliminación de pequeñas imperfecciones explicado en la Sección 4.1.1 desarrollado en la función *EPI()*. La función comienza buscando las componentes conexas para cada clase presente en la imagen mediante el algoritmo *findContours()* de OpenCV. Se recorre cada una de ellas y los píxeles de las que sean menor al umbral de tamaño [*umbralEPI*] se etiquetan con un número de clase 24. Si dicho umbral fuera 0, el método no se aplica. A continuación, se buscan las componentes conexas de la clase 24 con el mismo algoritmo de OpenCV y a los píxeles de ellas se le asigna la clase mayoritaria en la vecindad inmediata. Dicha clase se determina con la función *mayorVecino()* que lo hace mediante operaciones morfológicas de OpenCV para obtener los vecinos inmediatos y funciones de NumPy para el recuento de aparición de clases de ellos.

5.7. Integración e interfaz

En este punto, se tienen desarrollados los módulos de pre-procesamiento, separación y post-procesamiento, por lo que el paso siguiente es conectarlos en un mismo proyecto [3]. Además, se busca abstraer la complejidad de la programación y uso de la herramienta mediante el desarrollo de una interfaz por línea de comandos. En la Figura 5.7 se muestra la lógica bajo la cual se integran los módulos en el script *segmentador.py*. A continuación se explica el funcionamiento de cada bloque y su interacción con los demás. Para más detalles se puede consultar el Apéndice A.

[3] https://github.com/sfenoglio/Segmentador

Se comienza definiendo los parámetros con los que se puede configurar la herramienta en el archivo *interfaz.py* mediante la librería argparse incluida en Python. Ésta permite que al ejecutar la herramienta se pueda determinar en el mismo comando distintas configuraciones como el directorio de la imagen a segmentar, directorio de salida, detalles de pre-procesamiento, etc. En la Tabla 5.1 se presentan los parámetros definidos indicando a qué módulo corresponden y su valor por defecto.

En el archivo *default_args.json* se almacenan los parámetros que se utiliza la herramienta por defecto y que pueden actualizarse mediante el parámetro *–saveDefaultArgs*. Éste, al finalizar la ejecución de la herramienta, toma los parámetros con que se lanzó y los guarda en el mencionado archivo. También en *interfaz.py* se verifica que los módulos y las correspondientes librerías que utilizan se encuentren disponibles en el entorno de ejecución. Además, este paso incluye la inicialización del módulo de separación, que consta del cargado de la arquitectura de la red convolucional, de la lectura de los parámetros entrenados de la misma y del envío a la GPU si fuera el caso.

Posteriormente se define la función *leerImagenes()*, que recibe el path de un archivo [*path*] y lo lee desde el disco. Este archivo puede ser una imagen con formato *'.jpg'*, *'.bmp'*, *'.png'*, *'.tiff'*, *'.jpeg'* o *'.tif'* que son los soportados por OpenCV. Adicionalmente, puede ser un archivo de texto *'.txt'* con el objetivo de que se pueda procesar más de una imagen con una configuración determinada, por lo que el archivo debe contener en cada línea el camino completo de una imagen con los formatos anteriormente mencionados. La función *leerImagenes()* también recibe el directorio de salida *[path_out]* y con él arma los directorios de salida para cada imagen, ya que para cada una se crea una carpeta que contendrá las imágenes de los cromosomas separados. Por defecto, el archivo de entrada es *example.txt* ubicado en la carpeta *images*, en la que se encuentran 12 imágenes microscópicas pertenecientes al dataset utilizado [32].

La conexión entre los módulos de pre-procesamiento, separación y post-procesamiento es simplemente pasar la salida de uno a la entrada de otro con el agregado de los parametros dados. El único detalle a tener en cuenta es que según los parámetros *–guardarPreproc* y *–noGuardarPreproc*, se guardan o no en la misma carpeta de salida los clusters obtenidos en el preprocesamiento. Éstos se almacenan de la forma *'n.png'*, donde n es el número de cluster extraído de la imagen. Para modificar la arquitectura de la red convolucional de este módulo, debe añadirse a la carpeta *src* un archivo que la defina utilizando PyTorch. Luego, con el parámetro *–nom_arch* se indica a la herramienta el nombre de dicho archivo y con *–path_model* se configura el directorio en que se encuentran los parámetros entrenados de la arquitectura. La clase que define la arquitectura debe tener el mismo nombre que el archivo.

Finalmente, se guardan las imágenes de los cromosomas individuales

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

obtenidas con nombres de la forma *'n_c.png'*, donde n es el número de cluster encontrado en la imagen y c es el número de clase del mismo. De esto se encarga la función *guardarImagenes()* que recibe las imágenes de los clusters de cromosomas [*imgs*], las máscaras de separación [*mascaras*] y el directorio de salida [*path_out*]. El valor por defecto de este último es la carpeta *out*. Previo al guardado, la función recorta la imagen ya que en la separación se agrandaron antes de que pase por la red convolucional. Para ello, utiliza la función *boundingRect()* de OpenCV para obtener el mínimo cuadrado que contiene al cromosoma dejando un margen de 5 píxeles por lado.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Tabla 5.1: Resumen de parámetros de cada módulo.

Parámetro	Nombre en interfaz	Valor por defecto
Pre-procesamiento		
Tamaño de cuadrado de fondo	–tamCuadFondo	10
Tamaño de ventana	–TilesGridSize	8
Límite de contraste	–ClipLimit	40
Vector de tamaños de ventana	–tamCuadUmbral	[0,100]
Tamaño de elemento estructurante	–eeSize	7x7
Tamaño máximo de agujero interno	–maxTamAgujero	10
Umbral de área	–umbralArea	4000
Umbral de proporción con área de convex hull	–umbralCH	0.8
Tamaño máximo de segmento en borde	–umbralSegm	30
Guardar imagen de cluster antes de la separación	–guardarPreproc –noGuardarPreproc	True
Separación		
Directorio de la definición de arquitectura	–path_arch	'RedW'
Directorio de los parámetros entrenados	–path_model	'./model/entrenado'
Realizar inferencia por GPU o por CPU	–device {'gpu','cpu'}	'gpu'
Tamaño al que se amplía el cluster	–tamImagenSeparar	256
Calcular la inversa de la imagen antes de pasarla por la red convolucional	–invertirImg –noInvertirImg	–invertirImg
Desvío para adecuar imagen con histogram matching	–adecStd	60
Desvío para adecuar imagen con filtro gaussiano	–adecBlur	0.7
No aplicar máscara de pre-procesamiento a la salida de la red convolucional	–aplicarMaskPreproc –noAplicarMaskPreproc	–aplicarMaskPreproc
Post-procesamiento		
Umbral para la corrección por canales	–umbralCC	0.9
Umbral para el algoritmo de k-NN	–umbralKNN	0
Umbral para el método EPI	–umbralEPI	100
Integración		
Directorio de archivo de entrada	–path_img	'./images/example.txt'
Directorio de salida	–path_out	'./out/'
Interfaz		
Ayuda sobre los parámetros	-h, –help	-
Listar parámetros por defecto	–verDefaultArgs	-
Guardar parámetros actuales para que sean por defecto	–saveDefaultArgs	-
Restaurar valores por defecto	–restoreDefaultArgs	-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

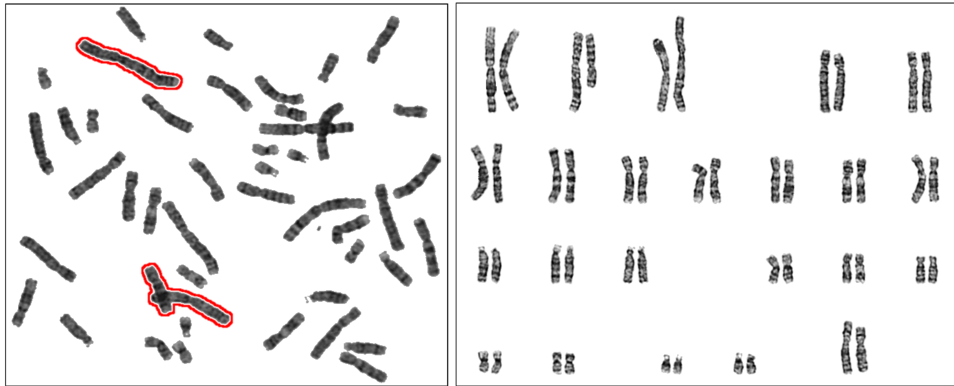


Figura 5.8: Imagen microscópica utilizada como entrada de la herramienta y su correspondiente cariograma.

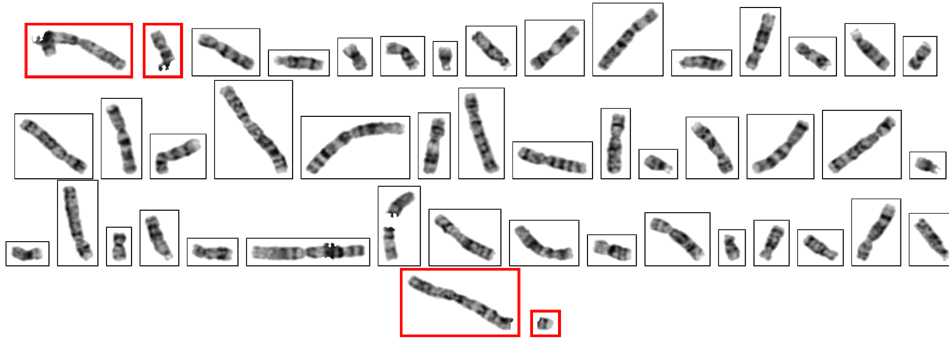


Figura 5.9: Imágenes de salida generadas por la herramienta.

5.8. Resultados y discusión

El objetivo de esta sección es mostrar el funcionamiento de la herramienta luego de la integración de los módulos y del desarrollo de la interfaz. Para ello, se toma una imagen al azar del dataset utilizado en [32] para comparar la segmentación realizada por la herramienta con el cariograma provisto por el mismo dataset. La imagen a segmentar con su correspondiente cariograma se muestran en la Figura 5.8. Los cromosomas segmentados por la herramienta se presentan en la Figura 5.9. Se observa que todos los cromosomas de la imagen de entrada son correctamente segmentados, a excepción de unas pequeñas imperfecciones en los casos que se marca en rojo. En las dos primeras imágenes, se ve que la separación de los dos cromosomas involucrados en el solapamiento no es perfecta ya que se recorta un trozo de uno de ellos. Luego, en las dos últimas, se ve que un único cromosoma es separado en dos partes. Este último caso podría corregirse modificando los parámetros de post-procesamiento.

Un detalle que se quiere destacar es la correcta resolución de la herra-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

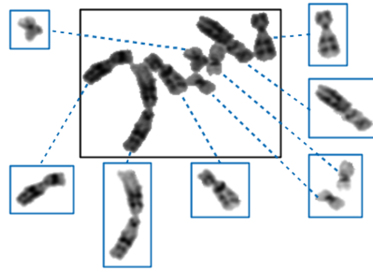


Figura 5.10: Cluster de 8 cromosomas y la separación realizada por la herramienta.

mienta al enfrentarse a un solapamiento con una gran cantidad de cromosomas. En la Figura 5.10 se muestra un cluster de 8 cromosomas correspondiente a otra imagen del dataset y la separación realizada por la herramienta. Se observa una correcta segmentación, a excepción de la imagen inferior derecha que corresponde a dos cromosomas de una misma clase, caso que ya se mencionó que la herramienta no puede resolver.

Los resultados sobre imágenes microscópicas reales son muy prometedores teniendo en cuenta la red convolucional fue entrenada con datos sintéticos generados a partir de los cromosomas extraídos de los cariogramas. Éstos difieren de los cromosomas crudos de las imágenes microscópicas ya que poseen un procesamiento desconocido. Por ende, es de esperar que las imágenes reales no sean perfectamente segmentadas. Si se conociera el procesamiento realizado para llevar de la imagen microscópica al cariograma o se tuvieran imágenes reales con su correspondiente segmentación para entrenar la red convolucional, podrían mejorarse aún más el desempeño de la herramienta.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Capítulo 6

Conclusiones

6.1. Conclusiones

En este trabajo se llevó a cabo el desarrollo de una herramienta para la segmentación automática de cromosomas empleando redes neuronales convolucionales. Ésta se construyó empleando un enfoque modular, desarrollando cada etapa de forma separada y luego integrando todos los módulos en un único flujo de trabajo.

El módulo de pre-procesamiento se abordó con una combinación de métodos de procesamiento de imágenes para extraer los objetos de la imagen y descartarlos en caso de que no fueran cromosomas. Se observó que los parámetros del módulo dependen fuertemente de la imagen de entrada, en aspectos tales como contraste, tamaño de los objetos y fondo. Sin embargo, se demostró el correcto funcionamiento para imágenes de cromosomas reales tomadas al azar con los valores recomendados del módulo.

El módulo de separación de solapamientos se basa en la aplicación de una red convolucional que, a partir de una imagen que contiene un único cluster de cromosomas, genera una máscara para cada cromosoma involucrado en el solapamiento. Debido a la falta de datos etiquetados para la segmentación de cromosomas solapados, primero se abordó la construcción un generador de datos sintéticos para producir las imágenes requeridas. El mismo permite generar solapamientos de un número variable de cromosomas, y generar rotaciones con variaciones de las características de los mismos. Estos datos fueron empleados para entrenar diferentes arquitecturas neuronales para segmentación de imágenes. Se consideraron modelos reportados en la literatura así como propuestas propias, principalmente basadas en la red U-Net. Las medidas de desempeño empleadas mostraron que la red HuV3BNN propuesta en este trabajo produce los mejores resultados. También se observó que la red segmenta correctamente las imágenes en las que hay sólo un cromosoma, a pesar de haber sido entrenada sólo con solapamientos de al menos dos cromosomas.

Para el módulo de post-procesamiento, primero se compararon métodos que no incluían entrenar redes convolucionales utilizando las mismas medidas que en la separación, con el agregado de una nueva llamada confiabilidad. El mejor desempeño se obtuvo con la combinación de los métodos de corrección por canales y de eliminación de pequeñas imperfecciones, que mejora notablemente la confiabilidad pero prácticamente no modifica las medidas de calidad de segmentación. Luego, se probó añadir una red convolucional similar a la usada en separación para que se encargue del post-procesamiento formando entre ambas lo que se llamó Red W. Ésta aumenta notablemente el desempeño respecto a la calidad de segmentación pero disminuye en cuanto a la confiabilidad. Por lo tanto, combinando la red W con los métodos de corrección por canales y de eliminación de pequeñas imperfecciones se logra utilizar las ventajas que ambos tipos de métodos proveen.

Como resultado la integración de éstos módulos se logró desarrollar una herramienta de código abierto, que alcanza muy buenos resultados con solapamientos sintéticos. Además, las segmentaciones producidas a partir de imágenes de solapamientos reales son prometedoras. Aunque no se cuenta con la información de la segmentación correcta, la inspección visual de los resultados indica que los cromosomas son segmentados de manera correcta en muchos casos. Adicionalmente, como no hay otras aproximaciones que usen redes convolucionales para esta tarea, el aporte de este trabajo puede servir como punto de partida para nuevos desarrollos. Cabe destacar que, aunque el objetivo de este trabajo es la segmentación de los cromosomas, la red convolucional utilizada está preparada para realizar también la clasificación de ellos.

6.2. Trabajo a futuro

Uno de los aspectos más importante en los que se podrían introducir mejoras es en la arquitectura de la red convolucional empleada. En este sentido, resultaría interesante explorar otros modelos como Masked Region Based Convolution Neural Network (Mask R-CNN), que utiliza dos redes convolucionales: una que se especializa en la detección de objetos y otra que usa esa información adicional a la imagen de entrada para clasificar los objetos detectados [54]. A continuación se presenta en cada sección las posibles vías de trabajo futuro para los distintos módulos.

6.2.1. Pre-procesamiento

En el módulo dedicado al pre-procesamiento, se podría trabajar sobre las imágenes microscópicas originales con el objetivo de que se parezcan más a las del cariograma ya que estas últimas son las utilizadas para el entrenamiento. Esto no es una tarea sencilla ya que el procesamiento realizado para llegar de unas a otras es desconocido y, según experimentos preliminares

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

realizados en este trabajo, no es lineal. Una posibilidad es el desarrollo de un modelo de redes adversarias [55], de forma que una de ellas aprenda a transformar las imágenes crudas en imágenes similares a las del cariograma. Así, estas imágenes transformadas deberían poder segmentarse con la red presentada en este trabajo sin inconvenientes.

6.2.2. Separación de solapamientos

En lo referente a la separación de solapamientos, un enfoque que podría explorarse es el aumento de información de entrada de la red convolucional como canales de la imagen. Ésta podría ser la misma imagen procesada con distintos valores de desvío para histogram matching o distintos desvíos de filtro gaussiano. También podría ser información que utilizan los métodos geométricos tales como el esqueleto morfólogico, el contorno, puntos angulosos del contorno, triangulación Delaunay del contorno o pale paths.

Un problema que quedó pendiente es la separación del solapamiento de de dos cromosomas del mismo tipo. Una opción sería utilizar la arquitectura Mask R-CNN mencionada anteriormente ya que la misma realiza detección de instancias de una misma clase. Alternativamente, podrían utilizarse métodos geométricos en un problema simplificado ya que teóricamente sólo podría haber dos de ellos normalmente o tres en casos excepcionales. Además, sabiendo el tipo de los dos o tres cromosomas podrían plantearse parámetros de acuerdo las características de ellos tales como tamaño o bandeo.

Nuevamente se menciona que un detalle que mejoraría los resultados sería la optimización de la generación de datos, por lo que sería una vía posible a continuar trabajando. Podrían generarse solapamientos sin variaciones para que luego en el entrenamiento de la red convolucional se modifiquen características del mismo tales como rotaciones, histogram matching, filtros convolucionales, ampliación o reducción del solapamiento. Esto se conoce como data augmentation online y el objetivo es que los datos varíen continuamente, de forma que la red pueda aprender características generales de las imágenes. La alternativa obvia a esto sería obtener una cantidad considerable de imágenes microscópicas con su correspondiente segmentación a fin de aplicar data augmentation con ellos.

