UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD IZTAPALAPA

DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERIA

POSGRADO EN CIENCIAS (INGENIERIA BIOMEDICA)

Segmentación 3D acelerada por GPU de IRM cerebral utilizando filtrado espacial y aprendizaje profundo

TESIS

PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS (INGENIERÍA BIOMÉDICA)

PRESENTA: Leonel Adan Rosas Castillo

MATRÍCULA: 2213800903

ORCID: 0009-0009-9188-5249

CORREO ELECTRÓNICO: leonelrosascas@gmail.com

DIRECTOR: MTRO. ÓSCAR YÁÑEZ SUÁREZ

JURADO:

PRESIDENTE: DR. JORGE LUIS PÉREZ GONZÁLEZ

SECRETARIO: DR. JUAN RAMÓN JIMÉNEZ ALANIZ

VOCAL: DR. EDUARDO BARBARÁ MORALES

IZTAPALAPA, CIUDAD DE MÉXICO, 16 DE DICIEMBRE DE 2024

Índice general

Índice de tablas

Índice de figuras

Prefacio

Resumen

Existe una gran cantidad de algoritmos para la segmentación de volúmenes médicos, en general están diseñados para propósitos específicos y no para una segmentación multiclase general. En el caso de la imagenología por resonancia magnética, no hay una excepción, porque se conocen métodos que hacen uso de redes neuronales convencionales, redes convolucionales o una combinación de ellas, pero en su mayoría se aplican a imágenes 2D que componen un volumen. Además, la mayoría de las investigaciones que exploran la segmentación directa de volúmenes 3D tienen como objetivo resolver la clasificación binaria (tumoral, por ejemplo).

Este trabajo propone un enfoque general para la segmentación 3D directa de volúmenes de resonancia magnética cerebral basada en datos, específicamente utilizando un sistema de filtrado espacial compuesto por el algoritmo del corrimiento de media potenciado con la participación de un mapa de confianza de bordes, para después acoplar el volumen filtrado a una red neuronal profunda, ambos métodos tomados del dominio del aprendizaje automático. El objetivo del filtrado espacial es preparar los volúmenes para la entrada a la red profunda, produciendo volúmenes más claros, limpios y contrastados entre clases para el sistema de aprendizaje profundo. Esto a su vez reduce el número de capas ocultas de la red, mejora el proceso de entrenamiento al producir curvas de aprendizaje más suaves (pérdida y precisión) y, finalmente, ofrece segmentaciones con un menor porcentaje de error de clasificación.

El sistema clasifica la materia gris, la materia blanca, el líquido cefalorraquídeo, el crá-

neo y el fondo, operando directamente en el espacio volumétrico 3D, esto con valores de precisión de cada prueba establecida en la validación cruzada que superan el 95 % y el modelo final presentado tiene una precisión del 97,56 % ($\pm$ 1,18 %) . Por otro lado, las curvas de aprendizaje reportadas muestran que la aplicación previa del filtrado, por corrimiento de media más el mapa de confianza de borde, a la segmentación de la red, produce volúmenes que facilitan el curso del entrenamiento de la red, con progresiones parsimoniosas y estables en los valores de error y precisión, época en época, en comparación a un sistema de las mismas características pero sin el uso del filtro espacial.
Además, dado que el filtrado espacial exige un alto tiempo de ejecución, se realizó una implementación en un entorno acelerado por GPU y a su vez, este recurso fue aprovechado para el entrenamiento y la evaluación de la red, lo que redujo de sobremanera los tiempos de procesamiento de todos los métodos.

Contenido

El Capítulo 1 introduce los temas relacionados y revisados en el proyecto, en esta compilación se exponen tópicos tales como la imagenología por resonancia magnética, la segmentación como parte del procesamiento digital de imágenes, el aprendizaje profundo, entre otros.

En el Capítulo 2 se expone de manera estructurada el sentido y la motivación de esta tesis, conteniendo el planteamiento del problema, la pregunta de investigación, hipótesis, objetivos y la justificación.

Los temas relacionados a este trabajo tienen un trasfondo importante para el desarrollo del mismo, por lo cual, en el Capítulo 3 se presentan hechos y logros de distintos trabajos de gran influencia para este proyecto.

El Capítulo 4 trata acerca de los métodos llevados a cabo, que van desde la adquisición de los volúmenes utilizados hasta la evaluación del segmentador mediante la validación cruzada.

En el Capítulo 5 se muestran los resultados obtenidos de los métodos planteados en el capítulo anterior.

Los resultados obtenidos son analizados en el Capítulo 6, se destaca que en este apartado se busca encontrar las consecuencias directas del filtrado espacial por medio del contraste de dos sistemas de segmentación iguales con excepción en que uno no hace uso del filtrado.

El Capítulo 7 cierra las ideas y nuevo conocimiento encontrado a lo largo del trabajo, sentando un nuevo antecedente en la segmentación de imágenes médicas y temas relacionados. Además, se establece como sugerencia a continuar, el trabajo a futuro con sentido del crecimiento de este proyecto.

Capítulo 1

Introducción

1.1. Imagenología por resonancia magnética

La imagenología por resonancia magnética es una técnica no invasiva que produce, en forma tridimensional, imágenes anatómicas y funcionales de la zona de estudio. El principio físico de esta técnica es la estimulación magnética y la subsecuente detección del cambio en la dirección del eje de rotación de protones de hidrógeno presentes en las moléculas de agua y grasa de los tejidos vivos. [1]

Las imágenes de esta técnica resultan del uso del escáner de resonancia magnética, mostrado en la Figura 1.1, compuesto de cuatro componentes principales:

1. Un imán que rodea al paciente, resistivo o superconductor de naturaleza, siendo más comúnes los imanes superconductores cilíndricos, consisten en bobinas enrolladas alrededor del orificio central, estas bobinas se enfrían cercanas al valor cero absoluto con helio líquido, lo que permite un flujo persistente de alta corriente eléctrica a través de las bobinas y creando el campo magnético necesario para la RM. Este campo es magnético B_0 es de gran intensidad con valores típicos de 1 a 7 Teslas.