6.2.3. Post-procesamiento

En la etapa de post-procesamiento, también existe margen para introducir mejoras. Por ejemplo, sería interesante evaluar el uso combinado de métodos geométricos con el método de corrección de pequeñas imperfecciones (EPI). Además del umbral del tamaño que utiliza, EPI podría incorporar métodos geométricos para el análisis de las componentes conexas tales como esqueletos, convex hull o contornos. De esta forma, podría subirse el umbral sin temor a eliminar cromosomas completos ya que los métodos mencionados

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

deberían detectar si la componente conexa es efectivamente un cromosoma o no.

Respecto al método de corrección por canales, podría pensarse en utilizar varios umbrales de forma tal que, según un análisis previo de los resultados para cada uno, se pueda determinar mediante algún algoritmo el número de cromosomas presentes en el solapamiento. Esto haría que el método no sea tan determinístico y pueda adaptarse a predicciones con muchos píxeles de baja probabilidad. Otra opción sería utilizar otro método para determinar el número n de cromosomas (otra red convolucional por ejemplo), para luego quedarse con las n clases de mayor probabilidad y hacer cero las probabilidades de los demás canales.

6.2.4. Integración e interfaz

Un punto a seguir mejorando es la compatibilidad de la herramienta con más plataformas. Actualmente se desarrolló con Python en Linux, con el objetivo de que el código sea abierto para su mejora continua y distribuida. Sin embargo, podría trabajarse en un paquete que contenga a la herramienta y que sea multiplataforma.

Por último, un aspecto a trabajar en la herramienta es el desarrollo de una interfaz gráfica que permita de forma más simple la configuración de los parámetros y la ejecución de la herramienta. Además, podría mostrar las imágenes de salida por pantalla de forma que puedan marcarse los cromosomas mal segmentados y a éstos se le aplique algún tratamiento alternativo. Esto no sólo ayudaría a evitar guardar resultados erróneos, sino que también permitiría saber cuáles fueron los cromosomas correctamente procesados y así almacenar su máscara de segmentación para generar así un dataset con cromosomas reales.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Bibliografía

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[2] N. Education, "Scitable." https://www.nature.com/scitable/definition/cytogenetics-72. (Accedido el 16-03-2019).

[3] A. P. Britto and G. Ravindran, "A review of cytogenetics and its automation," *J Med Sci*, vol. 7, pp. 1–18, 2007.

[4] W. K. Kwon, J. Y. Lee, Y. C. Mun, C. M. Seong, W. S. Chung, and J. Huh, "Clinical utility of fish analysis in addition to g-banded karyotype in hematologic malignancies and proposal of a practical approach," *The Korean journal of hematology*, vol. 45, no. 3, pp. 171–176, 2010.

[5] T. Arora and R. Dhir, "A novel approach for segmentation of human metaphase chromosome images using region based active contours," *International Arab Journal of Information Technology*, 2016.

[6] S. Minaee, M. Fotouhi, and B. H. Khalaj, "A geometric approach to fully automatic chromosome segmentation," in *Signal Processing in Medicine and Biology Symposium (SPMB), 2014 IEEE*, pp. 1–6, IEEE, 2014.

[7] G. C. Charters and J. Graham, "Trainable grey-level models for disentangling overlapping chromosomes," *Pattern Recognition*, vol. 32, no. 8, pp. 1335–1349, 1999.

[8] R. L. Hu, J. Karnowski, R. Fadely, and J.-P. Pommier, "Image segmentation to distinguish between overlapping human chromosomes," *arXiv preprint arXiv:1712.07639*, 2017.

[9] A. S. Imaging, "Genasis karyotyping." http://www.spectral-imaging.com/applications/cytogenetics/karyotyping. (Accedido el 17-05-2018).

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

[10] MetaSystems, “Rapidscore: Automation of fish spot counting.” https://metasystems-international.com/en/products/solutions/signal-analysis/. (Accedido el 17-05-2018).

[11] L. BioSystems, “Cytovision: Automated cytogenetics platform.” https://www.leicabiosystems.com/clinical-microscopy-surgery-radiology/cytogenetics/products/cytovision/. (Accedido el 17-05-2018).

[12] I. Belevich, M. Joensuu, D. Kumar, H. Vihinen, and E. Jokitalo, “Microscopy image browser: a platform for segmentation and analysis of multidimensional datasets,” *PLoS biology*, vol. 14, no. 1, p. e1002340, 2016.

[13] G. Mirzaghaderi and K. Marzangi, “Ideokar: an ideogram constructing and karyotype analyzing software,” *Caryologia*, vol. 68, no. 1, pp. 31–35, 2015.

[14] C. Vizzarri, C. E. Martínez, and M. Gerard, “Herramienta de código abierto para la construcción automática de cariogramas,” in *IX Congreso Argentino de Informática y Salud (CAIS)-JAIIO 47 (CABA, 2018)*, 2018.

[15] P. Wayalun, P. Chomphuwiset, N. Laopracha, and P. Wanchanthuek, “Images enhancement of g-band chromosome using histogram equalization, otsu thresholding, morphological dilation and flood fill techniques,” in *Computing and Networking Technology (ICCNT), 2012 8th International Conference on*, pp. 163–168, IEEE, 2012.

[16] M. V. Munot, J. Mukherjee, and M. Joshi, “A novel approach for efficient extrication of overlapping chromosomes in automated karyotyping,” *Medical & biological engineering & computing*, vol. 51, no. 12, pp. 1325–1338, 2013.

[17] S. Saiyod and P. Wayalun, “A hybrid technique for overlapped chromosome segmentation of g-band mataspread images automatic,” in *Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on*, pp. 400–404, IEEE, 2014.

[18] N. Madian, K. Jayanthi, and S. Suresh, “Contour based segmentation of chromosomes in g-band metaphase images,” in *Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on*, pp. 943–947, IEEE, 2015.

[19] E. Poletti, F. Zappelli, A. Ruggeri, and E. Grisan, “A review of thresholding strategies applied to human chromosome segmentation,” *Com-*

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

puter methods and programs in biomedicine, vol. 108, no. 2, pp. 679–688, 2012.

[20] E. Grisan, E. Poletti, and A. Ruggeri, "Automatic segmentation and disentangling of chromosomes in q-band prometaphase images," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 4, pp. 575–581, 2009.

[21] P. Moallem, A. Karimizadeh, and M. Yazdchi, "Using shape information and dark paths for automatic recognition of touching and overlapping chromosomes in g-band images," *International Journal of Image, Graphics and Signal Processing*, vol. 5, no. 5, p. 22, 2013.

[22] M. Munot, M. Joshi, and N. Sharma, "Automated karyotyping of metaphase cells with touching chromosomes," *Int J Comput Appl*, vol. 29, no. 12, pp. 14–20, 2011.

[23] L. Ji, "Intelligent splitting in the chromosome domain," *Pattern Recognition*, vol. 22, no. 5, pp. 519–532, 1989.

[24] L. Ji, "Fully automatic chromosome segmentation," *Cytometry: The Journal of the International Society for Analytical Cytology*, vol. 17, no. 3, pp. 196–208, 1994.

[25] D. Somasundaram, S. Palaniswami, R. Vijayabhasker, and V. Venkatesakumar, "G-band chromosome segmentation, overlapped chromosome separation and visible band calculation," *International Journal of Human Genetics*, vol. 14, no. 2, pp. 73–81, 2014.

[26] W. Srisang, K. Jaroensutasinee, and M. Jaroensutasinee, "Segmentation of overlapping chromosome images using computational geometry," *Walailak Journal of Science and Technology (WJST)*, vol. 3, no. 2, pp. 181–194, 2011.

[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[28] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing*, vol. 39, no. 3, pp. 355–368, 1987.

[29] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

[30] J. Serra and P. Soille, *Mathematical morphology and its applications to image processing*, vol. 2. Springer Science & Business Media, 2012.

[31] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[32] G. Ritter and L. Gao, "Automatic segmentation of metaphase cells based on global context and variant analysis," *Pattern Recognition*, vol. 41, no. 1, pp. 38–55, 2008.

[33] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3309–3318, IEEE, 2017.

[34] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2881–2890, 2017.

[35] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[36] U. of Passau, "Passau chromosome image data, pki-3." http://www.fim.uni-passau.de/en/faculty/former-professors/mathematical-stochastics/chromosome-image-data/. (Accedido el 07-08-2018).

[37] A. K. Dilip, "Visual segmentation of chromosomal preparations." https://github.com/abhaikollara/Chromosome-Segmentation/blob/master/README.md. (Accedido el 29-10-2018).

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[39] Neurohive, "Vgg16 – convolutional network for classification and detection." https://neurohive.io/en/popular-networks/vgg16/. (Accedido el 09-04-2019).

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[42] M. R. Spiegel, *Estadística.* McGraw-Hill Interamericana,, 1991.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

[43] OpenCV, "Open source computer vision library." https://opencv.org/. (Accedido el 07-08-2018).

[44] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[45] SciPy, "Numpy." http://www.numpy.org/. (Accedido el 07-08-2018).

[46] OpenSource, "Pytorch." https://pytorch.org/. (Accedido el 31-10-2018).

[47] Nvidia, "Cuda." https://www.nvidia.es/object/cuda-parallel-computing-es.html. (Accedido el 04-03-2019).

[48] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[49] SciPy, "scikit-learn." https://scikit-learn.org/stable/. (Accedido el 03-03-2019).

[50] P. S. Foundation, "arparse." https://docs.python.org/3/library/ argparse.html. (Accedido el 04-03-2019).

[51] P. S. Foundation, "json." https://docs.python.org/3/library/json.html. (Accedido el 29-07-2019).

[52] CIMEC, "Cluster pirayú." https://cimec.org.ar/c3/pirayu/. (Accedido el 03-03-2019).

[53] Google, "Colab." https://colab.research.google.com/. (Accedido el 03-03-2019).

[54] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[55] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Apéndice A

Documentación de la herramienta

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete interfaz

Archivo que incluye las funciones y definiciones necesarias para definir la interfaz de usuario.

Funciones

def **guardarDefaultArgs** ()
  Función que guarda los parámetros utilizados en la ejecución como parámetros por defecto.
def **leerImagenes** (path, path_out)
  Función que se encarga de leer del disco las imágenes de entrada a la herramienta.
def **guardarImagenes** (imgs, mascaras, path_out)
  Función que se encarga de almacenar las imágenes de salida que genera la herramienta.

Descripción detallada

Paquete que incluye las funciones y definiciones necesarias para definir la interfaz de usuario.

Los parámetros que acepta la interfaz se definen mediante la librería argparse y se utiliza JSON para almacenar los parámetros por defecto. Luego, se definen funciones que se encargan del cargado y almacenamiento de las imágenes de entrada y salida.

Documentación de las funciones

def interfaz.guardarDefaultArgs ()

Función que guarda los parámetros utilizados en la ejecución como parámetros por defecto.

Éstos se almacenan con formato JSON como 'default_args.json' ubicado en la carpeta 'config'.

def interfaz.guardarImagenes (*imgs*, *mascaras*, *path_out*)

Función que se encarga de almacenar las imágenes de salida que genera la herramienta.

Para cada imagen, se busca en su correspondiente máscara las clases que ésta contiene. Para cada clase, se genera una máscara y se aplica a la imagen para obtener sólo el cromosoma de esa clase. Esta última imagen es almacenada en la carpeta de salida en formato '.png' con un nombre del tipo 'nn_cc.png', donde nn es el número de cluster y cc el número de clase.

Parámetros:

imgs	Lista con las imágenes a guardar en formato de arreglo de NumPy.
mascaras	Lista con las máscaras correspondiente a cada imagen de ‘imgs’ en formato NumPy.
path_out	String que es el directorio de salida de las imágenes de salida.

def interfaz.leerImagenes (*path*, *path_out*)

Función que se encarga de leer del disco las imágenes de entrada a la herramienta.

Acepta imágenes del tipo [".jpg", ".bmp", ".png", ".tiff", ".jpeg", ".tif"] o un archivo ".txt" en el que haya en cada línea un path a las imágenes con el formato mencionado anteriormente. Esto último es para permitir procesar muchas imágenes a la vez.

Parámetros:

path	Directorio del archivo a procesar. Se verifica que al menos tenga 7 caracteres.
path_out	Directorio de salida, utilizado para generar un path de salida para cada imagen de entrada ya que se crea una carpeta para cada una de ellas.

Devuelve:

Tupla de dos listas: una con las imágenes cargadas y la segunda con el path de salida para cada una.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del módulo postprocesamiento

Paquete que incluye las funciones necesarias para el post-procesamiento de la predicción obtenida por la red convolucional con el objetivo de mejorar el desempeño de la misma.

Funciones

def **lineal2prob** (img)

Transforma la salida lineal de la red convolucional en probabilidades mediante la función softmax de PyTorch.

def **mayorVecino** (img, maskParcial)

Dada una zona de una imagen, devuelve la clase con más ocurrencias en los vecinos inmediatos.

def **EPI** (imgFinal, tamDespreciable=100)

Función que descarta las zonas menores al tamaño indicado directamente de la imagen y le asigna la clase que predomine en su vecindad inmediata.

def **knn** (img, umbralKNN=.9)

Postprocesamiento usando el algoritmo de k-NN provisto por la librería scikit-learn.

def **CC** (dataNP, umbralCC=0.9)

Funcion utilizada para hacer 0 los canales de las clases que tengan menor probabilidad al umbral y posteriormente corregir las probabilidades.

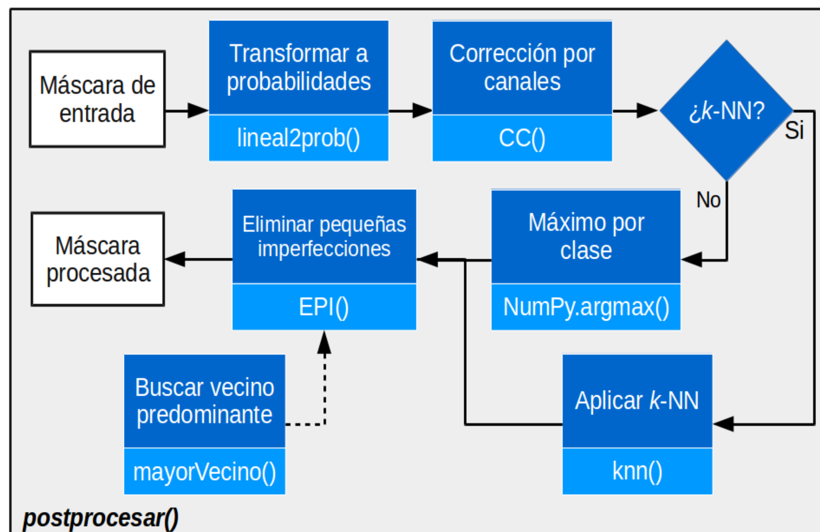
def **postprocesar** (data, umbralCC=0.9, umbralKNN=0, umbralEPI=100)

Función que integra a las demás para cumplir el objetivo del paquete.

Descripción detallada

Paquete que incluye las funciones necesarias para el post-procesamiento de la predicción obtenida por la red convolucional con el objetivo de mejorar el desempeño de la misma.

En la siguiente figura se ve como se interrelacionan las funciones definidas.



Primero se aplica la corrección por canales ‘CC()’ si el umbral ‘umbralCC’ fuera mayor a 0.

Luego, se aplica el algoritmo k-NN ‘knn()’ si ‘umbralKNN’ fuera mayor a 0, sino se elige la clase definitiva para cada píxel simplemente como la que tenga mayor probabilidad.

Por último, se aplica la eliminación de pequeñas imperfecciones ‘EPI()’ si ‘umbralEPI’ fuese mayor a 0.

---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Documentación de las funciones

def postprocesamiento.CC (*dataNP*, *umbralCC* = `0.9`)

Funcion utilizada para hacer 0 los canales de las clases que tengan menor probabilidad al umbral y posteriormente corregir las probabilidades.

Se determinan las clases confiables como aquellas que tengan al menos un píxel mayor a 'umbralCC'. Luego se hacen 0 los canales de las restantes clases y se recalculan las probabilidades para cada píxel de forma que sumen 1.

Parámetros:

img	Arreglo de NumPy de cuatro dimensiones correspondiente a la salida de la red.
umbralCC	Umbral utilizado para determinar las "clases confiables".

Devuelve:

Arreglo de NumPy de cuatro dimensiones con los canales de las "clases no confiables" llevados a cero.

def postprocesamiento.EPI (*imgFinal*, *tamDespreciable* = `100`)

Función que descarta las zonas menores al tamaño indicado directamente de la imagen y le asigna la clase que predomine en su vecindad inmediata.

Para ello, genera una máscara para cada clase y busca componentes conexas con el algoritmo 'findContours()' de OpenCV. Luego, analiza cada componente y si es menor al tamaño dado 'tamDespreciable', le asigna una clase indefinida número '24'. Por último, se detectan las componentes conexas que correspondan a dicha clase indefinida '24' y se le asigna la clase mayoritaria en su vecindad mediante '**mayorVecino()**'.

Parámetros:

imgFinal	Arreglo de NumPy de dos dimensiones que contiene en cada píxel la clase a la que éste pertenece.
tamDespreciable	Umbral de tamaño. Se descartan todas las zonas menores a él.

Devuelve:

Arreglo de NumPy de dos dimensiones con las zonas menores al umbral de tamaño reemplazadas por su vecino más concurrente.

def postprocesamiento.knn (*img*, *umbralKNN* = `.9`)

Post-procesamiento usando el algoritmo de k-NN provisto por la librería scikit-learn.

Se entrena k-NN con los píxeles que tengan una mayor probabilidad a 'umbralKNN' y luego se predice la clase de los restantes. Cada píxel tiene 24 valores puesto que posee una probabilidad para cada clase.

Parámetros:

img	Arreglo NumPy de 24 canales con probabilidades por clase.
umbralKNN	Umbral utilizado para determinar los "píxeles confiables".

Devuelve:

Arreglo de NumPy de dos dimensiones que contiene en cada píxel la clase a la que éste pertenece.

def postprocesamiento.lineal2prob (*img*)

Transforma la salida lineal de la red convolucional en probabilidades mediante la función softmax de PyTorch.

Para ello primero convierte la imagen en Tensor de PyTorch y luego reconvierte nuevamente a arreglo de NumPy.

Parámetros:

img	Arreglo 2D de NumPy con la salida de la red convolucional.

Devuelve:

Arreglo de NumPy con probabilidades de pertenencia a cada clase.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

def postprocesamiento.mayorVecino (***img*, *maskParcial*)**

Dada una zona de una imagen, devuelve la clase con más ocurrencias en los vecinos inmediatos.

Para ello, se aplica la operación morfológica de dilatación para obtener los vecinos de la zona indicada y luego se cuentan las clases mediante la función 'unique' de NumPy.

Parámetros:

img	Arreglo de NumPy 2D que contiene en cada píxel la clase a la que éste pertenece.
maskParcial	Arreglo de NumPy 2D que indica la zona en la que se quiere evaluar los vecinos. Se toman los píxeles mayores a cero.

Devuelve:

La clase con más ocurrencias en las inmediaciones de la zona indicada.

def postprocesamiento.postprocesar (***data*, *umbralCC*= `0.9`, *umbralKNN*= `0`, *umbralEPI*= `100`)**

Función que integra a las demás para cumplir el objetivo del paquete.

Parámetros:

data	Arreglo 4D de NumPy correspondiente a la salida de la red convolucional.
umbralCC	Umbral utilizado para la determinación de "clases confiables" para la corrección por canales. Por defecto es 0.9. Si es 0 no se hace.
umbralKNN	Umbral utilizado para la determinación de "píxeles confiables" para la corrección mediante el algoritmo de k-nn. Por defecto es 0 y no se hace.
umbralEPI	Umbral que se utiliza en la corrección de pequeñas imperfecciones. Por defecto es 100. Si es 0, no se aplica EPI.

Devuelve:

Arreglo de NumPy de dos dimensiones que contiene en cada píxel la clase a la que éste pertenece.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del módulo preprocesamiento

Paquete que incluye las funciones necesarias para el pre-procesamiento de la imagen de entrada y la extracción de los cromosomas del fondo.

Funciones

def **preprocesar** (img, tamCuadFondo=10, tamCuadUmbral=[0,100], maxTamAgujero=10, eeSize=7, umbralSegm=30, umbralArea=4000, umbralCH=0.8, TilesGridSize=8, ClipLimit=40)
    Función que integra a las demás para cumplir el objetivo del paquete.

def **esNegroFondo** (img, tamCuadFondo=10)
    Verifica si el fondo de la imagen es negro o no tomando ROIs de las esquinas de la imagen.

def **realceImagen** (img, TilesGridSize=8, ClipLimit=40)
    Realza una imagen en escala de grises aplicando CLAHE y luego normaliza en el rango [0-255].

def **umbralAdaptado** (img, tamCuadUmbral=[0,100])
    Función que segmenta mediante la combinación de umbrales adaptados de Otsu de distintos tamaños de ventana.

def **eliminarResiduos** (img, eeSize=7)
    Función que elimina los pequeños residuos mediante operaciones morfológicas.

def **rellenoAgujeros** (img, menoresA=0)
    Función que rellena los agujeros de una imagen binaria utilizando reconstrucción morfológica por dilatación geodésica.

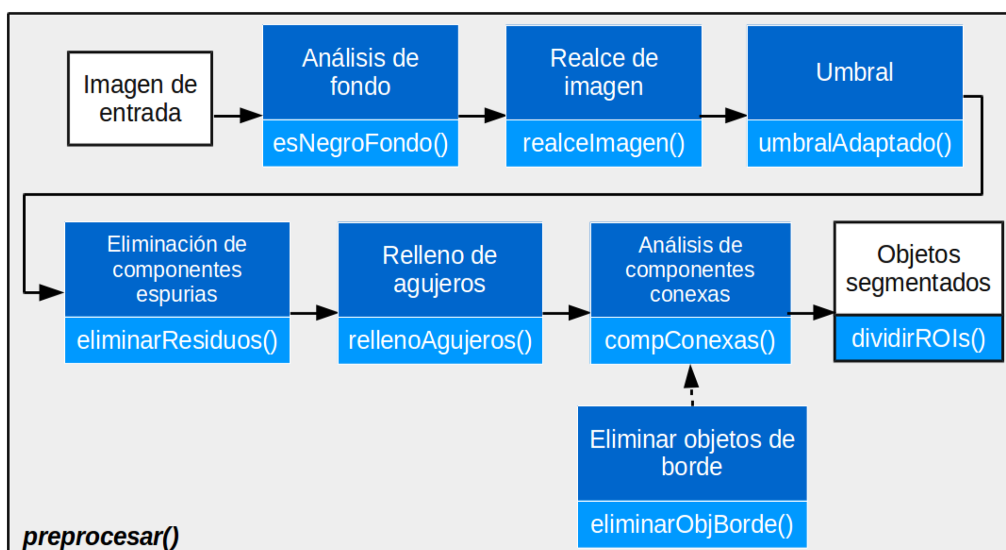
def **compConexas** (img, umbralSegm=30, umbralArea=4000, umbralCH=0.8)
    Función analiza cada componente conexa para determinar si corresponde a uno o más cromosomas, o a objetos no deseados.

def **dividirROIs** (data, mask, contours)
    Función que se encarga de separar los objetos de una imagen en un vector de subimágenes.

def **eliminarObjBorde** (img, umbralSegm=0)
    Función que elimina los objetos que están en contacto con el borde de una imagen verificando el tamaño del segmento que pertenece tanto al borde como al objeto.

Descripción detallada

Paquete que incluye las funciones necesarias para el pre-procesamiento de la imagen de entrada y la extracción de los cromosomas del fondo. En la imagen se ve como se relacionan las funciones.



sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Primero se verifica que el fondo sea negro, de lo contrario calcula la inversa de la imagen de entrada 'img', si 'tamCuadFondo' es mayor a 0. Luego, en este orden, se van aplicando las funciones del paquete para realzar la imagen 'realceImagen()', binarizarla con 'umbralAdaptado()', eliminar residuos mediante 'eliminarResiduos()' rellenarle los agujeros con 'rellenoAgujeros()'. Al resultado, se le analiza las componentes conexas para la eliminación de residuos mediante 'compConexas()'. Por último, se devuelven los objetos segmentados en forma de lista con su correspondiente máscara mediante el uso de la función 'dividirROIs()'.

Documentación de las funciones

def preprocesamiento.compConexas (*img*, *umbralSegm* = `30`, *umbralArea* = `4000`, *umbralCH* = `0.8`)

Función analiza cada componente conexa para determinar si corresponde a uno o más cromosomas, o a objetos no deseados.

Primero se eliminan los objetos del borde con la función **eliminarObjBorde()**. Luego, utilizando la función de detección de bordes de cada componente conexa provista por OpenCV, se analiza el área de cada una de ellas. De ser necesario, se obtiene su convex hull con una función de OpenCV y se calcula la proporción entre ambas áreas para comparar si el objeto debe descartarse o no.

Parámetros:

img	Imagen binaria en formato de arreglo de NumPy de dos dimensiones.
umbralSegm	Umbral que determina el tamaño máximo que puede tener el segmento en contacto con el borde de la imagen de un elemento. Si es mayor, se elimina.
umbralArea	Umbral que determina el tamaño máximo que puede tener un elemento para no ser considerado un posible residuo. Si es mayor, se utiliza el criterio del convex hull.
umbralCH	Umbral que determina la máxima proporción entre el área de un objeto y el área de su convex hull para no ser considerado un residuo. Si es mayor, se elimina.

Devuelve:

Lista que contiene los contornos de las componentes que no son residuos.

def preprocesamiento.dividirROIs (*data*, *mask*, *contours*)

Función que se encarga de separar los objetos de una imagen en un vector de subimágenes.

Para cada componente conexa de contours se calcula el mínimo rectángulo que la contiene con una función de OpenCV, se le aplica la máscara para la eliminación de otros objetos que pueda haber en el rectángulo, se le agrega un margen de dos píxeles a cada lado y luego se devuelven juntas en un vector de imágenes.

Parámetros:

data	Imagen en escala de grises en formato de arreglo de NumPy de dos dimensiones.
mask	Máscara que indica los píxeles que pertecen al fondo y a los objetos.
contours	Lista en la que cada elemento es un vector que contiene los contornos de un objeto.

Devuelve:

Tupla de listas. En la primera, cada elemento es una imagen de un objeto segmentado, mientras que en la segunta está su correspondiente máscara que indica con 255 los píxeles donde está el cromosoma y con 0 donde es fondo.

def preprocesamiento.eliminarObjBorde (*img*, *umbralSegm* = `30`)

Función que elimina los objetos que están en contacto con el borde de una imagen verificando el tamaño del segmento que pertenece tanto al borde como al objeto.

Se inicializa la imagen semilla que serán utilizadas para la reconstrucción morfológica por dilatación geodésica mediante la función provista por skimage. Antes de dicha reconstrucción, mediante erosiones con dos elementos estructurantes horizontal y vertical se eliminan de la semilla los elementos que tengan un segmento en contacto con el borde de la imagen menor al parámetro dado. Así, con la reconstrucción se obtienen los objetos a eliminar de la imagen binaria.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Parámetros:

img	Imagen binaria.
umbralSegm	Umbral que determina el tamaño máximo que puede tener el segmento en contacto con el borde de la imagen de un elemento. Si es mayor, se elimina. Si es 0 elimina todos los objetos que están en el borde.

Devuelve:

Imagen binaria sin los objetos del borde que no cumplen el criterio mencionado.

def preprocesamiento.eliminarResiduos (***img*, *eeSize*** **=** `7`**)**

Función que elimina los pequeños residuos mediante operaciones morfológicas.

Primero se realiza una erosión con un elemento estructurante cuadrado de tamaño eeSize con todos los componentes iguales a 255 mediante una función de OpenCV provista para tal fin. Luego, se lleva a cabo una reconstrucción morfológica por dilatación geodésica mediante una función de skimage.

Parámetros:

img	Imagen binaria en formato de arreglo de NumPy de dos dimensiones
eeSize	Tamaño del elemento estructurante que es usado para eliminar los residuos pequeños.

Devuelve:

Imagen binaria sin los residuos pequeños.

def preprocesamiento.esNegroFondo (***img*, *tamCuadFondo*** **=** `10`**)**

Verifica si el fondo de la imagen es negro o no tomando ROIs de las esquinas de la imagen.

Parámetros:

img	Imagen en escala de grises en formato de arreglo de NumPy 2D.
tamCuadFondo	Tamaño del cuadrado del fondo que se toma de las esquinas.

Devuelve:

Verdadero si el promedio de los valores de las ROIs es más cercano a 0, de lo contrario Falso.

def preprocesamiento.preprocesar (***img*, *tamCuadFondo*** **=** `10`**,** ***tamCuadUmbral*** **=** `[`**0, 100**`]`**,** ***maxTamAgujero*** **=** `10`**,** ***eeSize*** **=** `7`**,** ***umbralSegm*** **=** `30`**,** ***umbralArea*** **=** `4000`**,** ***umbralCH*** **=** `0.8`**,** ***TilesGridSize*** **=** `8`**,** ***ClipLimit*** **=** `40`**)**

Función que integra a las demás para cumplir el objetivo del paquete.

Primero se verifica que el fondo sea negro, de lo contrario calcula la inversa de la imagen de entrada 'img', si 'tamCuadFondo' es mayor a 0. Luego, en este orden, se van aplicando las funciones del paquete para realzar la imagen 'realceImagen()', binarizarla con 'umbralAdaptado()', eliminar residuos mediante 'eliminarResiduos()' rellenarle los agujeros con 'rellenoAgujeros()'. Al resultado, se le analiza las componentes conexas para la eliminación de residuos mediante 'compConexas()'. Por último, se devuelven los objetos segmentados en forma de lista con su correspondiente máscara mediante el uso de la función 'dividirROIs()'.

Parámetros:

img	Imagen en escala de grises.
tamCuadFondo	Tamaño del cuadrado del fondo que se toma de las esquinas. Si es 0, no se verifica.
tamCuadUmbral	Lista que contiene los tamaños de la ventana cuadrada que se utiliza en el umbral adaptado. Si es 0, calcula el umbral de Otsu sobre toda la imagen.
maxTamAgujero	Tamaño máximo que puede tener un agujero interno a un cromosoma para que se rellene. Cuando es mayor, no se rellena.
eeSize	Tamaño del elemento estructurante que es utilizado para la eliminación de los residuos pequeños.
umbralSegm	Umbral que determina el tamaño máximo que puede tener el segmento en contacto con el borde de la imagen de un elemento. Si es mayor, se elimina. Si es 0 elimina todos los objetos que están en el borde.
umbralArea	Umbral que determina el tamaño máximo que puede tener un elemento para no ser considerado un posible residuo. Si es mayor, se utiliza el criterio del convex hull.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

umbralCH	Umbral que determina la máxima proporción entre el área de un objeto y el área de su convex hull para no ser considerado un residuo. Si es mayor, se elimina.
TilesGridSize	Tamaño de las ventanas cuadradas que se aplican para CLAHE.
ClipLimit	Límite para el contraste utilizado en CLAHE.

Devuelve:

Tupla de listas. En la primera, cada elemento es una imagen de un objeto segmentado, mientras que en la segunta está su correspondiente máscara que indica con 255 los píxeles donde está el cromosoma y con 0 donde es fondo.

def preprocesamiento.realceImagen (*img*, *TilesGridSize* = `8`, *ClipLimit* = `40`)

Realza una imagen en escala de grises aplicando CLAHE y luego normaliza en el rango [0-255].

Primero se aplica Constrained Limited Adaptive Histogram Equalization (CLAHE), que aplica la ecualización por ROIs limitando la amplificación del contraste. Luego, se normaliza en el rango [0-255] mediante transformaciones afines. Ambas operaciones se realizan con las funciones provistas por OpenCV.

Parámetros:

img	Imagen en escala de grises.
TilesGridSize	Tamaño de las ventanas cuadradas que se aplican para CLAHE.
ClipLimit	Límite para el contraste utilizado en CLAHE.

Devuelve:

Imagen en escala de grises realzada.

def preprocesamiento.rellenoAgujeros (*img*, *menoresA* = `0`)

Función que rellena los agujeros de una imagen binaria utilizando reconstrucción morfológica por dilatación geodésica.

Se calcula el complemento de la imagen y se inicializa la semilla que serán utilizadas para la reconstrucción morfológica por dilatación geodésica mediante la función provista por skimage. Luego, se opera para obtener sólo los agujeros rellenos y se analizan los mismos utilizando la función de detección de bordes de cada componente conexa provista por OpenCV para saber si corresponde llenarlos o no.

Parámetros:

img	Imagen binaria en formato de arreglo de NumPy de dos dimensiones.
menoresA	Tamaño máximo que puede tener un agujero interno a un cromosoma para que se rellene. Cuando es mayor, no se rellena. Si es 0, rellena todos los agujeros.

Devuelve:

Imagen binaria con los agujeros rellenos en formato de arreglo de NumPy de dos dimensiones.

def preprocesamiento.umbralAdaptado (*img*, *tamCuadUmbral* = `[0,100]`)

Función que segmenta mediante la combinación de umbrales adaptados de Otsu de distintos tamaños de ventana.

La segmentación de la imagen se realiza calculando el umbral de Otsu por ventanas usando la función provista por OpenCV para tal fin y luego se interpola para llevar los umbrales al tamaño de la imagen de entrada. Luego, aplica el umbral a cada píxel. Por último, combina los resultados de cada tamaño de ventana utilizado mediante operaciones OR bit a bit entre ellas.

Parámetros:

img	Imagen en escala de grises a segmentar en formato de arreglo de NumPy de dos dimensiones.
tamCuadUmbral	Lista que contiene los tamaños de la ventana cuadrada que se utiliza en el umbral adaptado. Si es 0, calcula el umbral de Otsu sobre toda la imagen.

Devuelve:

Imagen binaria resultante de aplicar los umbrales correspondiente a cada tamaño. El resultado es un OR bit a bit entre cada máscara obtenida.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete RedW

Paquete que incluye la clase 'RedW' que define la arquitectura de la red W, similar a dos red U conectadas.

Clases

class **RedW**

Descripción detallada

Paquete que incluye la clase 'RedW' que define la arquitectura de la red W, similar a dos red U conectadas.

Dicha red U se determina en 'OverlapSegmentationNet.py'. Se hereda de la clase 'nn.Module' de PyTorch y se sobrecarga la función que se encarga de hacer el pasaje hacia adelante de la imagen 'forward()'.

---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete segmentador

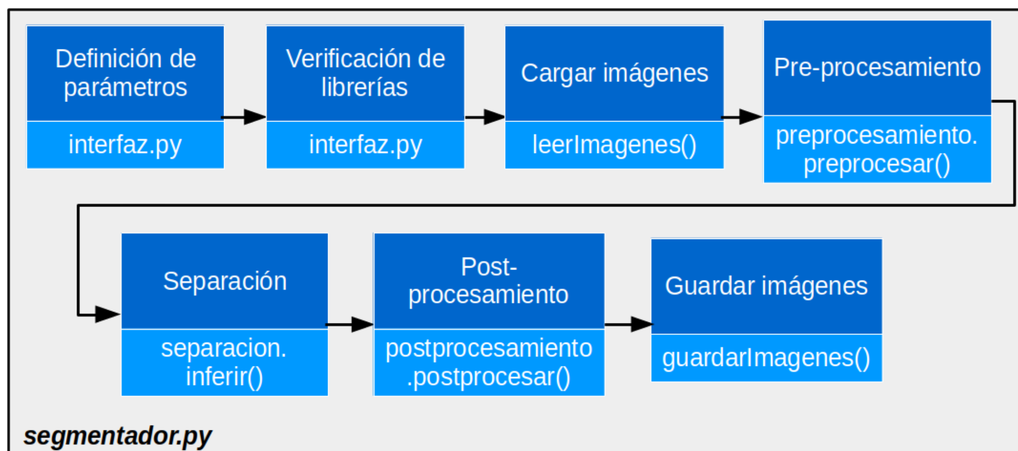
Archivo que incluye a los demás módulos para la integración y ejecución de la herramienta.

Descripción detallada

Archivo que incluye a los demás módulos para la integración y ejecución de la herramienta.