2. Bobinas de gradiente, las cuales se encuentran en los planos x, y y z del imán. Éstas proporcionan un gradiente de campo magnético adicional en cada dirección y son

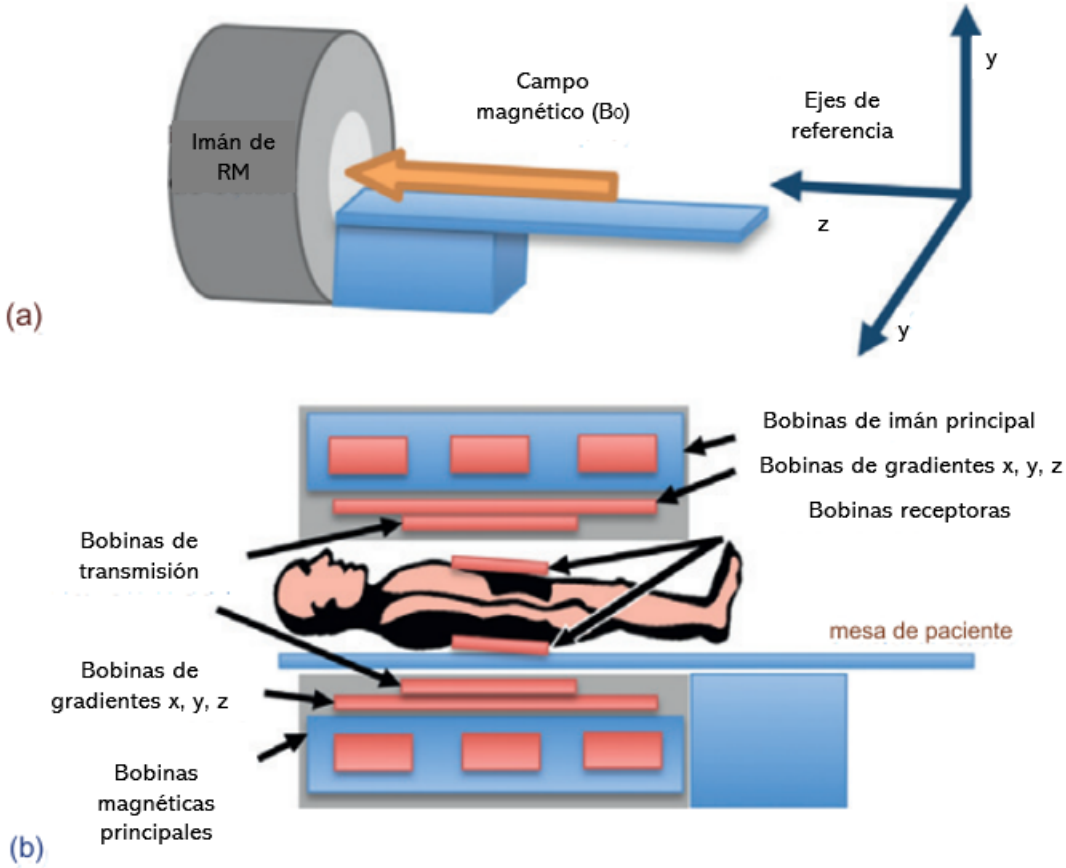


Figura 1.1: *a) Estructura básica del resonador magnético con enfásis en la dirección del campo magnético principal B_o y los ejes de referencia resultantes [1]. b) Diagrama de bobinas en el resonador magnético [2] .*

necesarias principalmente para codificar espacialmente la señal de la RM durante la obtención de imágenes.

3. Una bobina de radiofrecuencia (RF), encargada de transmitir los pulsos y secuencias de RF necesarios para producir la señal de RM.

4. Una bobina receptora de RF que capta la señal de resonancia creada en el paciente, aunque generalmente la bobina emisora actúa también como receptora de manera sincronizada.

La generación de imágenes comienza con la activación del imán, que tiene como consecuencia la magnetización paralela al campo principal de los espines de hidrógeno de los distintos tejidos. Bajo esta situación es imposible detectar la señal de estos espines,

por lo que es necesario que la magnetización de éstos deba desalinearse del campo principal con dirección hacia el plano xy, esto se logra mediante un pulso de RF (excitación). Posteriormente, el pulso de RF cesa y se detecta la respuesta a este efecto mediante la antena RF receptora [1].
La información obtenida mediante la RF receptora es almacenada y procesada computacionalmente, la construcción de la imagen y características dadas por el brillo o intensidad del píxel o vóxel dependen de una serie de factores, tales como el número físico de protones de agua en una región determinada y la tasa de su relajación después de la excitación inicial de regreso al estado fundamental. La señal de RM recibida vuelve a su estado de preexcitación a través de dos procesos independientes conocidos como T1 y T2. La relajación T1 es el componente de relajación longitudinal que ocurre en el eje z y es el resultado de que los espines excitados en el conjunto regresen a su estado fundamental de preexcitación. A medida que las moléculas giran, se crean fluctuaciones locales en el campo magnético, que varían según su tamaño. Este tipo de modalidad muestra en forma óptima la anatomía normal del tejido blando y la grasa.
Después del pulso de RF inicial, los espines están completamente en fase y producen una cantidad máxima de señal, con el tiempo, los espines adyacentes intercambian energía y adquieren diferencias de fase, reduciendo la señal en una caída exponencial. Después de un cierto tiempo conocido como T2, queda el 37 % de la señal. El resultado de este fenómeno son las imágenes ponderadas en T2 que muestran de manera óptima líquido y alteraciones (tumores, inflamación, traumatismo).
Para concluir, se puede apreciar en la Figura 1.2 que las imágenes ponderadas en T2 demuestran una alta intensidad de señal en los surcos y ventrículos, lo cual es compatible con el líquido cefalorraquídeo también resaltado. Mientras que las imágenes ponderadas en T1 demuestran una intensidad de señal baja dentro del líquido cefalorraquídeo (LCR) con materia blanca brillante y materia gris más oscura.

1.2. Procesamiento digital de imágenes

Esta ciencia se trata de la manipulación de datos digitales en forma de imágenes (por extensión también volúmenes) con la ayuda de hardware y software para producir datos digitales en los que se ha extraído, discriminado y/o resaltado información específica. Tiene como objetivos principales la mejora de la información visual para la interpretación humana y la transformación de datos como imágenes para almacenamiento, análisis, pre-

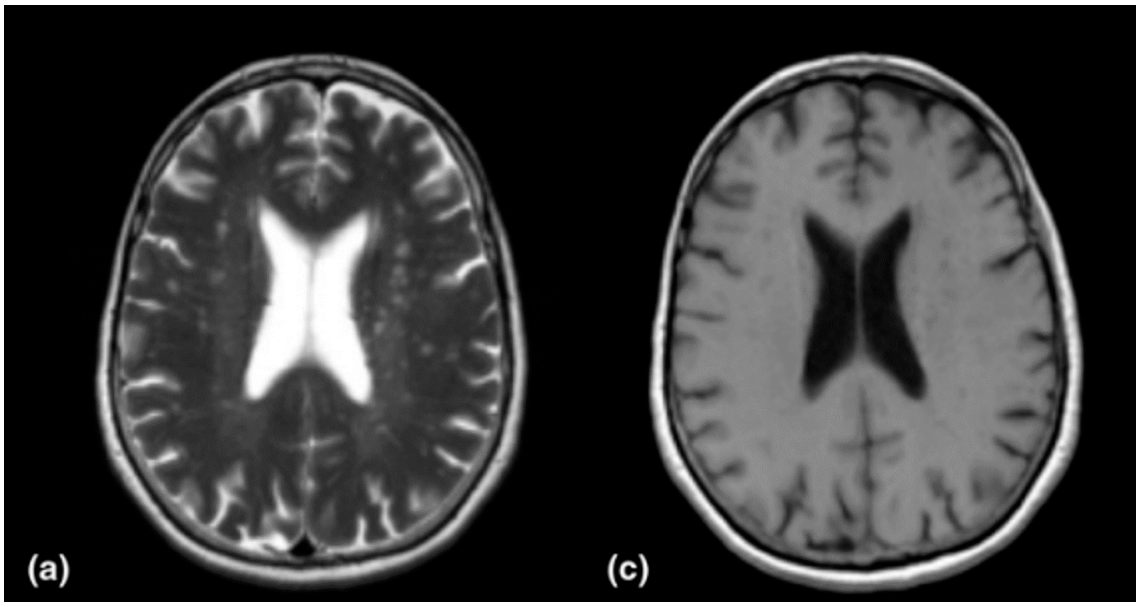


Figura 1.2: *Contraste entre tipos de ponderación en imágenes de resonancia magnética cerebral [1]. a) Imagen ponderada en T2. c) Imagen ponderada en T1.*

dicción, transmisión y representación para la percepción de sistemas computacionales. Una imagen es una representación espacial de un escenario bidimensional, para fines de manipulación se representa matemáticamente con una matriz, en la cual cada elemento representa un pixel. Para el interés de este proyecto, el tipo de datos en cuestión es el conjunto de múltiples imágenes concatenadas de forma ordenada, conocido como volumen y que tiene al vóxel como elemento operacional, formando así un dato tridimensional.

A continuación se enumeran las técnicas principales de esta ciencia [3]:

1. Adquisición de imágenes: la adquisición de imágenes implica el procesamiento previo a capturar u obtener una imagen de cualquier naturaleza.

2. Mejoramiento de imágenes: las técnicas de mejora resaltan detalles, tonos generales oscurecidos dentro de una imagen, además resaltan ciertas características de interés en una imagen, como cambios en el nivel del brillo, contraste, etc.

3. Procesamiento de imágenes a color: es un área que incluye modelado y procesamiento de matrices del espacio de color en un dominio digital, con relevancia de los modos de color utilizados en la imagen. La Figura 1.3 denota el aumento de com-