Primero define los parámetros mediante 'interfaz.py' para luego carga las imágenes de entrada con '**interfaz.leerImagenes()**'. Para cada imagen, conecta los módulos 'preprocesamiento', 'separacion' y 'postprocesamiento' pasando la salida de uno a la entrada del siguiente. A continuación, guarda las imágenes de salida utilizando la funcion '**interfaz.guardarImagenes()**'. El proceso se repite si hubiese más imágenes para finalmente guardar los parámetros por defecto mediante '**interfaz.saveDefaultArgs()**' si así se pidiese.

En la siguiente imagen se ve cómo se relacionan los bloques que lo componen.



---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del módulo separacion

Archivo que incluye la clase separacion, necesaria para separar un solapamiento de cromosomas mediante la aplicación de una red convolucional.

Clases

class **separacion**

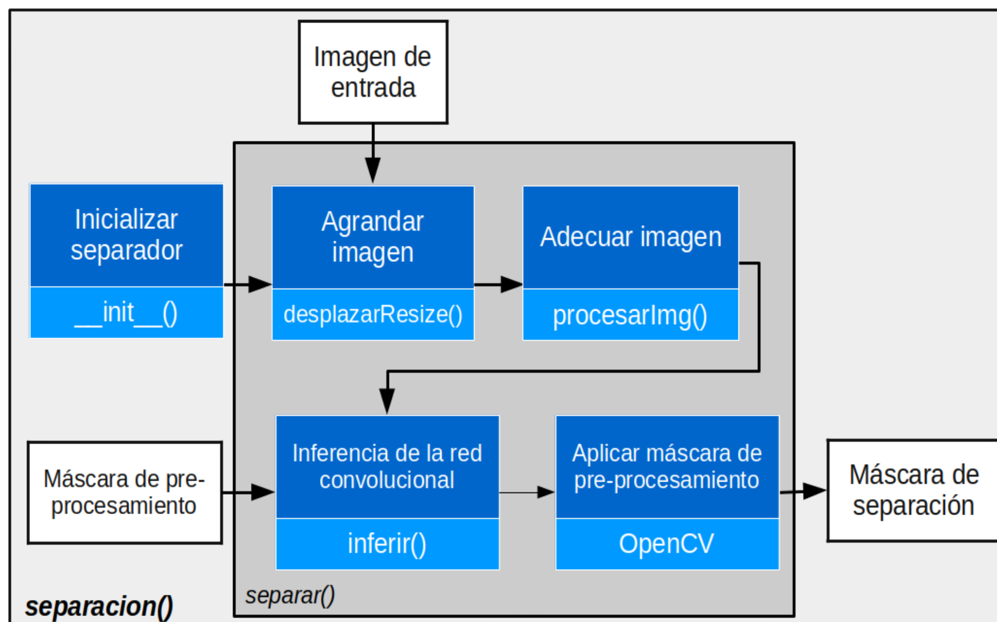
Descripción detallada

Paquete que incluye la clase separacion, necesaria para separar un solapamiento de cromosomas mediante la aplicación de una red convolucional.

Se inicializa la arquitectura de la red, se cargan sus parámetros entrenados y se manda a la GPU si se pidiera y pudiera en '__init__()'.

Luego se incluyen métodos para asemejar la imagen de entrada a los datos sintéticos generados 'procesarImg()' si así se quisiese, para redimensionar la imagen 'desplazarResize()' y para pasar la imagen por la red convolucional 'inferir()'. Por último, el método 'separar()' combina todas las funciones anteriores.

En la siguiente figura se ve cómo diseñó el mismo.



---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia de la Clase OverlapSegmentationNet.OverlapSegmentationNet

Métodos públicos

def **__init__** (self, canalesEntrada=1)

Constructor que define la estructura de la red convolucional.

def **forward** (self, x)

Sobrecarga de la función homóloga de PyTorch que realiza la pasada hacia adelante por la red.

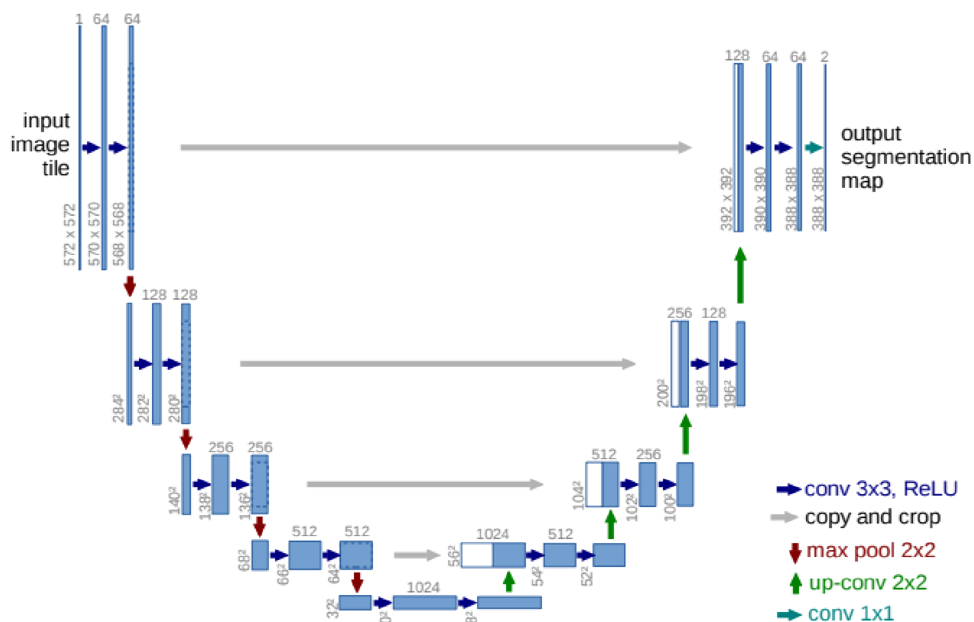
Atributos públicos

canalesEntrada: determina cuantos canales de entrada se definirán en la primera capa del modelo.

Documentación del constructor

def OverlapSegmentationNet.OverlapSegmentationNet.__init__ (*self*, *canalesEntrada*=1)

Define la estructura de la red convolucional. Ésta es similar a la red U de la siguiente imagen.



Documentación de las funciones miembro

def OverlapSegmentationNet.OverlapSegmentationNet.forward (*self*, *x*)

Sobrecarga de la función homóloga de PyTorch que realiza la pasada hacia adelante por la red.

Se encarga de determinar cómo se interrelacionan entre sí las capas definidas en el constructor.

Parámetros:

x	Entradas en formato Tensor de 4D de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales y las últimas dos al tamaño de la imagen.

Devuelve:

Salida de la red convolucional en formato Tensor de 4 dimensiones de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales (24 por la cantidad de clases) y las últimas dos al tamaño de la imagen.

La documentación para esta clase fue generada a partir del siguiente fichero:

0 OverlapSegmentationNet.py

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia de la Clase RedW.RedW

Métodos públicos

def **__init__** (self)
  Constructor que define la estructura de la red convolucional.

Documentación del constructor y destructor

def RedW.RedW.__init__ (*self*)

Constructor que define la estructura de la red convolucional, que consta de dos redes U definidas en ‘OverlapSegmentationNet.py’, en la que se conecta la salida de una a la entrada de la otra.

Documentación de las funciones miembro

def RedW.RedW.forward (*self*, *x*)

Sobrecarga de la función homóloga de PyTorch que realiza la pasada hacia adelante por la red convolucional.

Se encarga de determinar cómo se interrelacionan entre sí las capas definidas anteriormente en el constructor y de normalizar los datos con una media de 0.5 y un desvío de 0.5, mediante la función Normalize() de la librería torchvision.

Parámetros:

x	Entradas en formato Tensor de 4D de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales y las últimas dos al tamaño de la imagen.

Devuelve:

Salida de la red convolucional en formato Tensor de 4 dimensiones de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales (24 por la cantidad de clases) y las últimas dos al tamaño de la imagen.

La documentación para esta clase fue generada a partir del siguiente fichero:

1 RedW.py

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia de la Clase separacion.separacion

Métodos públicos

def **__init__** (self, path, model, gpu=True)
Constructor de la clase, encargado de cargar los parámetros entrenados de un modelo dado.

def **histGauss** (self, source, sigma=60)
Función que aplica histogram matching con una gaussiana de media 128 y desvío indicado.

def **desplazarResize** (self, img, tamImagenSalida, fondo=0)
Agranda la imagen que recibe al tamaño especificado, dejando en el centro la primera.

def **procesarImg** (self, img, std=60, blur=0.7, invertirImg=True, fondo=0)
Calculando la inversa de una imagen, aplica histogram matching y un filtro gaussiano de 3x3.

def **inferir** (self, img)
Dada una imagen, la segmenta mediante el pasaje de ella por la red convolucional.

def **separar** (self, imgs, imgs_mask=[], tamImagen=(256, 256), std=60, blur=0.7, invertirImg=True)
Función integradora de las demás del paquete.

Atributos públicos

model: red convolucional utilizada para la separación.

Documentación del constructor y destructor

def separacion.separacion.__init__ (*self*, *path*, *model*, *gpu* = `True`)

Constructor de la clase, encargado de cargar los parámetros entrenados de un modelo dado.

Tener en cuenta que el modelo debe estar guardado en el formato que se indica en el parámetro (por compatibilidad con el proyecto utilizado para el entrenamiento de la red).

Parámetros:

path	Path del archivo que contiene los parámetros del modelo. Debe estar en un diccionario con la clave "state_dict" y guardado serialmente (idealmente mediante la función 'save()' de PyTorch).
model	Modelo de PyTorch al cual se le cargarán los parámetros entrenados.

Devuelve:

Modelo de PyTorch con los parámetros cargados.

Documentación de las funciones miembro

def separacion.separacion.desplazarResize (*self*, *img*, *tamImagenSalida*, *fondo* = `0`)

Agranda la imagen que recibe al tamaño especificado, dejando en el centro la primera.

Si la imagen fuera mayor al tamaño deseado, éste se duplica y se intenta nuevamente.

Parámetros:

img	Imagen a agrandar en formato de arreglo NumPy de dos dimensiones.
tamImagenSalida	Tupla que indica el tamaño de salida deseado.
fondo	Indica el color del fondo para rellenar la imagen agrandada.

Devuelve:

Imagen agrandada en formato de arreglo NumPy de dos dimensiones.

def separacion.separacion.histGauss (*self*, *source*, *sigma* = `60`)

Función que aplica histogram matching con una gaussiana de media 128 y desvío indicado.

Parámetros:

source	Imagen o píxeles a los que se aplicará.
sigma	Desvío de la gaussiana que se utilizará.

Devuelve:

Imagen o píxeles corregidos según la gaussiana.

def separacion.separacion.inferir (***self*, *img*)**

Dada una imagen, la segmenta mediante el pasaje de ella por la red convolucional.

También se encarga de llevar la imagen del rango [0,255] al rango [0,1], del pasaje de NumPy a tensor de PyTorch para la inferencia y del pasaje inverso para la devolución del resultado.

Parámetros:

img	Imagen a segmentar en formato de arreglo numpy de dos dimensiones.

Devuelve:

Imagen en formato NumPy de dos dimensiones indicando en cada píxel el número de clase.

def separacion.separacion.procesarImg (***self*, *img*, *std* = `60`, *blur* = `0.7`, *invertirImg* = `True`, *fondo* = `0`)**

Calculando la inversa de una imagen, aplica histogram matching y un filtro gaussiano de 3x3.

Parámetros:

img	Imagen a segmentar en formato de arreglo NumPy de dos dimensiones.
std	Desvío utilizado para aplicar histogram matching con una gaussiana.
blur	Desvío utilizado en el filtro gaussiano.
invertirImg	Booleano que indica si se calcula la inversa de la imagen o no.
fondo	Indica el color del fondo para excluirlo del calculo del histogram matching.

Devuelve:

Imagen en formato NumPy de dos dimensiones indicando en cada píxel el número de clase.

def separacion.separacion.separar (***self*, *imgs*, *imgs_mask* = `[]`, *tamImagen* = `(256,256)`, *std* = `60`, *blur* = `0.7`, *invertirImg* = `True`)**

Función integradora de las demás del paquete.

También se encarga de manejar múltiples inferencias sin tener que recargar el modelo.

Parámetros:

imgs	Lista de imágenes a segmentar en formato de arreglo NumPy 2D.
imgs_mask	Lista de máscaras de las imágenes a segmentar. Si la lista tiene longitud distinta a imgs, no se aplica.
tamImagen	Tupla que indica el tamaño al que se agrandará la imagen antes de pasar por la red convolucional.
std	Desvío utilizado para histogram matching. Si es 0 no se aplica.
blur	Desvío utilizado para aplicar filtro gaussiano de 3x3. Si es 0 no se aplica.

Devuelve:

Tupla con dos listas. Una de la imagen original ampliada y otra de imágenes en formato de arreglo NumPy 2D indicando en cada píxel el número de clase.

La documentación para esta clase fue generada a partir del siguiente fichero:

2 separacion.py

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Apéndice B

Documentación de la generación de datos

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

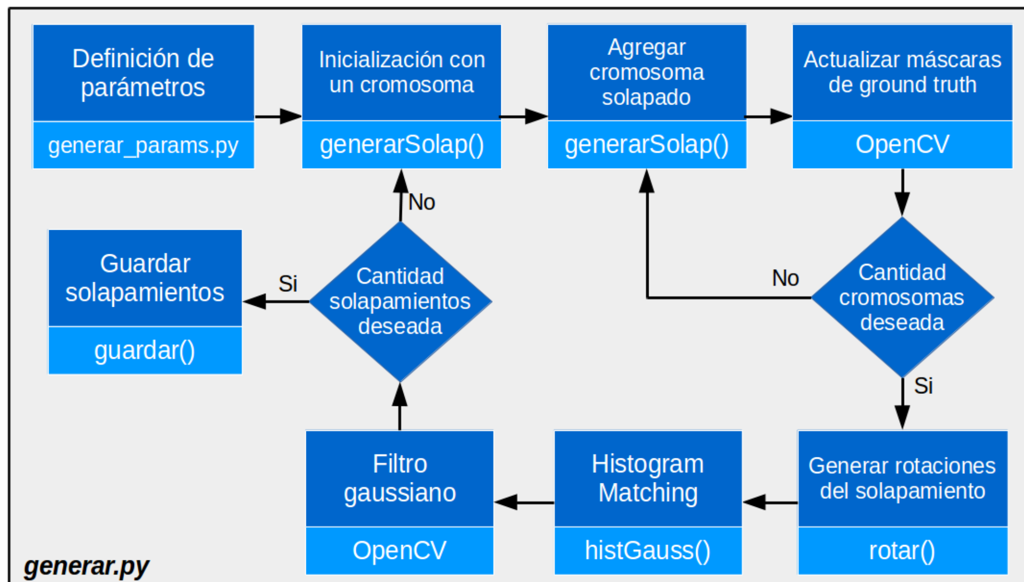
Referencia del módulo generar

Script que aplica las funciones definidas en 'generarAux.py' para la generación de solapamientos sintéticos a partir de los parámetros definidos en '**generar_params.py**'.

Descripción detallada

Script que aplica las funciones definidas en 'generarAux.py' para la generación de solapamientos sintéticos a partir de los parámetros definidos en '**generar_params.py**'.

En la siguiente imagen puede verse cómo se relacionan entre sí.



Los cromosomas individuales en imágenes separadas se obtienen mediante la función 'procesarCariogramas()' definida en el archivo 'separarKaryo.py'.

En el archivo 'generar_params.py' se definen los parámetros con los que se crearán los solapamientos sintéticos. Ver la documentación de dicho archivo para ver listados los parámetros.

El proceso comienza con la creación de una imagen en blanco con un cromosoma e inicializando la lista que contendrá el ground truth de la correcta separación del solapamiento ('masksCanales'). Posteriormente, se entra en un bucle en el que se desplaza un poco al azar el solapamiento que se tiene hasta el momento con 'desplazarResize()' antes de agregar un cromosoma al azar al mismo con 'generarSolap()'. Luego, se actualizan las máscaras de ground truth teniendo en cuenta qué cromosomas son visibles mediante el uso de operaciones lógicas entre ellas de OpenCV. Se sale del mencionado bucle cuando se logra un solapamiento con la cantidad buscada de cromosomas o se recomienza por algún error.

Una vez que se tiene un solapamiento, el mismo rota un ángulo al azar la cantidad de veces indicada en 'generar_params,py' mediante la función 'rotar()'. Luego, a cada uno se le aplica el proceso de histogram matching con un desvío al azar entre los parámetros definidos con el uso de la función 'histGauss()'. Seguido a esto, cada imagen se filtra con un kernel gaussiano de 3x3 con una media al azar entre otros parámetros definidos. Por último, los solapamientos se centran en una imagen de tamaño definido en 'generar_params.py' con 'desplazarResize()'.

El proceso se repite hasta obtener la cantidad de solapamientos deseada para las cantidades de cromosomas configuradas. Éstos se guardan mediante la función 'guardar()'.

---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del archivo generar_params.py

Script que define los parámetros que se utilizan para la generación de los datos.

Descripción detallada

Script que define los parámetros que se utilizan para la generación de los datos.

En la siguiente tabla se detallan los parámetros, indicando los que trae por defecto que corresponden al dataset Gen5todosKaryoNormVar.

Parámetro	Detalle	Valor
directorio	Carpeta en que están los cromosomas individuales.	“./Separados/raw/”
out_path	Directorio de salida de los datos generados.	“./salida/”
total_cario	Cantidad total de cariogramas.	536
cuantos_raw	Cantidad deseada de archivos de salida.	12
intervalo	Cantidad de cariogramas que se utilizan para generar cada archivo.	40
tamImagen	Tupla que indica el tamaño de las imágenes genradas.	(256,256)
cant_angles	Cantidad de rotaciones que se realizan para cada solapamiento.	4
CantDespl	Cantidad de desplazamientos que se realizan para cada rotación.	1
train_cant	Cantidad deseada de solapamientos generados por cariograma y por cantidad de cromosomas para train.	10
test_cant	Similar al anterior para test.	10
cant_crom_min	Cantidad mínima de cromosomas por solapamiento.	2
cant_crom_max	Cantidad máxima de cromosomas por solapamiento.	5
porcMinSolap	Porcentaje mínimo de solapamiento respecto al cromosoma agregado.	0,01
porcMaxSolap	Porcentaje máximo de solapamiento medido respecto al nuevo cromosoma que se agrega.	0,27
minNewEP	Lista en la que se elige al azar la cantidad mínima de end points nuevos por solapamiento.	[1,2]
maxIntentos	Cantidad máxima de intentos para agregar un nuevo cromosoma antes de cambiarlo por otro.	20
maxIntentosAux	Máximo de intentos para agregar un cromosoma con ‘generarSolap()’.	20
minStd	Desvío mínimo para la aplicación de ‘histGauss()’.	50
maxStd	Desvío máximo para la aplicación de ‘histGauss()’.	70
minBlur	Desvío mínimo para la aplicación del filtro gaussiano de 3x3.	0
maxBlur	Desvío máximo para la aplicación del filtro gaussiano de 3x3.	0,7
cant_clases	Cantidad de clases contando la correspondiente al fondo.	24

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete generarAux

Archivo que incluye funciones auxiliares utilizadas por 'generar.py' tales como rotar, recortar, agregar cromosoma, guardar imágenes, etc.

Funciones

def **rotar** (img, mascara, angle, maxTam=(0, 0))

Dada una imagen y su máscara, los rota una cantidad de ángulos con la función 'rotate()' de scikit-image, rellenando con blanco los píxeles extra de la imagen y con negro los de las máscaras.

def **desplazarResize** (img, mascara, tamImagenSalida, despl=-1, fondo=255)

Función que re-dimensiona una imagen y su máscara, dando la posibilidad de desplazar la original dentro de la nueva más grande.

def **cantEP** (mask)

Función que calcula la cantidad de end points que tiene una imagen binaria.

def **suavizarPegada** (data1, mask1, data2, mask2, sigmaGauss=1.5)

Adiciona dos clusters de cromosomas haciendo que los píxeles de borde queden suaves.

def **generarSolap** (img1, mascara1, img2, porcMinSolap, porcMaxSolap, minNewEP, maxIntentos=30)

Agrega a un solapamiento 'img1' otro cromosoma 'img2', quedando 'img2' por detrás de 'img1'.

def **guardar** (salida, tamImagen, nombre, folder="./")

Función que se encarga de guardar las imágenes generadas como archivo ".npz".

def **recortar** (img, mascaras, margen, fondo=255)

Función que dada una imagen y su máscara o lista de máscaras las recorta al mínimo cuadrado que la contiene dejando un margen dado por 'margen'.

def **histGauss** (source, sigma=60)

Función que aplica histogram matching con una gaussiana de media 128 y desvío indicado.

Documentación de las funciones

def generarAux.cantEP (*mask*)

Función que calcula la cantidad de end points que tiene una imagen binaria.

Se utiliza la función 'thinning()' de OpenCV para el cálculo del esqueleto morfológico, para luego convolucionar con un filtro de 3x3 con todos números 1 y un 10 en el centro. Entonces, se cuentan los píxeles que resulten con un valor de 11: quiere decir que tienen sólo un vecino.

Parámetros:

mask	Imagen binaria de dos dimensiones en formato arreglo de NumPy.

Devuelve:

Entero que indica la cantidad de end points de la máscara.

def generarAux.desplazarResize (*img*, *mascara*, *tamImagenSalida*, *despl*= `-1`, *fondo*= `255`)

Función que re-dimensiona una imagen y su máscara, dando la posibilidad de desplazar la original dentro de la nueva más grande.

Primero se genera una imagen con color de fondo igual a 'fondo' y máscaras de fondo negro. Luego, según 'despl' se genera desplazamientos al azar (-1), se centra (-2) o se aplica (si fuera una tupla que indique el desplazamiento en x y en y). Finalmente se aplica dicho desplazamiento.

Parámetros:

img	Imagen 2D en formato arreglo de NumPy.
mascara	Imagen binaria 2D en formato arreglo de NumPy o lista que contenga imágenes de ese tipo, ya que pueden ser las máscaras de los cromosomas que ya se agregaron.
tamImagenSalida	Tupla que indica el tamaño que debe tener la imagen de salida.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

despl	Tupla que indica el desplazamiento en x y en y. Si es un entero igual a -1, se calcula un desplazamiento al azar que no supere el tamaño de la imagen. Si es -2, la imagen original se centra dentro el nuevo lienzo.
fondo	Color de fondo con el que se rellena la imagen redimensionada.

Devuelve:

Tupla con imagen y máscara/lista de máscaras 2D en formato de arreglo de NumPy.

def generarAux.generarSolap (***img1*, *mascara1*, *img2*, *porcMinSolap*, *porcMaxSolap*, *minNewEP*, *maxIntentos*** **=** `30`**)**

Agrega a un solapamiento 'img1' otro cromosoma 'img2', quedando 'img2' por detrás de 'img1'.

Primero se obtiene la máscara del cromosoma a agregar y se rellenan los agujeros con la función 'rellenoAgujeros()' del módulo de 'preprocesamiento'. Luego, se itera rotando 'img2' un ángulo al azar y agregándolo al solapamiento hasta que cumpla con las condiciones de porcentaje de solapamiento y cantidad de end points nuevos. Si se supera el máximo de intentos, se devuelve una máscara con todos 255 indicando que se falló y que se debe intentar agregar otro cromosoma.

Parámetros:

img1	Imagen 2D en formato arreglo de NumPy correspondiente al solapamiento actual.
mask1	Imagen binaria 2D en formato arreglo de NumPy.
img2	Imagen 2D en formato NumPy correspondiente al cromosoma que se agregará.
porcMinSolap	Porcentaje mínimo de solapamiento que debe tener el nuevo cluster medido respecto al cromosoma que se agrega.
porcMaxSolap	Porcentaje máximo de solapamiento del cromosoma agregado.
minNewEP	Cantidad mínima de end points que debe tener el nuevo cluster.
maxIntentos	Cantidad máxima de intentos que se prueba agregar el nuevo cromosoma.

Devuelve:

Tupla que contiene la imagen generada y una lista con las máscaras de cada cromosoma. Si se supera la cantidad máxima de intentos, la última máscara de la lista es una imagen con todos los píxeles 255.

def generarAux.guardar (***salida*, *tamImagen*, *nombre*, *folder*** **=** `"./"`**)**

Función que se encarga de guardar las imágenes generadas como archivo ".npz".

Primero se definen las matrices de NumPy en las que se almacenan las imágenes generadas y las máscaras. Éstas tienen 4 dimensiones tal como se utilizan en PyTorch posteriormente: la primera es la cantidad de datos, la segunda la cantidad de canales (siempre 1 en este caso) y las restantes el tamaño de la imagen. Luego se recorren las listas de 'salida' para ir completándolas.

Por último se guarda la inversa de las imágenes mediante la función 'savez_compressed()' de NumPy puesto que la misma es eficiente comprimiendo matrices sparse. Los datos se guardan bajo la clave 'data' y los máscaras bajo la clave 'maskLineal'.

Parámetros:

salida	Tupla que contiene tres listas: una con las imágenes de los solapamientos generados, otra que es a su vez una lista con las máscaras correspondientes a cada cromosoma y una tercera que es una lista que contiene el número de clase del mismo.
tamImagen	Tupla con el tamaño de la imagen que se guarda.
nombre	Nombre del archivo '.npz' que se genera.
folder	Carpeta de salida en que se guarda el archivo generado.

def generarAux.histGauss (***source*, *sigma*** **=** `60`**)**

Función que aplica histogram matching con una gaussiana de media 128 y desvío indicado.

Parámetros:

source	Imagen o píxeles a los que se aplicará.
sigma	Desvío de la gaussiana que se utilizará.

Devuelve:

Imagen o píxeles corregidos según la gaussiana.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

def generarAux.recortar (***img*****,** ***mascaras*****,** ***margen*****,** ***fondo*** **=** `255`**)**

Función que dada una imagen y su máscara o lista de máscaras las recorta al mínimo cuadrado que la contiene dejando un margen dado por 'margen'.

Se comienza buscando el mínimo rectángulo que contiene a los cromosomas de la imagen, sabiendo el color de fondo de la misma, con la función 'boundingRect' de OpenCV. Se genera la imagen de salida con color 'fondo' y se copia la información relevande de la imagen original. También se generan máscaras de salida con el mismo rectángulo obtenido anteriormente.

Parámetros:

img	Imagen 2D en formato arreglo de NumPy.
mascaras	Imagen binaria 2D en formato arreglo de NumPy o lista que contenga imágenes de ese tipo, ya que pueden ser las máscaras de los cromosomas que se fueron agregando.
margen	Entero que indica el margen que se deja a cada lado del mínimo rectángulo.
fondo	Color de fondo que se utiliza en la búsqueda del mínimo rectángulo.

Devuelve:

Tupla con imagen y máscara/lista de máscaras 2D en formato de arreglo de NumPy.

def generarAux.rotar (***img*****,** ***mascara*****,** ***angle*****,** ***maxTam*** **=** `(0,0)`**)**

Dada una imagen y su máscara, los rota una cantidad de ángulos con la función 'rotate()' de scikit-image, rellenando con blanco los píxeles extra de la imagen y con negro los de las máscaras.

Luego aplica una apertura morfológica en las máscaras ya que la rotación provoca una interpolación en los bordes de la imagen que produce una difuminación del mismo. Además, estas máscaras se aplican en la imagen para evitar dicha interpolación. Por último se recorta la imagen con '**recortar()**' porque la rotación suele agrandar la imagen. Antes de devolverla, se controla que al rotar la imagen no supere el tamaño máximo especificado (si 'maxTam' no es (0,0)).

Parámetros:

img	Imagen 2D en formato arreglo de NumPy.
mascara	Imagen binaria 2D en formato arreglo de NumPy o lista que contenga imágenes de ese tipo, ya que pueden ser las máscaras de los cromosomas que se fueron agregando.
angle	Ángulo que se giran las imágenes.
maxTam	Tupla que indica el tamaño máximo que puede tener la imagen. Si es (0,0) no se controla.

Devuelve:

Tupla con imagen y máscara/lista de máscaras 2D en formato de arreglo de NumPy. Devuelve -1 si la imagen rotada es mayor a 'maxTam'.

def generarAux.suavizarPegada (***data1*****,** ***mask1*****,** ***data2*****,** ***mask2*****,** ***sigmaGauss*** **=** `1.5`**)**

Adiciona dos clusters de cromosomas haciendo que los píxeles de borde queden suaves.

Para ello, primero se obtiene el contorno de la zona de solapamiento mediante operaciones morfológicas de OpenCV. Luego se suman las dos imágenes de cromosomas 'data1' y 'data2' normalmente. A continuación, se le aplica un filtro gaussiano con desvío igual a 'sigmaGauss' en el borde obtenido anteriormente.

Parámetros:

data1	Imagen 2D en formato arreglo de NumPy.
mask1	Imagen binaria 2D en formato arreglo de NumPy.
data2	Imagen 2D en formato arreglo de NumPy, con la zona de solapamiento ya quitada previamente de forma de permitir la suma con 'data1' para generar el solapamiento.
mask2	Imagen binaria 2D en formato arreglo de NumPy, con la zona de solapamiento ya quitada previamente.
sigmagauss	Desvío utilizado en el filtro gaussiano de 3x3 para la suavización del borde de la zona de solapamiento.

Devuelve:

Tupla con la imagen y una lista con las máscaras 2D en formato de arreglo de NumPy.

Referencia del paquete separarKaryo

Archivo que define y utiliza las funciones para extraer los cromosomas de los cariogramas para la posterior generación de datos.

Funciones

def **compare** (c1, c2)
: *Dada dos listas de contornos, compara cuál de las dos está más arriba y a la izquierda tomando como referecia el punto más inferior y a la derecha de cada componente conexa.*

def **cmp_to_key** (mycmp)
: *Función que transforma la función de comparación en una 'key' aceptada por la función 'sorted' que viene por defecto en Python.*

def **dividirCariograma** (img, minTam=200)
: *Función que dada una imagen que es un cariograma con fondo blanco, guarda una subimagen de cada cromosoma en salida.*

def **guardarSeparados** (separados, dirSalida)
: *Función que se encarga de guardar los cromosomas extraídos en la carpeta de salida.*

def **procesarCariogramas** (directorio, cuantos, salida)
: *Función que toma tantos cariogramas como se le indique para extraer los cromosomas individuales.*

Descripción detallada

Archivo que define y utiliza las funciones para extraer los cromosomas de los cariogramas para la posterior generación de datos.

En la última línea del archivo contiene el llamado a la función 'procesarCariogramas()' utilizado para extraer los cromosomas de todos los cariogramas que poseen 46 cariogramas.

Documentación de las funciones

def separarKaryo.cmp_to_key (*mycmp*)

Función que transforma la función de comparación en una 'key' aceptada por la función 'sorted' que viene por defecto en Python.

Parámetros:

mycmp	Función de comparación.

Devuelve:

Clase compatible con el parámetro 'key' de 'sorted'.

def separarKaryo.compare (*c1*, *c2*)

Dada dos listas de contornos, compara cuál de las dos está más arriba y a la izquierda tomando como referencia el punto más inferior y a la derecha de cada componente conexa.

Se utiliza para ordenar espacialmente los contornos obtenidos del cariograma de arriba hacia abajo y de izquierda a derecha con la función 'sorted' que viene integrada en Python.

Parámetros:

c1	Lista de puntos obtenidos con la función 'findContours()' de OpenCV: es un arreglo de enteros de 3 dimensiones.
c2	Lista de puntos obtenidos con la función 'findContours()' de OpenCV: es un arreglo de enteros de 3 dimensiones.

Devuelve:

Un entero que es 0 si son iguales, mayor a 0 si 'c1' es mayor y menor a 0 si 'c2' es mayor.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

def separarKaryo.dividirCariograma (*img*, *minTam* = 200)

Función que dada una imagen que es un cariograma con fondo blanco, guarda una sub-imagen de cada cromosoma en salida.

Les agrega un número incremental para evitar repetidos y la extensión '.tiff'. Además las ordena de acuerdo a su posición espacial, empezando por la esquina superior izquierda y continuando hacia la derecha. Para ello, aplica un umbral de 254 sabiendo que el fondo de los mismos es blanco. A la máscara obtenida, le rellena los agujeros con la función 'rellenoAgujeros()' del módulo de 'preprocesamiento' de la herramienta de segmentación automática de cromosomas y se detectan las componentes conexas con 'findContours()' de OpenCV. Para cada componente conexa, si es mayor al tamaño indicado 'minTam', la imagen se agrega a la lista que se devuelve posteriormente.

Parámetros:

img	Cadena de texto que indica el directorio del cariograma.

Devuelve:

Lista de imágenes que contienen los cromosomas extraídos del cariograma.

def separarKaryo.guardarSeparados (*separados*, *dirSalida*)

Función que se encarga de guardar los cromosomas extraídos en la carpeta de salida.

Se guardan de la forma "cco.png", donde 'cc' es la clase del cromosoma y 'o' es igual a 'a' o a 'b', para guardar las dos ocurrencias de cada clase en el cariograma. Se utiliza la función 'imwrite()' de OpenCV.

Parámetros:

separados	Lista de imágenes de dos dimensiones con formato arreglo de NumPy.
dirSalida	Cadena de texto que indica la carpeta de salida.

def separarKaryo.procesarCariogramas (*directorio*, *cuantos*, *salida*)

Función que toma tantos cariogramas como se le indique para extraer los cromosomas individuales.

Primero se obtienen todos los archivos del directorio pasado como parámetro y luego se van leyendo una a una en un bucle. Para cada una se crea una carpeta dentro del directorio de salida con nombre "rawN", donde N es el número de cariograma a procesar. Luego se extraen los cromosomas con '**dividirCariograma()**' y se guardan con '**guardarSeparados()**' si es que tienen la cantidad de cromosomas indicada en 'cuantos'. Además, en la carpeta de salida se guarda el cariograma original para saber de donde provienen los cromosomas extraídos.

Parámetros:

directorio	Cadena de texto que indica la carpeta en la que se encuentran los cariogramas.
cuantos	Cantidad de cromosomas que debe tener el cariograma para procesarlos. Usado para obtener sólo los que poseían 46 cromosomas.
salida	Cadena de texto que indica la carpeta de salida.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Apéndice C

Documentación del entrenamiento de la red convolucional

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete dataLoader

Paquete que define las clases y funciones necesarias para el cargado de los datos.

Clases

class **genDataset**

Funciones

def **splitAndLoader** (dataset, val_split, batch_size, random_seed=42)

Función que genera una partición de validación y transforma el dataset en un 'DataLoader' de Pytorch.

Descripción detallada

Paquete que define las clases y funciones necesarias para el cargado de los datos.

Define la clase '**genDataset**', la cual hereda de la clase 'Dataset' de PyTorch, a fin de cargar en él los datos. La misma almacena los datos en memoria con formato NumPy y sólo transforma a Tensor de PyTorch los que está usando en el momento para pasar la red. Luego, define la función '**splitAndLoader()**' que toma el dataset generado para crear un 'DataLoader' de PyTorch, el cual provee facilidades para iterar por batch sobre el dataset. Además, la función anterior reserva una partición de los datos para validación, si así se indicase.

Documentación de las funciones

def dataLoader.splitAndLoader (*dataset*, *val_split*, *batch_size*, *random_seed* = 42)

Función que genera una partición de validación y transforma el dataset en un 'DataLoader' de Pytorch.

Si el porcentaje de los datos a reservar para validación 'val_split' es mayor a 0, se generan dos DataLoaders. Uno para validación con el porcentaje indicado y otro para entrenar con lo restante. Esta división se realiza mezclando los índices de los datos con la función 'shuffle()' de NumPy, a la cual se le puede proporcionar una semilla 'random_seed' para poder hacer reproducible el experimento. Luego, con la función 'SubsetRandomSampler()' de PyTorch se generan las dos particiones para crear los DataLoaders. A éstos últimos también debe pasársele el tamaño de batch que se utilizará.

Parámetros:

dataset	Dataset de la clase definida anteriormente al cual se procesará.
val_split	Porcentaje de datos reservados para validación. Si fuese 0, no se reserva ninguno y se devuelve sólo un DataLoader con todos los datos.
batch_size	Tamaño de batch que usará el DataLoader.
random_seed	Semilla que se le pasa a NumPy para hacer reproducible el experimento. Por defecto, es igual a 42.