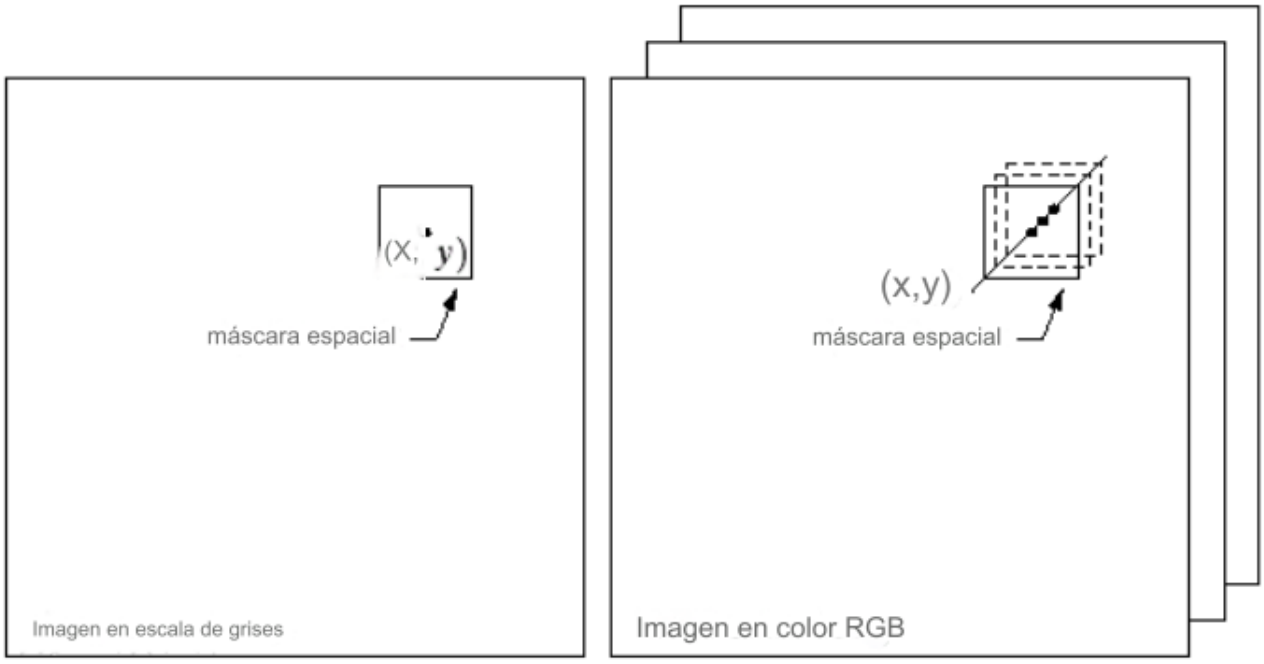


Figura 1.3: *Máscaras espaciales para escala de grises y formato RGB en imágenes [3].*

plejidad de un problema de procesamiento en este caso, desde el punto de vista de operar con objetos de dominio 3D.

4. Filtrado: es uno de los procesamientos de imagen más extensos y estudiados, en él se utilizan técnicas de filtrado para mejorar y modificar imágenes digitales según lo establecido. Los filtros de imágenes se utilizan para difuminar, reducir el ruido, enfocar y detectar bordes, también se usan para suprimir frecuencias altas (técnicas de suavizado) y bajas (mejora de imagen, detección de bordes).
   La Figura 1.4 muestra un diagrama en el que se clasifican las familias de filtros de procesamiento digital de imágenes. El filtrado espacial es el método tradicional de filtrado de imágenes, se utiliza directamente en los píxeles de la imagen, mientras que los filtros de frecuencia trabajan en el dominio de frecuencias resultante de la Transformada de Fourier bidimensional, su principal propósito es eliminar frecuencias altas y bajas en bandas específicas de forma manual o automatizada.
   Dentro del conjunto de filtros espaciales se encuentran los filtros no lineales, éstos se utilizan para detectar bordes y son conocidos por ser más efectivos que los filtros lineales, ya que en el filtrado lineal, generalmente los detalles y bordes de la

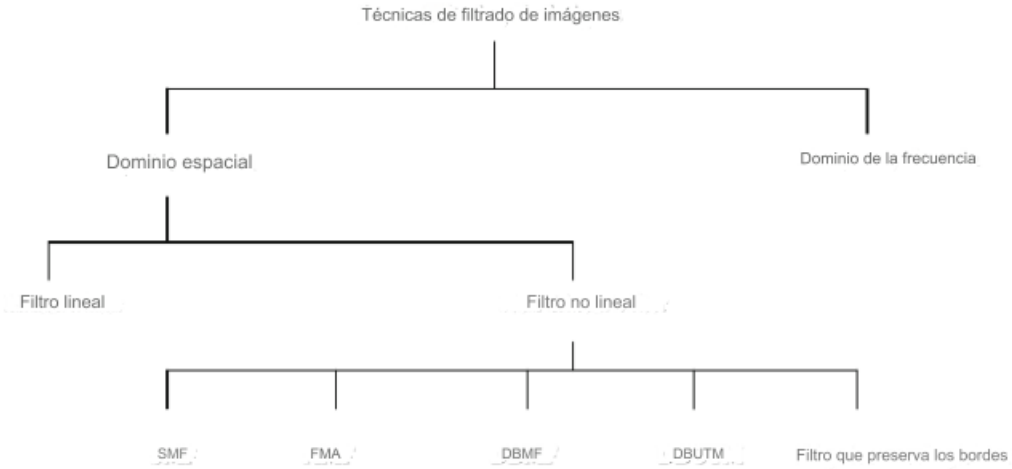


Figura 1.4: *Clasificación de filtros de imagen [4].*

imagen tienden a desenfocarse, por lo tanto, aplican una fórmula de reducción de ruido a todos los píxeles sin identificar los píxeles ruidosos; el filtro Gaussiano, el filtro Laplaciano y el filtro promedio de vecindad son ejemplos de filtros lineales. Los filtros de mediana y los filtros de corrimiento de media son filtros no lineales, la Figura 1.4 expone algunos ejemplos: filtro de mediana simple, filtro de mediana adaptativo simple (AMF), filtro de mediana basado en decisiones (DBMF) y filtro de mediana sin recortar basado en decisiones (DBUTM).

5. Wavelets y procesamiento multiresolución: a diferencia de la transformada de Fourier cuya base son las funciones sinusoides, las transformadas se basan en pequeñas ondas llamadas wavelets de frecuencia variable y extensión limitada. Las Wavelets fueron la primer demostración de un nuevo y poderoso descubrimiento para el procesamiento de señales, la llamada teoría de la multirresolución, que por definición trata del análisis de la imagen en más de una resolución, con la ventaja de que ciertas características que no pueden detectarse en cierta resolución podrían ser detectadas en otra. Las imágenes se subdividen en regiones más pequeñas como resultado de la aplicación de wavelets, y tienen aplicación en la compresión de datos y la representación piramidal, tal como se muestra en la Figura 1.5.

6. Compresión: las técnicas de compresión reducen el almacenamiento necesario para guardar una imagen o el ancho de banda para transmitirla. Especialmente para el uso a través de Internet, es muy necesario comprimir los datos.