Devuelve:

Una tupla que contiene el DataLoader de entrenamiento y el DataLoader de validación, o simplemente un único DataLoader con todos los datos si 'val_split' es 0.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del archivo inferir

Script que itera sobre los datos pasados como parámetro para calcular la salida de la red convolucional.

Descripción detallada

Script que itera sobre los datos pasados como parámetro para calcular la salida de la red convolucional.

Es una versión simplificada del archivo **'trainModel.py'** ya que simplemente infiere sobre los datos, no realiza partición de validación ni utiliza datos de test o de cromosomas reales. Además, permite la posibilidad de realizar la inferencia sin conocer el ground truth de la imagen mediante el parámetro 'hayLabels'. Otra diferencia es que este archivo obviamente no realiza gráficas de pérdida, recall y Jaccard por época, sino que sólo las calcula en el caso de que haya ground truth.

Tampoco carga el optimizador ya que la idea no es continuar un entrenamiento.

---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del archivo inferir_params.py

Script que define los parámetros que se utilizan para realizar inferencias con la red convolucional.

Descripción detallada

Script que define los parámetros que se utilizan para realizar inferencias con la red convolucional.

En la siguiente tabla se detallan los parámetros, indicando los que trae por defecto (que fueron utilizados para inferir con la red W sobre cromosomas reales).

Parámetro	Detalle	Valor por defecto
batch_size	Tamaño del batch en que se toman los datos para pasar por la red convolucional.	10
nombreArq	Nombre de la arquitectura de red (sin ‘.py’).	“RedW”
nombreData	Nombre de los datos (sin ‘.npz’)	“reales”
hayLabels	Booleano que indica si los datos poseen ground truth o no.	False
cant_train	Cantidad de archivos de entrenamiento usados (sólo cuando se usan archivos de entrenamiento)	1
nameData	Cadena de texto que se le agrega a los nombres de los archivos de salida	“”
pathAnterior	Directorio de los parámetros entrenados del modelo, utilizado para continuar un entrenamiento previo.	“./modelo/entrenado”
DATA_PATH	Carpeta en que se encuentran los datos	“./data/”
OUT_PATH	Directorio en que se guardan los archivos de salida	“./salida/”+nameData
PREDICT_PATH	Directorio en que se guardan las predicciones de la red como imágenes	"./salida/"+"predicciones/" + nameData
verGraficas	Booleano que indica si se quieren ver las gráficas online	False
cadaCuantas	Entero que indica cada cuantas imágenes guardar las máscaras predichas.	1

---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete modelLoader

Paquete que define las funciones necesarias para el cargado y guardado de un modelo entrenado.

Funciones

def **saveModel** (epoch, model, optimizer, is_best=False, path="./model/")
Función que guarda el modelo en el directorio indicado con la función 'save()' de PyTorch.
def **loadModel** (path, model, optimizer=None)
Función que carga el modelo desde el directorio indicado con la función 'load()' de PyTorch.
def **getOptimizer** (model, name, learning_rate=0.001, momentum=0.9)
Función que devuelve el optimizador de PyTorch requerido.
def **getLossFunc** (name)
Función que devuelve la función de pérdida de PyTorch requerida.

Descripción detallada

Paquete que define las funciones necesarias para el cargado y guardado de un modelo entrenado.

Además, define funciones para obtener funciones de pérdida y optimizadores.

Documentación de las funciones

def modelLoader.getLossFunc (*name*)

Función que devuelve la función de pérdida de PyTorch requerida.

Pueden ser "BCE" (Binary Cross Entropy), "BCEwLogits" (Binary Cross Entropy with Logits), "CrossEntropy" y "MSE" (Mean Squared Error).

Parámetros:

name	Cadena de texto que indica la función a devolver.

Devuelve:

Función de pérdida indicada de PyTorch.

def modelLoader.getOptimizer (*model*, *name*, *learning_rate* = `0.001`, *momentum* = `0.9`)

Función que devuelve el optimizador de PyTorch requerido.

Pueden ser gradiente descendiente estocástico "SGD", "Adam" o "Adadelta". En el primer caso se debe pasar la tasa de aprendizaje 'learning_rate' y el momentum 'momentum'. En el segundo, sólo la tasa de aprendizaje. Mientras que en el último no necesita ninguno de los dos ya que son parámetros que aprende durante el entrenamiento.

Parámetros:

model	Arquitectura de PyTorch que se optimizará.
name	Cadena de texto que indica el optimizador que se quiere.
learning_rate	Tasa de aprendizaje.
momentum	Momentum, sólo utilizado en "SGD".

Devuelve:

Optimizador de PyTorch requerido.

def modelLoader.loadModel (*path*, *model*, *optimizer* = `None`)

Función que carga el modelo desde el directorio indicado con la función 'load()' de PyTorch.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Con la función 'load_state_dict()' del modelo de PyTorch se re-establecen los parámetros entrenados del modelo. Si se quiere cargar el optimizador para continuar el entrenamiento, se hace con la función 'load_state_dict()' del optimizador de PyTorch.

Parámetros:

path	Directorio del modelo a cargar.
model	Arquitectura de PyTorch del modelo a cargar.
optimizer	Optimizador del cual se quiere reestablecer el estado previo al guardado. Si es 'None', no se carga.

Devuelve:

Devuelve una tupla que contiene el número de época en que se cargó, el modelo con los parámetros cargados y el optimizador con el estado re-establecido.

def modelLoader.saveModel (***epoch*****,** ***model*****,** ***optimizer*****,** ***is_best*** **=** `False`**,** ***path*** **=** `"./model/"`**)**

Función que guarda el modelo en el directorio indicado con la función 'save()' de PyTorch.

Los parámetros entrenados se guardan en un diccionario con la clave 'state_dict' y el estado del optimizador como 'optim_dict', a fin de poder reanudar el entrenamiento. Además, almacena la época en que se hace con la clave 'epoch'.

Parámetros:

epoch	Entero que indica la época en que se guarda el modelo.
model	Modelo de PyTorch al cual se quieren almacenar los parámetros entrenados.
optimizer	Optimizador que se utilizó en el entrenamiento y se quiere almacenar el estado.
is_best	Booleano que indica si se está guardando el modelo con menor pérdida o no, para guardarlos diferenciadamente. En el primer caso, le añade "_best" al nombre y en el segundo "_final".
path	Directorio en que se almacena.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del archivo parameters.py

Script que define los parámetros que se utilizan para el entrenamiento de la red convolucional.

Descripción detallada

Script que define los parámetros que se utilizan para el entrenamiento de la red convolucional.

En la siguiente tabla se detallan los parámetros, indicando los que trae por defecto que fueron utilizados para entrenar la red W.

Parámetro	Detalle	Valor por defecto
num_epochs	Cantidad de épocas que se entrena.	30
batch_size	Cantidad de datos para pasar por la red simultáneamente.	10
learning_rate	Tasa de aprendizaje.	0,001
momentum	Momentum (sólo aplicado para función de pérdida SGD).	0,9
nombreArq	Nombre de la arquitectura de red (sin '.py').	"RedW"
tipoOptim	Optimizador usado para el entrenamiento.	"Adam"
tipoLoss	Función de pérdida usada para el entrenamiento.	"CrossEntropy"
cant_train	Cantidad de archivos de entrenamiento usados.	12
val_split	Porcentaje de datos reservados para la validación.	0,15
nameData	Texto que se agrega al nombre de los archivos de salida	""
pathAnterior	Directorio de los parámetros entrenados del modelo, (útil para reanudar un entrenamiento). Si es vacío, no se carga.	""
nameModel	Similar a 'nameData' para los parámetros entrenados.	""
DATA_PATH	Carpeta en que se encuentran los datos	"./data/"
MODEL_STORE_PATH	Directorio en que se guardan los parámetros del modelo entrenado.	"./modelo/"+tipoOptim+ "_"+tipoLoss + "_"+ nameModel
OUT_PATH	Directorio en que se guardan los archivos de salida.	"./salida/"+nameData
PREDICT_PATH	Directorio en que se guardan las predicciones como imágenes.	"./salida/"+"predicciones/" + nameData
REALES_PATH	Path para las predicciones de la red sobre datos reales.	"./inferencias/"+ nameData
verGraficas	Booleano que indica si se quieren ver las gráficas online.	False
cadaCuantas	Entero que indica cada cuantas épocas guardar las gráficas.	1
cadaCuantos	Indica cada cuantos batches guardar las predicciones.	300
cadaCuantosVal	Similar al anterior para validación.	60
cadaCuantosTest	Similar al anterior para test.	20

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete RedW

Paquete que incluye la clase '**RedW**' que define la arquitectura de la red W, similar a dos red U conectadas.

Clases

class **RedW**

Descripción detallada

Paquete que incluye la clase '**RedW**' que define la arquitectura de la red W, similar a dos red U conectadas.

Dicha red U se determina en 'OverlapSegmentationNet.py'. Se hereda de la clase 'nn.Module' de PyTorch y se sobrecarga la función que se encarga de hacer el pasaje hacia adelante de la imagen '**forward()**'.

---

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del paquete resultados

Archivo en el que se definen las clases y funciones utilizadas para guardar los resultados parciales y finales que se obtienen.

Clases

class **graficas**

Funciones

def **inicializarLogger** (filename='log.txt')

Función que inicializa el logger utilizado para guardar cada paso en el entrenamiento.

def **log** (cadena, porConsola=True)

Función que guarda en el archivo de log la cadena que se le pasa.

def **medidas** (predicted, labels)

Función que calcula el recall y el Jaccard promedio de un batch de imágenes dados sus correspondientes ground truths.

Descripción detallada

Archivo en el que se definen las clases y funciones utilizadas para guardar los resultados parciales y finales que se obtienen.

Incluye el cálculo de las medidas de recall y Jaccard, así como también la graficación de ellos y la pérdida. Estas gráficas se realizan para el entrenamiento y para la validación mediante la librería 'matplotlib'. También incluye la muestra online de predicciones y el almacenamiento de ellas.

Documentación de las funciones

def resultados.inicializarLogger (*filename* = `'log.txt'`)

Función que inicializa el logger utilizado para guardar cada paso en el entrenamiento.

Utiliza la librería 'logging' que trae por defecto Python.

Parámetros:

filename	Nombre del archivo en el que se guarda.

def resultados.log (*cadena*, *porConsola* = `True`)

Función que guarda en el archivo de log la cadena que se le pasa.

Parámetros:

cadena	Cadena de texto que se guarda en el archivo de log.
porConsola	Booleano que indica si la cadena también se muestra por consola.

def resultados.medidas (*predicted*, *labels*)

Función que calcula el recall y el Jaccard promedio de un batch de imágenes dados sus correspondientes ground truths.

Para ello, primero calcula la clase de cada píxel mediante la función 'argmax()' de NumPy. A continuación, calcula la matriz de confusión mediante la función 'confusion_matrix()'. Luego, recorre cada fila (correspondiente a cada clase) y, si hay algo en ella, hace el cálculo de las medidas y las añade a una lista. Si son todos ceros significa que esa clase existe pero nunca fue acertada y traería problemas de división por cero. En ese caso, directamente se añade un recall y Jaccard de valor 0. Se calcula el recall como el valor de la diagonal dividido la suma de la fila [TP / (TP+FN)]. El Jaccard es el valor de la diagonal dividido la suma de la fila y de la columna

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

[TP / (TP+FN+FP)]. Entre corchetes se indica la fórmula, significando TP verdaderos positivos, FN falsos negativos y FP falsos positivos. Por último, se promedia el recall y el Jaccard y se devuelve como tupla.

Parámetros:

predicted	Arreglo 4D de NumPy. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales (clases) y las restantes al tamaño de la imagen.
labels	Arreglo de NumPy 3D que contiene el número de clase de cada píxel. La primera dimensión es la cantidad de datos y las restantes el tamaño de la imagen. El número indica la clase correcta de cada clase o equivalentemente el canal de 'predicted' correcto.

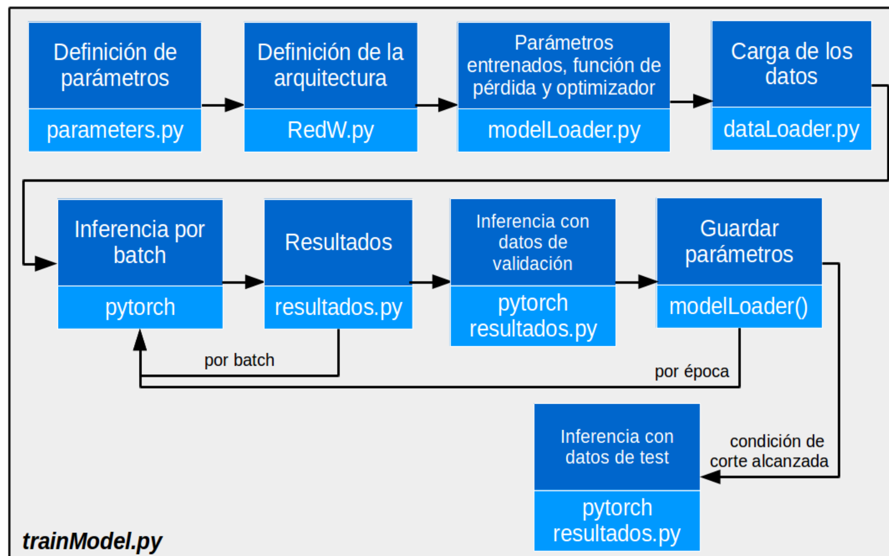
Devuelve:

Tupla conteniendo los valores promedio de recall y Jaccard.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia del archivo trainModel

Script que itera sobre los datos de entrenamiento para entrenar la red convolucional. En la siguiente imagen se ve un diagrama de bloques de cómo funciona el mismo.



Los parámetros se cargan desde el archivo 'parameters.py' en que se definen. La arquitectura de la red convolucional utilizada se determina en 'RedW.py' mediante la librería PyTorch. Luego, se utilizan las funciones definidas en 'modelLoader.py' para obtener la función de pérdida 'getLossFunc()', obtener el optimizador 'getOptimizer()' y para cargar parámetros ya entrenados para el modelo (si los hubiera) 'loadModel()'. También se inicializa el logger en el que se guardará datos referidos a cada batch y la clase graficas definida en 'resultados.py'.

Se cargan los datos de train mediante la clase 'genDataset()' definida en 'dataLoader.py'. Luego, con la función 'splitAndLoader()' también definida en 'dataLoader.py', se divide el dataset en una partición de entrenamiento y otra de validación. Ambas se transforman ahí mismo en un DataLoader definido por PyTorch, que permite iterar fácilmente por batch sobre cada partición. Un procedimiento similar se realiza con las imágenes de cromosomas reales (sin particionar).

A continuación, se define un bucle en el que se recorren los datos de train de a batches, se envían a la GPU si estuviera disponible mediante 'to()', se infiere sobre las imágenes del mismo con la función 'forward()' mencionada anteriormente, se actualizan los parámetros según la pérdida calculada con la función 'backward()' y se guardan los resultados obtenidos mediante la clase graficas definida en 'resultados.py'. Esta última posee una función 'cambiarPredicted()' utilizada para guardar predicciones que realiza la red. También se acumulan en listas las medidas de recall y coeficiente de Jaccard obtenidos mediante la función 'medidas()' definida en 'resultados.py'. Luego, se acumula en otras listas las medidas anteriores promediadas por época y mediante la función 'graficar()' de 'graficas()' se genera una figura que muestra gráficas del progreso de las medidas mencionadas en entrenamiento y validación durante las épocas. Sumado a lo anterior, con la función 'log()' definida en 'resultados.py' se actualiza en OUT PATH un archivo 'log.txt' con las medidas en cada batch. Antes de proseguir con la siguiente iteración, se guardan los parámetros entrenados si es que la pérdida promedio en validación es menor a la mejor pérdida promedio obtenida hasta el momento. Esto se hace mediante la función 'saveModel()' de 'modelLoader.py'.

Luego, se repite el proceso desde el recorrido de los datos de entrenamiento hasta alcanzar la cantidad de épocas límite definida en 'parameters.py', que es la única condición de corte del entrenamiento. Una vez finalizado, se realiza un procedimiento similar con las imágenes de test.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Documentación de las clases

Referencia de la Clase dataLoader.genDataset

Métodos públicos

def **__init__** (self, paraQue, hayLabels=True, cant_train=1, path="./data/")
  Constructor de la clase que lee los datos del disco.

def **__getitem__** (self, index)
  Sobrecarga del método homólogo de la clase 'Dataset' de Pytorch.

def **__len__** (self)
  Sobrecarga del método homólogo de la clase 'Dataset' de Pytorch.

Documentación del constructor

def dataLoader.genDataset.__init__ (***self*****,** ***paraQue*****,** ***hayLabels*** **=** `True`**,** ***cant_train*** **=** `1`**,** ***path*** **=** `"./data/"`**)**

Constructor de la clase que lee los datos del disco.

Si los datos fueran para entrenar (para "train") carga en 'data' todos los archivos con forma "trainN.npz", donde N es un número del 1 a 'cant_train'. De lo contrario, simplemente carga el archivo indicado. También se le pasa como parámetro si los datos tienen ground truth o no, para que los cargue o no en 'labels'. Además, define la transformación que se le realizarán a los datos para normalizarlos con media 0.5 y desvío 0.5 mediante la librería 'torchvision'.

Parámetros:

paraQue	String que indica si es para "train" o en otro caso es el nombre del archivo (sin '.npz').
hayLabels	Booleano que indica si los datos tienen ground truth o no.
cant_train	Entero que indica la cantidad de archivos de entrenamiento en el caso de "train".
path	Carpeta en la que se encuentran los mencionados archivos.

Documentación de las funciones miembro

def dataLoader.genDataset.__getitem__ (***self*****,** ***index*****)**

Sobrecarga del método homólogo de la clase 'Dataset' de Pytorch.

Define cómo se obtiene una imagen y su ground truth (si hubiera) del dataset. Además, los transforma a Tensor de PyTorch y aplica la transformación definida en '**__init__** ()'.

Parámetros:

index	Índice del dato a obtener.