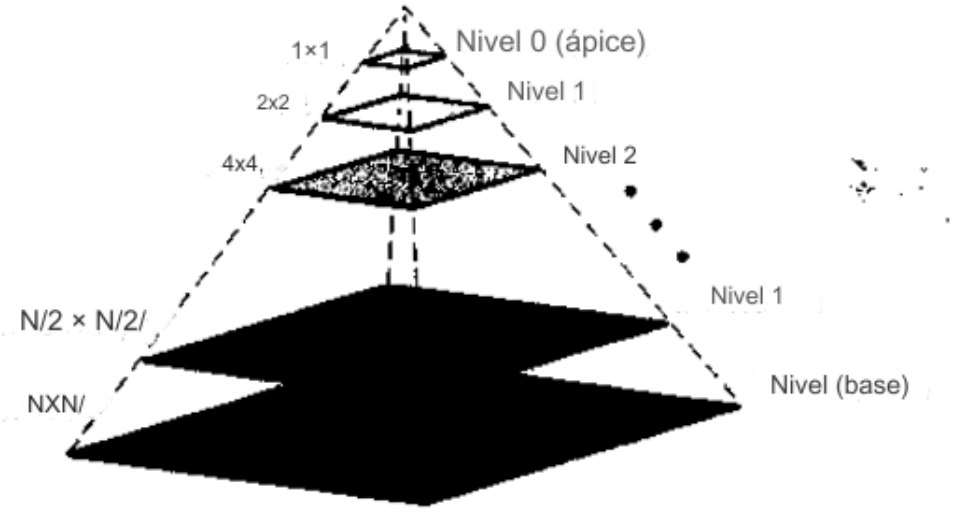


Figura 1.5: *Imagen representada como pirámide al utilizar Wavelets [3].*

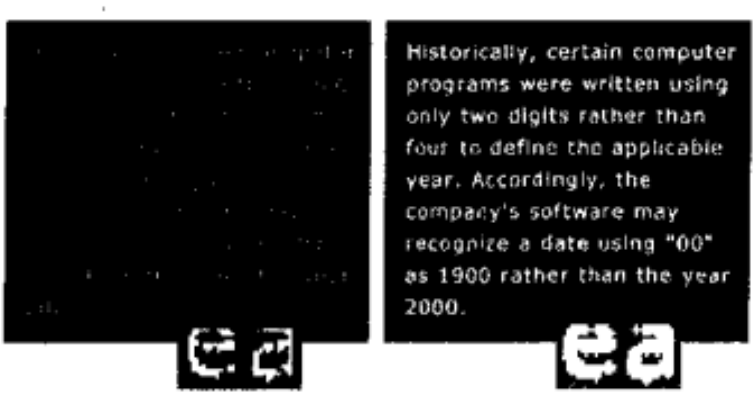


Figura 1.6: *Aplicación de un elemento estructurante de dilatación sobre una imagen de texto. Izquierda: muestra de texto con pobre resolución. Derecha: dilatación de segmentos dentro de la imagen de entrada que resulta de una imagen con una notable mejor en resolución [3].*

7. Morfología matemática: el procesamiento morfológico extrae y modifica componentes directamente de la imagen que son útiles en la representación y descripción de la forma, haciendo uso de operadores llamados elementos estructurantes. La Figura 1.6 muestra un ejemplo de una aplicación de este procesamiento sobre texto, en la cual aplicando un elemento estructurante de dilatación logra una mejor apreciación del texto.

8. Segmentación: este procesamiento tiene como objetivo dividir una imagen en par-

Figura 1.7: *Mapas de probabilidad posterior que clasifican (de izquierda a derecha) fondo, líquido cefalorraquídeo, materia gris y materia blanca, en una imagen de resonancia magnética cerebral [5].*

tes u objetos constituyentes según un criterio dado. En general, la segmentación automática es una de las tareas más difíciles en el procesamiento de imágenes digitales, ya que implica un robusto esfuerzo computacional, además del uso de más de una sola técnica particular, normalmente se busca que los objetos se identifiquen individualmente para luego diferenciarse. La Figura 1.7 ejemplifica una segmentación dentro de una imagen, para ese caso en una imagen de resonancia magnética se etiquetan los distintos tejidos que la conforman. Las imágenes de resonancia magnética son procesadas con el fin de encontrar patrones que lleven a detectar enfermedades o padecimientos en un paciente. Esta técnica es una de las más usadas en el procesamiento de imágenes y se resume como la división en subpartes de una colección de datos obtenidos, donde cada subparte tenga características que permitan diferenciarlas de otras y por ende su clasificación.

9. Reconocimiento de objetos: es el proceso en el que a un objeto dentro de una imagen se le asigna una etiqueta en función de sus descriptores previamente estudiados, debido a que se identifica un conjunto de características similares, con ello, se puede reconocer un patrón preestablecido. Este procesamiento está fuertemente ligado a algoritmos de aprendizaje maquinal y aprendizaje profundo, tareas fundamentalmente automatizadas. El ejemplo mostrado en la Figura 1.7 relaciona la segmentación con este procesamiento, ya que para este caso los dos procesos tienen como objetivo etiquetar objetos dentro de una imagen, como lo son tejidos dentro de una imagen de resonancia magnética cerebral.

1.3. Corrimiento de media

El corrimiento de media (*mean shift*) es un procedimiento recursivo que permite determinar las modas de la función de densidad de probabilidad de una muestra de datos en un espacio multidimensional [5]. Para el caso de los volúmenes de RM, este espacio multidimensional se conformará por los valores de los voxeles de la imagen, así como por sus coordenadas espaciales, lo que resulta de un espacio 4D, su función de densidad de probabilidad se estima empleando la expresión:

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left\{\frac{1}{h}(x - X_i)\right\} \tag{1.1}$$

donde h es el ancho de una vecindad alrededor del vóxel x, $K(x)$ es una función kernel definida para un x d-dimensional, y X_i es cada uno de los datos multivariados que conforman el volumen. En este estudio se empleó el kernel de Epanechnikov, definido por:

$$f(x) = \begin{cases} K(x) = \frac{1}{2c_d^{-1}(d+2)(1-x^T x)} & si \quad x^T x_1 < 1 \\ \\ 0 & otro\ caso \end{cases} \tag{1.2}$$

donde c_d es el volumen de una esfera unitaria d-dimensional. A partir de esta expresión, es posible definir el vector de corrimiento de media $M_h(x)$, correspondiente al desplazamiento iterativo que se aplica a cada punto de la distribución, para converger a las modas de la misma. La Figura 1.8 ilustra la trayectoria del vector corrimiento orientado a la moda local. Es posible demostrar que este vector apunta siempre en la dirección de máximo gradiente de la distribución:

$$M_h(x) = \frac{1}{n_x} \sum_{x_i \in S_h(x)} X_i - x = \frac{h^2 \widetilde{\nabla} f(x)}{(d+2)\hat{f}(x)} \tag{1.3}$$

donde la región $S_h(x)$ es una hiperesfera de radio h, centrada en x, y conteniendo nx puntos de datos en el espacio Euclidiano d-dimensional [6].

Las modas de la distribución en este espacio conjunto intensidad-localización corresponden a regiones homogéneas del volumen original. Sustituyendo el valor de cada voxel original por el valor de su moda en la distribución conjunta, el resultado es un volumen suavizado a tramos, donde diversos efectos de ruido habrán sido reducidos, así como una mayor diferenciación entre zonas homogéneas.

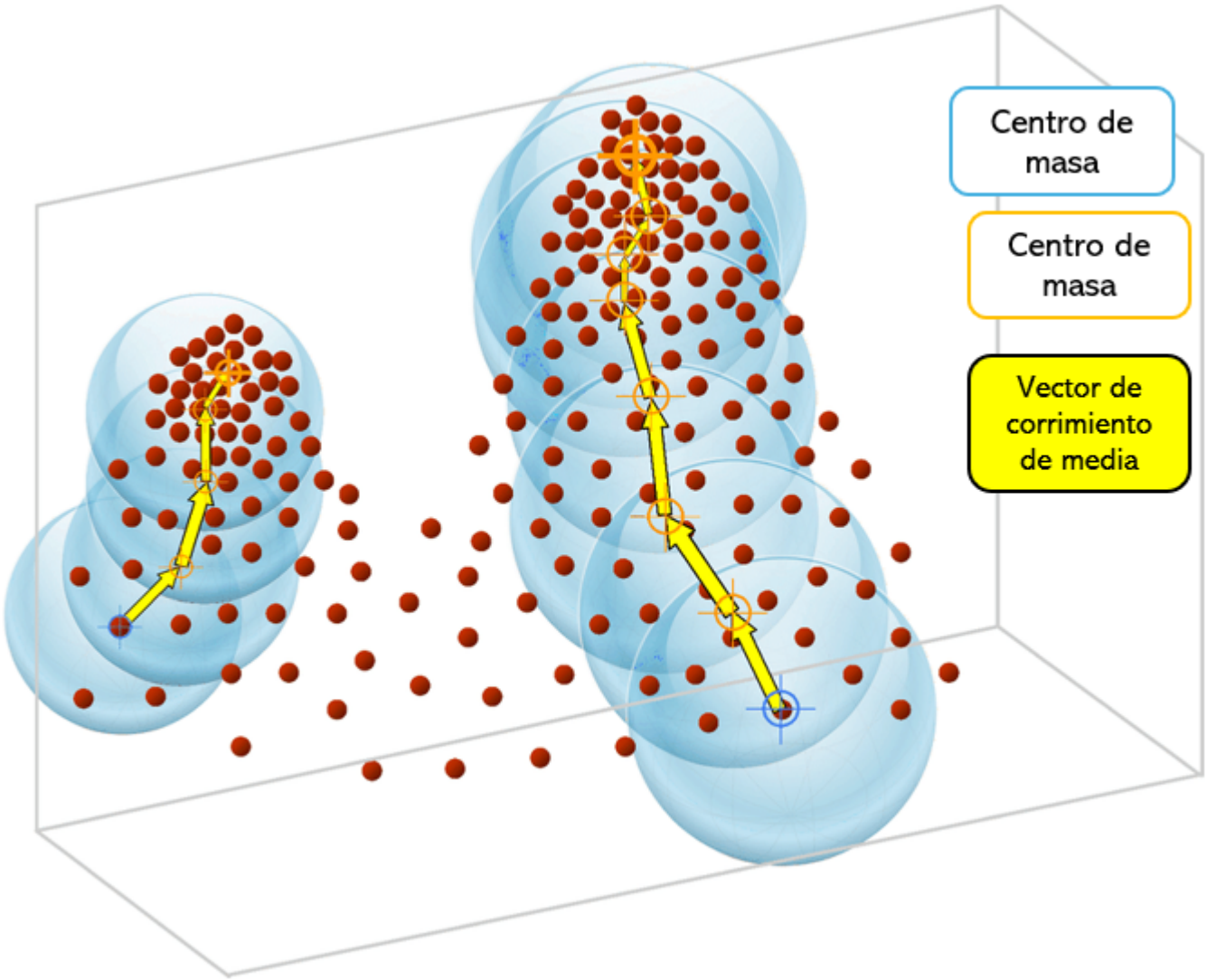


Figura 1.8: *Descripción intuitiva del procedimiento del corrimiento de media, remarcando la trayectoria del vector de corrimiento de media [6].*

1.4. Aprendizaje profundo

El aprendizaje profundo es una rama del aprendizaje automático, se considera una herramienta de suma importancia para la industria y academia, debido a sus capacidades de crear modelos basados en los datos. Esta ciencia se originó a partir de redes neuronales artificiales multicapa de gran tamaño y profundidad, se ha convertido en un tema muy popular en el contexto de la computación, además de sus múltiples aplicaciones en diversas áreas como en medicina, biomédica, reconocimiento visual, análisis de texto, ciberseguridad, entre otras.