Devuelve:

Imagen y ground truth (si hubiera) como Tensor de PyTorch de 3 y 4 dimensiones.

def dataLoader.genDataset.__len__ (***self*****)**

Define cómo se obtiene la longitud del dataset.

La documentación para esta clase fue generada a partir del siguiente fichero:

0 dataLoader.py

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia de la Clase resultados.graficas

Métodos públicos

def **__init__** (self, online=True, size=(15, 8))
Constructor de la clase que inicializa la figura con las subfiguras en las que se graficará.

def **cambiarPredicted** (self, epoch, maskPredicted1, maskPredicted2, path="./", name="", real=False)
Actualiza las predicciones en la figura si 'online' es 'True', sino sólo las almacena como archivo.

def **graficar** (self, epoch, loss, acc, jacc, loss_val, acc_val, jacc_val, path="./", guardar=True, ultimo=False, name="")
Función que actualiza las gráficas de pérdida, recall y Jaccard de entrenamiento y de validación.

Documentación del constructor y destructor

def resultados.graficas.__init__ (*self*, *online* = `True`, *size* = `(15,8)`)

Constructor de la clase que inicializa la figura con las subfiguras en que se graficará.

Si 'online' es falso, las subfiguras son 6. De izquierda a derecha corresponden a la pérdida, recall y Jaccard promedio por época. Las tres superiores son de entrenamiento y las tres inferiores de validación. Si 'online' es verdadero, las subfiguras son 9 ya que se muestran también predicciones parciales de la red.

Parámetros:

online	Booleano que indica si las gráficas se muestran mientras se entrena o solamente se guardan en un archivo por época.
size	Tamaño de la figura.

Documentación de las funciones miembro

def resultados.graficas.cambiarPredicted (*self*, *epoch*, *maskPredicted1*, *maskPredicted2*, *path* = `"./"`, *name* = `""`, *real* = `False`)

Actualiza las predicciones en la figura si 'online' es 'True', sino sólo las almacena como archivo.

Parámetros:

epoch	Número de época utilizado para guardar las predicciones.
maskPredicted1	Arreglo 2D de NumPy correspondiente a la imagen que se quiere mostrar y guardar. Generalmente es la predicción de la red.
maskPredicted2	Arreglo 2D de NumPy correspondiente a la imagen que se quiere mostrar y guardar. Generalmente es el ground truth o la imagen de la que se predijo.
path	Directorio en que se guardan 'maskPredicted1' y 'maskPredicted2'.
name	Cadena de texto extra que se agrega al nombre de la magen al guardarla.
real	Booleano que si es verdadero guarda 'maskPredicted2' en escala de grises por corresponder a cromosomas, sino se guarda en colores por corresponder a una máscara que indica las clases.

def resultados.graficas.graficar (*self*, *epoch*, *loss*, *acc*, *jacc*, *loss_val*, *acc_val*, *jacc_val*, *path* = `"./"`, *guardar* = `True`, *ultimo* = `False`, *name* = `""`)

Función que actualiza las gráficas de pérdida, recall y Jaccard de entrenamiento y de validación.

Además, si se indica, se guardan las medidas obtenidas en un archivo '.npz' con claves iguales a los nombres de los parámetros.

Parámetros:

epoch	Número de época utilizado para guardar las predicciones.
loss	Lista que contiene la pérdida de entrenamiento en cada época.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

acc	Lista que contiene el recall de entrenamiento en cada época.
jacc	Lista que contiene el Jaccard de entrenamiento en cada época.
loss_val	Lista que contiene la pérdida de validación en cada época.
acc_val	Lista que contiene el recall de validación en cada época.
jacc_val	Lista que contiene el Jaccard de validación en cada época.
guardar	Booleano que indica si se guardan las medidas en un archivo '.npz' o no.
ultimo	Booleano que indica si es la última gráfica, entonces espera que se aprete un botón para cerrarse.
name	Cadena de texto extra que se agrega al nombre de la magen al guardarla.

La documentación para esta clase fue generada a partir del siguiente fichero:

1 resultados.py

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia de la Clase OverlapSegmentationNet.OverlapSegmentationNet

Métodos públicos

def **__init__** (self, canalesEntrada=1)

Constructor que define la estructura de la red convolucional.

def **forward** (self, x)

Sobrecarga de la función homóloga de PyTorch que realiza la pasada hacia adelante de la red.

Atributos públicos

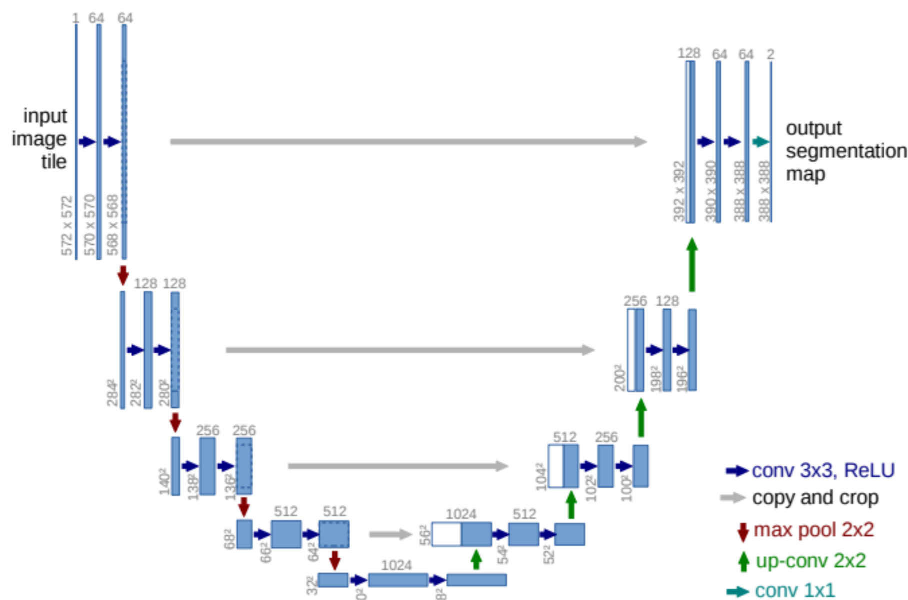
canalesEntrada: determina cuantos canales de entrada se definirán en la primera capa del modelo

---

Documentación del constructor y destructor

def OverlapSegmentationNet.OverlapSegmentationNet.__init__ (*self*, *canalesEntrada* = 1)

Constructor que define la estructura de la red convolucional. Ésta es similar a la red U de la siguiente imagen.



Documentación de las funciones miembro

def OverlapSegmentationNet.OverlapSegmentationNet.forward (*self*, *x*)

Sobrecarga de la función homóloga de PyTorch que realiza la pasada hacia adelante de la red.

Se encarga de determinar cómo se relacionan entre sí las capas definidas en el constructor.

Parámetros:

x	Entradas en formato Tensor de 4 dimensiones de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales y las últimas dos al tamaño de la imagen.

Devuelve:

Salida de la red convolucional en formato Tensor 4D de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales (24 clases) y las últimas dos al tamaño de la imagen.

La documentación para esta clase fue generada a partir del siguiente fichero:

2 OverlapSegmentationNet.py

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Referencia de la Clase RedW.RedW

Métodos públicos

def **__init__** (self)

Constructor que define la estructura de la red convolucional.

Documentación del constructor y destructor

def RedW.RedW.__init__ (*self*)

Constructor que define la estructura de la red convolucional. La misma consta de dos redes U definidas en ‘OverlapSegmentationNet.py’, en la que se conecta la salida de una a la entrada de la otra.

Documentación de las funciones miembro

def RedW.RedW.forward (*self*, *x*)

Sobrecarga de la función homóloga de PyTorch que realiza la pasada hacia adelante por la red convolucional.

Se encarga de determinar cómo se interrelacionan entre sí las capas definidas anteriormente en el constructor.

Parámetros:

x	Entradas en formato Tensor de 4 dimensiones de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales y las últimas dos al tamaño de la imagen.

Devuelve:

Salida de la red convolucional en formato Tensor de 4 dimensiones de PyTorch. La primera corresponde a la cantidad de datos, la segunda a la cantidad de canales (24 por la cantidad de clases) y las últimas dos al tamaño de la imagen.

La documentación para esta clase fue generada a partir del siguiente fichero:

3 RedW.py

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Apéndice D

Anteproyecto

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas

PROPUESTA DE PROYECTO FINAL DE CARRERA
INGENIERÍA INFORMÁTICA

Diseño y desarrollo de una herramienta de segmentación automática de cromosomas

Alumno: Fenoglio, Sebastián

Director: Dr. Martínez, César

Co-Director: Dr. Gerard, Matías

Santa Fe, Julio de 2018

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Índice

1. Resumen **3**

2. Justificación **3**

3. Objetivos **6**
- 3.1. Objetivo general 6
- 3.2. Objetivos específicos 6

4. Alcance **6**
- 4.1. Requerimientos funcionales 6
- 4.2. Requerimientos no funcionales 6
- 4.3. Restricciones 7
- 4.4. Supuestos 7

5. Metodología **7**

6. Plan de tareas **7**

7. Cronograma **9**
- 7.1. Hitos 9

8. Entregables y puntos de control **10**
- 8.1. Entregables 10
- 8.2. Puntos de control 11

9. Riesgos **11**

10. Recursos y presupuesto **12**

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

1. Resumen

La citogenética es la parte de la genética que se encarga del estudio de la estructura y de la función de los cromosomas. Estos estudios son un importante procedimiento de diagnóstico médico tanto para la detección como para el tratamiento de distintas enfermedades y discapacidades.

Para realizar un análisis sobre los cromosomas de un ser humano, un paso fundamental es la segmentación de los mismos a partir de una imagen microscópica obtenida de las células del ser humano mencionado. Esta tarea reviste una gran complejidad por la estructura no rígida de los cromosomas y por los solapamientos que puede haber entre ellos.

En este proyecto se propone el desarrollo de una herramienta computacional de código abierto que segmente los cromosomas de una imagen microscópica de forma totalmente automática y dé como resultado imágenes de dichos cromosomas separados, dado que en la actualidad no existe una herramienta de código abierto que lo haga.

Palabras clave: citogenética, diagnóstico médico, cariograma, segmentación automática, procesamiento de imágenes.

2. Justificación

La citogenética es la parte de la genética que se encarga del estudio de la estructura y de la función de los cromosomas. La información del genoma humano está almacenada típicamente en 23 pares de cromosomas, de los cuales 22 son homólogos y se denominan autosómicos, mientras que el restante se denomina par sexual. Este par es responsable de determinar los rasgos masculinos y femeninos de los individuos [1]. Así, se da lugar a 24 tipos de cromosomas humanos: los 22 homólogos y los 2 del par sexual.

Los estudios sobre cromosomas son un importante procedimiento de diagnóstico médico en dictámenes prenatales, en pacientes con retrasos mentales y múltiples problemas de nacimiento, en pacientes con anormal desarrollo sexual y en algunos casos de infertilidad o múltiples abortos espontáneos. También es útil para el estudio y tratamiento de pacientes con neoplasias malignas y desórdenes hematológicos [1].

Una representación muy utilizada para el análisis de los cromosomas es el cariograma [1] [2]. Para realizarlo, primero se extrae una muestra de células del ser humano bajo estudio, se le aplica alguna técnica de tinción para realzar los cromosomas y luego se obtiene una imagen microscópica de ellos generalmente en metafase [1]. A partir de ésta, para obtener el cariograma debe pre-procesarse la imagen, segmentarse los cromosomas y luego clasifi-

[1]Fase del ciclo celular en la que pueden apreciarse los patrones que forman las bandas de los cromosomas según la tinción utilizada.

carse en los 24 tipos de cromosomas humanos existentes [3] [4]. Un ejemplo puede verse en la Figura 1.

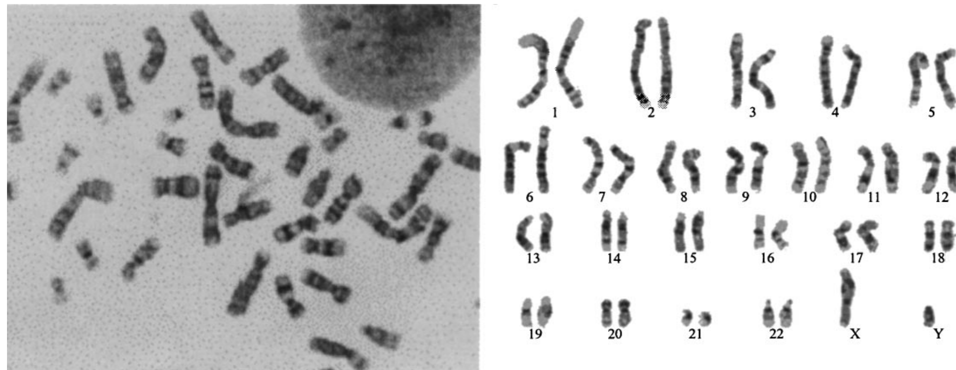


Figura 1: Cromosomas de un ser humano con tinción en banda G y su correspondiente cariograma [5].

En la Figura 1 se observan los cromosomas con tinción en banda G, la técnica más popular por su accesibilidad. Existen otras que logran distinguir cada tipo de cromosoma con distintos colores, como la llamada FISH, pero son más costosas y suelen utilizarse como información complementaria a la tinción en banda G [2].

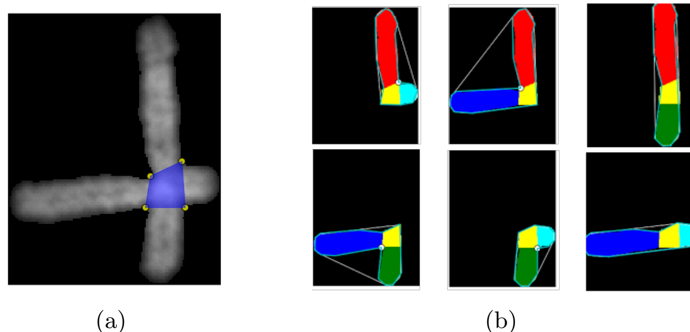
Un paso fundamental en la obtención del cariograma es la segmentación de los cromosomas a partir de una imagen obtenida de la observación microscópica de las células extraídas del ser humano bajo estudio. En la misma, dos o más cromosomas pueden estar solapados y generar ambigüedades sobre cómo separarlos. La Figura 2(a) presenta un ejemplo típico de ésta situación. Como puede apreciarse en la figura 2(b), existen tres formas diferentes de segmentar estos cromosomas, y cada caso puede conllevar una interpretación diferente. El problema real se complica aún más al ser los cromosomas estructuras no rígidas y al poder solaparse más de dos de ellos.

Realizar esta segmentación manualmente consumiría tiempo y recursos evitables, sin tener en cuenta lo tedioso de la tarea, ya que se puede automatizar computacionalmente.

Existen productos comerciales que realizan todo el proceso de obtención del cariograma, pero tienen algunas desventajas. Primero, se venden como paquetes cerrados [7] [8] [9], por lo que no se sabe cómo funcionan realmente, no pueden adaptarse a necesidades específicas ni mejorarse con el descubrimiento de nuevos métodos de segmentación. Algunos de ellos funcionan sólo con determinado equipamiento [8] o están atados a las técnicas de tinción más costosas [8] [9], limitando su aplicabilidad. También hay software de código abierto pero son semi-automáticos [10] [11], es decir, requieren la intervención humana en algún punto.

La segmentación automática de cromosomas es un tema actual en inves-

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.



(a) (b)

Figura 2: (a) Dos cromosomas con la zona de solapamiento marcada en azul. (b) Por columna se ve cada una de las tres segmentaciones posibles con la zona de solapamiento en amarillo y las diferentes partes de cada cromosoma en verde, celeste, rojo y azul. [6].

tigación, en el que se están analizando diferentes enfoques para su resolución, principalmente basados en métodos geométricos [3] [4], en aprendizaje maquinal [12] o en combinaciones de ambos [6].

Dado que en la actualidad no existe una herramienta de código abierto que realice la segmentación automática de cromosomas, en este proyecto se propone el desarrollo de una herramienta computacional de código abierto que, a partir de una imagen microscópica de cromosomas humanos con tinción en banda G, los segmente de forma totalmente automática y dé como resultado imágenes de dichos cromosomas separados, aún cuando éstos estén originalmente solapados.

Un beneficiario directo es el instituto de investigación sinc(i) de la Facultad de Ingeniería y Ciencias Hídricas (FICH) de la Universidad Nacional del Litoral (UNL) y, en general, la comunidad científica ya que, como se mencionó, es un tema actual de investigación. Además, una herramienta completamente automática puede permitir escalar ciertos experimentos a un número mucho más grande de cromosomas, lo que no sería posible sin la automatización. Por ejemplo, el análisis cuantitativo del largo de los extremos del cromosoma.

El hecho que sea de código abierto permite fundamentalmente dos cosas. Primero, una mejora continua y distribuida de la herramienta debido a que cualquier persona con los conocimientos suficientes puede inspeccionarla y optimizarla conforme avance la investigación sobre la segmentación de cromosomas. Segundo, la adaptación para aplicaciones específicas y la integración en otras herramientas más completas. Por ejemplo, podría combinarse con alguna herramienta de clasificación de cromosomas ya existente [13] y obtener una herramienta que realice automáticamente el cariograma, para que el experto en citogenética pueda analizar los cromosomas y detectar

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

anormalidades si las hubiera.

En consecuencia, la sociedad en general también se verá beneficiada ya que, al utilizar la tinción en banda G, haría que el estudio de los cromosomas sea un instrumento de diagnóstico accesible para la prevención y tratamiento de las enfermedades ya nombradas.

3. Objetivos

3.1. Objetivo general

Diseñar y desarrollar una herramienta que permita la segmentación automática de cromosomas a partir de una imagen microscópica.

3.2. Objetivos específicos

- Analizar y comparar el desempeño de las técnicas de segmentación de cromosomas en metafase disponibles en la literatura.
- Definir el algoritmo de segmentación y el conjunto de parámetros configurables del mismo.
- Desarrollar un prototipo funcional y bien documentado de la aplicación.
- Analizar el desempeño del algoritmo con medidas objetivas empleando corpus de imágenes de libre disponibilidad.
- Obtener una interfaz que permita la abstracción de las funciones y opciones de la herramienta por sobre su desarrollo algorítmico.

4. Alcance

4.1. Requerimientos funcionales

- Pre-procesar las imágenes para facilitar su posterior segmentación.
- Segmentar y separar los cromosomas.
- Post-procesar los cromosomas para facilitar su posterior análisis o clasificación.

4.2. Requerimientos no funcionales

- Robustez frente a distintos tipos de imágenes de entrada.
- Se utilizará como lenguaje de programación Python ya que éste prioriza la legibilidad y facilitaría la reutilización de la herramienta.

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

4.3. Restricciones

- Se desarrollará una interfaz sólo por línea de comandos.
- Se utilizarán imágenes de cromosomas humanos con tinción en banda G.

4.4. Supuestos

- Se dispondrá acceso al clúster de procesamiento del sinc(i), FICH UNL.
- Existen corpus de imágenes con las características ya mencionadas disponible para el desarrollo.

5. Metodología

Se propone la utilización del modelo en cascada o ciclo de vida clásico debido a que en este proyecto se conocen previamente los requerimientos y funcionalidades de la herramienta a desarrollar. Pero como el producto final puede dividirse en distintos módulos, se propone que cada etapa tenga su propio ciclo de vida en cascada para el desarrollo de cada uno de ellos. Es decir, se utilizará un modelo de cascada por subproyectos.

La primera etapa tendrá como objetivo recopilar antedecedentes en la segmentación automática de cromosomas y preparar los recursos necesarios para el desarrollo del proyecto, tales como la obtención de corpus de imágenes libres y la instalación de las librerías necesarias.

Luego, las tres siguientes etapas son los tres módulos a desarrollar. Idealmente podrían realizarse en forma paralela pero al contar con un único recurso humano se realizarán secuencialmente. El primero es el pre-procesamiento y segmentación, que consiste en disminuir los ruidos e interferencias de la imagen y extraer los cromosomas de la misma. El segundo es la separación de los cromosomas solapados extraídos previamente, módulo al cual se le dedicará mayor tiempo debido a su mayor complejidad. Mientras que el tercero es el encargado del post-procesamiento de los cromosomas para simplificar su análisis o clasificación, con procedimientos tales como el enderezamiento de los mismos.

La quinta fase consiste en integrar los módulos descriptos y desarrollar la interfaz que permitirá la abstracción sobre la complejidad de los algoritmos. Además, en este punto se obtiene la documentación final del desarrollo que se irá realizando fase a fase.

6. Plan de tareas

A continuación se muestran las actividades correspondiente a cada etapa y el esfuerzo requerido para cada una expresada en horas/hombre.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

1.	**Inicialización**	
1.1	Estudiar antecedentes en la segmentación de cromosomas disponibles en la literatura.	20 hs
1.2	Recopilar corpus de imágenes disponibles.	12 hs
1.3	Recopilar, elegir e instalar de librerías necesarias para el desarrollo.	12 hs
2.	**Pre-procesamiento y segmentación**	
2.1	Revisar bibliografía sobre métodos de pre-procesamiento y segmentación.	16 hs
2.2	Proponer y diseñar la estrategia de pre-procesamiento y segmentación.	16 hs
2.3	Desarrollar y depurar lo diseñado.	20 hs
2.4	Realizar pruebas para validar la estrategia.	16 hs
2.5	Documentar lo desarrollado en la etapa.	12 hs
2.6	Redactar informe de avance.	20 hs
3.	**Separación de solapamientos**	
3.1	Estudiar bibliografía sobre métodos geométricos utilizados.	20 hs
3.2	Estudiar bibliografía sobre redes convolucionales profundas.	20 hs
3.3	Proponer y diseñar la estrategia de separación de cromosomas solapados.	20 hs
3.4	Desarrollar y depurar lo diseñado.	32 hs
3.5	Realizar pruebas para validar la estrategia propuesta.	
	3.5.1 Definir pruebas.	8 hs
	3.5.2 Ejecutar las pruebas definidas.	28 hs
	3.5.3 Analizar los resultados.	20 hs
3.6	Documentar lo desarrollado en la etapa.	20 hs
3.7	Redactar informe de avance.	20 hs
4.	**Post-procesamiento**	
4.1	Estudiar bibliografía sobre métodos post-procesamiento.	16 hs
4.2	Proponer y diseñar la estrategia de post-procesamiento.	16 hs
4.3	Desarrollar y depurar lo diseñado.	20 hs
4.4	Realizar pruebas para validar la estrategia propuesta.	16 hs
4.5	Documentar lo desarrollado en la etapa.	12 hs
4.6	Redactar informe de avance.	20 hs
5.	**Integración e interfaz**	
5.1	Integrar los módulos y realizar pruebas de integración.	20 hs
5.2	Definir opciones configurables y diseñar interfaz.	20 hs
5.3	Desarrollar, depurar y probar la interfaz.	28 hs
5.4	Redactar informe final.	28 hs
	Total	**528 hs**

7. Cronograma

Empleando una dedicación diaria de 4 horas de lunes a sábado e iniciando el proyecto el 10/07/18, el mismo se finaliza el 11/12/18. En la Figura 3 puede verse el cronograma completo.

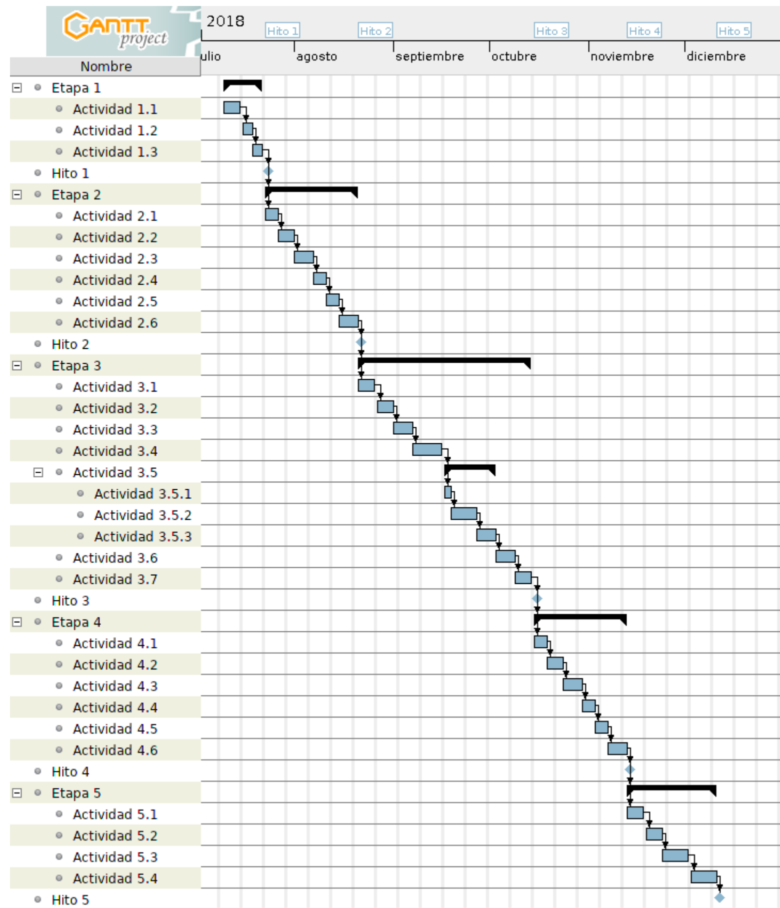


Figura 3: Cronograma propuesto.

7.1. Hitos

Se listan los hitos marcados en el cronograma de la Figura 3.

- Hito 1: 23/07/18. En este punto se alcanza conocimiento sobre el estado el arte del problema y se tienen definidas las librerías que se utilizarán en el desarrollo.
- Hito 2: 21/08/18. Se completó el desarrollo del módulo encargado del pre-procesamiento y de la segmentación y se verificó su correcto funcionamiento.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

- Hito 3: 15/10/18. Se finalizó el desarrollo del módulo encargado de la separación de los cromosomas solapados y se midió su rendimiento.
- Hito 4: 13/11/18. Se completó el desarrollo del módulo encargado del post-procesamiento y se verificó su correcto funcionamiento.
- Hito 5: 11/12/18. Finalización del proyecto. Se tiene el prototipo funcional de la herramienta con su funcionamiento verificado y su correspondiente documentación.

8. Entregables y puntos de control

8.1. Entregables

A continuación se detallan los entregables del proyecto junto con sus criterios de aceptación.

Entregable 001: Recopilación y librerías
Se obtiene al finalizar la primera etapa. Informe en el que se detalla el estado del arte del problema de segmentación automática de cromosomas y se explican las decisiones tomadas en cuanto a librerías que se usarán.
Criterio de aceptación: Aprobación de los directores.

Entregable 002: Código de pre-procesamiento y segmentación
Se obtiene al finalizar la segunda etapa. Es la codificación de las funciones que permitirán reducir los ruidos de la imagen y extraer los cromosomas de la misma.
Criterio de aceptación: Cumpla las funcionalidades requeridas con un desempeño que cuente con la aprobación de los directores.

Entregable 003: Código de separación de cromosomas
Se obtiene al finalizar la tercera etapa. Es la codificación de las funciones que logran dividir los cromosomas superpuestos.
Criterio de aceptación: Cumpla las funcionalidades requeridas con un desempeño que cuente con la aprobación de los directores.

Entregable 004: Código de post-procesamiento
Se obtiene al finalizar la cuarta etapa. Es la codificación de las funciones que facilitan la posterior clasificación de los cromosomas.
Criterio de aceptación: Cumpla las funcionalidades requeridas con un desempeño que cuente con la aprobación de los directores.

Entregable 005: Herramienta
Se obtiene al finalizar la quinta etapa. Es la herramienta de segmentación automática de cromosomas completamente desarrollada.
Criterio de aceptación: Cumpla las funcionalidades requeridas con un desempeño que cuente con la aprobación de los directores.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Entregable 006: Documentación
Se obtiene al finalizar la quinta etapa. Es la documentación de la herramienta totalmente desarrollada.
Criterio de aceptación: Se detalle correctamente cómo se codificó cada funcionalidad.

8.2. Puntos de control

Se definen los siguientes tres puntos de control, en los que se entregará un informe de avance.

- 21/08/18: Al finalizar la segunda etapa, se entregará junto al informe de avance los entregables Recopilación y librerías y Código de preprocesamiento y segmentación.
- 15/10/18: Al finalizar la tercera etapa, se adjuntará junto al informe de avance el entregable Código de separación de cromosomas.
- 13/11/18: Al finalizar la cuarta etapa, se adjuntará junto al informe de avance el entregable Código de post-procesamiento.

9. Riesgos

Se detallan los riesgos detectados en la ejecución del proyecto, junto con la estrategia de respuesta y de contingencia.

Riesgo 001: No disponibilidad recursos humanos	
Indicador: Viajes imprevistos, problemas de salud o intercambios estudiantiles.	
Probabilidad: Baja	Impacto: Medio
Estrategia: Aceptar.	
Contingencia: En caso de que el recurso no disponible sea el alumno, se modificará el cronograma. Si fuese alguno de los directores por no estar en la ciudad, se buscará contactar con ellos mediante videollamada. En el caso extremo, también se modificará el cronograma.	
Riesgo 002: Falla de computadora	
Indicador: Funcionamiento incorrecto de la computadora.	
Probabilidad: Baja	Impacto: Alto
Estrategia: Mitigar. Se realizarán back-ups mediante herramientas automáticas en la nube para evitar la pérdida de información.	
Contingencia: En caso de que la computadora falle, se recurrirá a otra de menor rendimiento que ya tiene la información sincronizada con la herramienta anterior.	

sinc(i) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

Riesgo 003: No disponibilidad del clúster del sinc(i)	
Indicador: Problemas técnicos con el clúster o que no haya tiempo disponible de procesamiento.	
Probabilidad: Media	Impacto: Medio
Estrategia: Mitigar. Se dispone de un chip con gran capacidad de procesamiento perteneciente a uno de los directores.	
Contingencia: En caso de no disponibilidad del clúster ni del chip, se tienen en cuenta herramientas web como Google Colab que ofrecen una limitada cantidad de tiempo de procesamiento hasta que se pueda disponer de alguno de los dos.	

Riesgo 004: No disponibilidad de corpus de imágenes	
Indicador: Corpus de imágenes con características inadecuadas para el proyecto o poca cantidad de imágenes.	
Probabilidad: Baja	Impacto: Alto
Estrategia: Mitigar. Pedir con anticipación a varios sitios web que requieren un proceso de autenticación como alumno o investigador para el envío de las imágenes.	
Contingencia: Estudiar la forma de adecuar las imágenes. En caso de que fueran pocas, analizar técnicas de aumento de datos.	

Riesgo 005: Conflictos o falta de funcionalidad de librerías	
Indicador: Carencia de funcionalidades de alguna librería. Conflicto en la integración de librerías.	
Probabilidad: Baja	Impacto: Medio
Estrategia: Mitigar. Utilizar siempre la misma versión de Python y verificar compatibilidades previamente.	
Contingencia: Según la magnitud del problema puede implicar cambiar la librería o codificar algunas funciones, ya sea para sustituir la funcionalidad propiamente dicha o para acoplar las funciones en conflicto.	

10. Recursos y presupuesto

Todos los recursos listados en el siguiente presupuesto son los necesarios para la ejecución del proyecto y ya están disponibles.

Se omiten del listado ciertos recursos sin costo como por ejemplo los utilizados para la bibliografía. Tanto la biblioteca Ezio Emiliani como gran cantidad de publicaciones en revistas científicas pueden accederse de forma gratuita en la FICH.

Para cada bien de capital se realiza la amortización correspondiente con la fórmula lineal:

$$A = \frac{VN - VR}{VU},$$

en la que A es la amortización, VN el valor a nuevo del bien, VR su valor residual y VU la vida útil del mismo.

Bienes de capital			
Descripción	Amortización	Tiempo	Total
Computadora	$128/mes	6 meses	$768
Notebook	$64/mes	6 meses	$384
Chip	$32/mes	6 meses	$192
Materiales e insumos			
Descripción	Precio unitario	Unidades	Total
Informes de avance	$100/unidad	3 unidades	$300
Preinforme	$400/unidad	3 unidades	$1200
Informe final	$1200/unidad	1 unidades	$1200
Artículos de librería	$100/mes	6 meses	$600
Servicios			
Descripción	Precio unitario	Unidades	Total
Tiempo de procesamiento en el clúster del sinc(i)	$22.5/hora	50 horas	$1125
Recursos humanos			
Descripción	Precio unitario [2]	Unidades	Total
Alumno	$642/hora	528 horas	$338976
Director	$1059/hora	50 horas	$ 52950
Co-director	$1059/hora	50 horas	$ 52950
Transporte y viáticos			
Descripción	Precio unitario	Unidades	Total
Transporte	$12,50/viaje	60 viajes	$750
Otros	$100/mes	6 meses	$600
Gastos varios			
Descripción	Precio unitario	Unidades	Total
Electricidad (Consumo: 0.383 KWh)	$2,098KWh	528 horas	$424
Internet	$900/mes	6 meses	$5400
		TOTAL	**$457819**

Referencias

[1] A. P. Britto and G. Ravindran, “A review of cytogenetics and its automation,” *J Med Sci*, vol. 7, pp. 1–18, 2007.

[2] W. K. Kwon, J. Y. Lee, Y. C. Mun, C. M. Seong, W. S. Chung, and J. Huh, “Clinical utility of fish analysis in addition to g-banded karyotype in hematologic malignancies and proposal of a practical approach,” *The Korean journal of hematology*, vol. 45, no. 3, pp. 171–176, 2010.

[2]Según Colegio de Profesionales de Ciencias Informáticas de Entre Ríos. Web: http://www.coprocier.org.ar/seccion_2.html (Consultada 13/06/2018)

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.

[3] T. Arora and R. Dhir, “A novel approach for segmentation of human metaphase chromosome images using region based active contours,” *International Arab Journal of Information Technology*, 2016.

[4] S. Minaee, M. Fotouhi, and B. H. Khalaj, “A geometric approach to fully automatic chromosome segmentation,” in *Signal Processing in Medicine and Biology Symposium (SPMB), 2014 IEEE*, pp. 1–6, IEEE, 2014.

[5] G. C. Charters and J. Graham, “Trainable grey-level models for disentangling overlapping chromosomes,” *Pattern Recognition*, vol. 32, no. 8, pp. 1335–1349, 1999.

[6] R. L. Hu, J. Karnowski, R. Fadely, and J.-P. Pommier, “Image segmentation to distinguish between overlapping human chromosomes,” *arXiv preprint arXiv:1712.07639*, 2017.

[7] A. S. Imaging, “Genasis karyotyping.” http://www.spectral-imaging.com/applications/cytogenetics/karyotyping. (Accedido el 17-05-2018).

[8] MetaSystems, “Rapidscore: Automation of fish spot counting.” https://metasystems-international.com/en/products/solutions/signal-analysis/. (Accedido el 17-05-2018).

[9] L. BioSystems, “Cytovision: Automated cytogenetics platform.” https://www.leicabiosystems.com/clinical-microscopy-surgery-radiology/cytogenetics/products/cytovision/. (Accedido el 17-05-2018).

[10] I. Belevich, M. Joensuu, D. Kumar, H. Vihinen, and E. Jokitalo, “Microscopy image browser: a platform for segmentation and analysis of multidimensional datasets,” *PLoS biology*, vol. 14, no. 1, p. e1002340, 2016.

[11] G. Mirzaghaderi and K. Marzangi, “Ideokar: an ideogram constructing and karyotype analyzing software,” *Caryologia*, vol. 68, no. 1, pp. 31–35, 2015.

[12] Y. Qiu, X. Lu, S. Yan, M. Tan, S. Cheng, S. Li, H. Liu, and B. Zheng, “Applying deep learning technology to automatically identify metaphase chromosomes using scanning microscopic images: an initial investigation,” in *Biophotonics and Immune Responses XI*, vol. 9709, p. 97090K, International Society for Optics and Photonics, 2016.

[13] sinc(i) FICH UNL, “Webdemo de cariotipado.” http://sinc.unl.edu.ar/web-demo/cariopynet/. (Accedido el 17-05-2018).

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
Sebastián Fenoglio, C. E. Martínez & M. Gerard; "Diseño y desarrollo de una herramienta de segmentación automática de cromosomas (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2019.