Es conocido que el aprendizaje profundo implica un gasto computacional considerable, su uso en computadoras convencionales extiende el tiempo de entrenamiento y pruebas en gran proporción. Por lo tanto, se busca un medio de desarrollo que permita reutilizar un paquete de tareas o instrucciones en múltiples unidades de procesamiento para así lograr un mayor rendimiento computacional.

La Figura 1.9 denota el constante problema en algoritmos de inteligencia artificial (IA), el volumen de datos. Conforme la complejidad del algoritmo aumenta, también lo hace la dificultad en su entrenamiento, la cantidad de párametros internos que se optimizan durante el entrenamiento exige una gran cantidad de datos para el conjunto de entrenamiento. Aún con esto, los resultados de bibliografía reportada en los últimos años prefieren enfrentar el problema de encontrar grandes cantidades de datos que implementar modelos de menor complejidad.

Es díficil encontrar la diferencia tajante entre el aprendizaje maquinal y el aprendizaje profundo, específicamente cuando una red neuronal es lo suficientemente profunda para ser parte del último campo mencionado, un ejemplo como apoyo a la diferenciación de esta cuestión es la red neuronal convolucional (CNN o ConvNet) [7] mostrada en la Figura 1.10, ésta es una popular arquitectura discriminativa de aprendizaje profundo que aprende directamente de la entrada extrayendo características de los datos sin necesidad de intervención humana. Este tipo de red incluye múltiples capas convolucionales y de agrupación, también utiliza un *dropout*. Este último metodo desactiva un número de neuronas de una red neuronal de forma aleatoria, con objetivo de disminuir la dependencia de estas neuronas desactivadas en otras neuronas cercanas. Este metodo ayuda a reducir el sobreentrenamiento de la red ya que las neuronas cercanas suelen aprender patrones muy particulares con los datos de entrenamiento, obstaculizando una predicción más general.[8].

Las CNN están destinadas específicamente a abordar datos de dimensión 2D o supe-

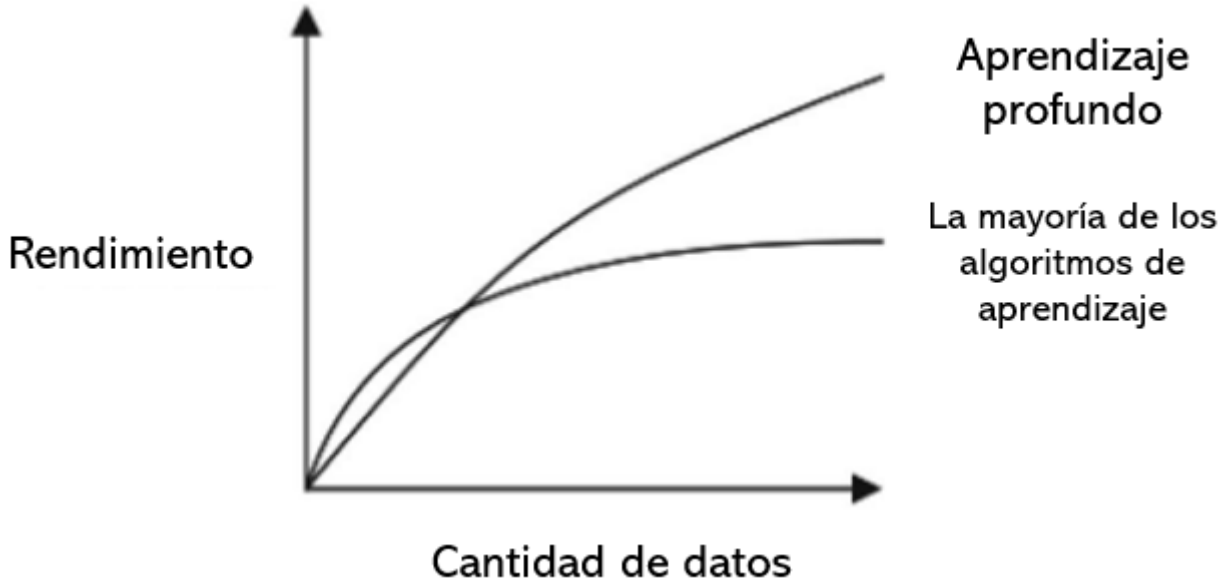


Figura 1.9: *Comparación del rendimiento entre aprendizaje profundo y algoritmos de aprendizaje maquinal [6].*

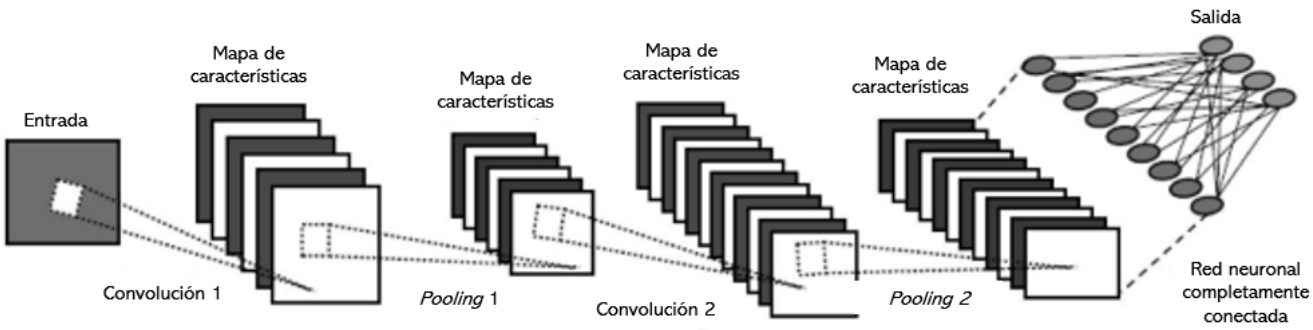


Figura 1.10: *Ejemplo de una red neuronal convolucional [14].*

rior y por lo tanto se emplean ampliamente en reconocimiento visual, procesamiento de imágenes médicas, segmentación de imágenes, procesamiento del lenguaje natural, entre otros. Existen muchas variantes de CNN tales como las arquitecturas VGG [9], AlexNet [10], Xception [11], Inception [12], ResNet [13], por mencionar algunas.

En resumen, el aprendizaje profundo es un sistema de proceso y automatización, en el cual un programa computacional aprende automáticamente representaciones y funciones útiles directamente de los datos brutos, adaptándose a ellos de manera automática. Los modelos más comunes en el aprendizaje profundo son variantes de redes neuronales artificiales. La principal característica de los métodos de aprendizaje profundo es la extracción de rasgos y patrones para descubrir funciones y realizar un proceso. Se ha de-

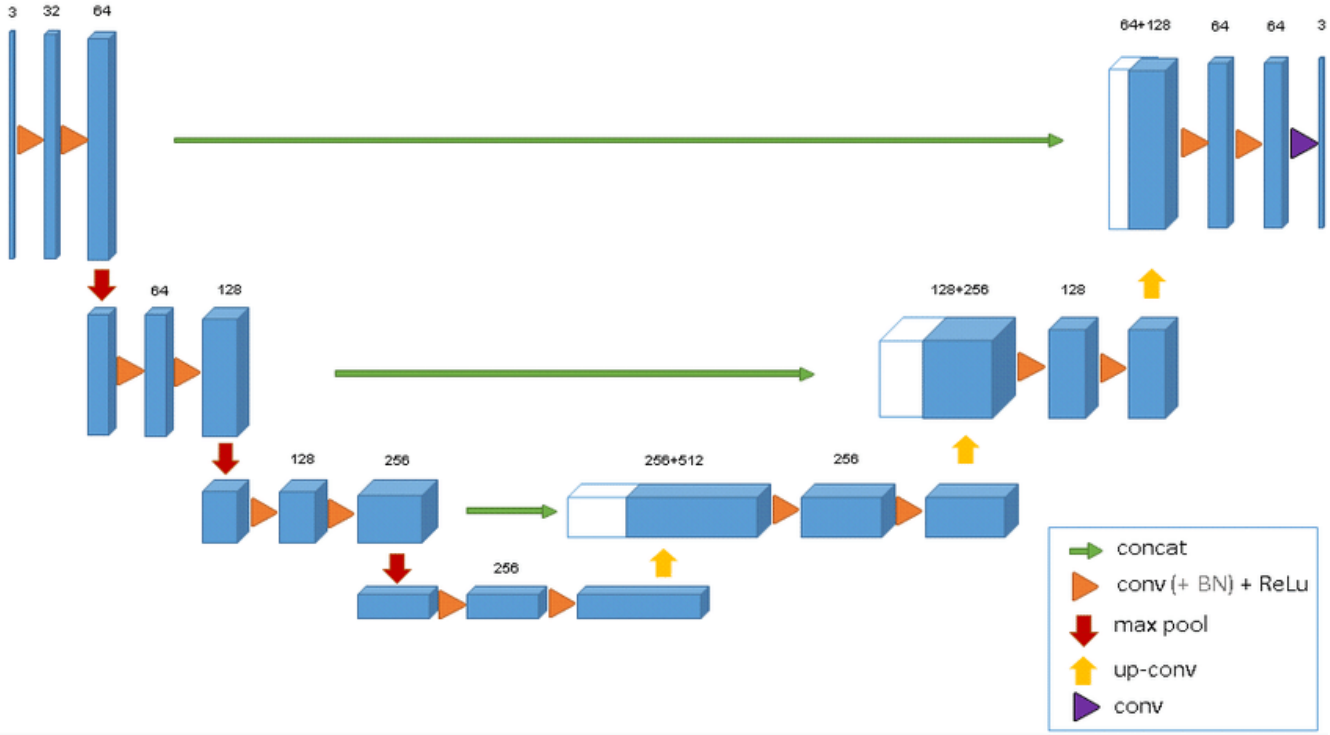


Figura 1.11: *Arquitectura de la red U-NET 3D [16].*

mostrado en trabajos previos [5], [15] que la aplicación de redes neuronales totalmente convolucionales, resuelven el problema de segmentar tejidos en imágenes con alto grado de efectividad.

1.5. Red convolucional UNET 3D

La red neuronal usada en este trabajo es de arquitectura U-Net 3D [16], la cual es una red profunda convolucional de múltiples capas, diseñada para convolucionar la imagen resultante de cada capa contra ventanas determinadas que se construyen según las características de los datos. La Figura 1.11 expone de forma gráfica su estructura, esta red puede dividirse en dos partes: la etapa de contracción y la etapa de expansión.
La primer etapa calculará parámetros y patrones particulares de cada volumen de entrada respecto de sus zonas. Este proceso se realiza con la convolución de kernels particulares para tener volúmenes de salida, donde se resalten ciertas características dado el kernel utilizado. Al aplicarse esta operación múltiples veces, una por cada característica a resaltar, se tiene en cada capa la misma cantidad de volúmenes resultantes. La cantidad

de volúmenes generados por capa es muy grande, por lo que es necesario muestrear; en este punto se utilizan capas para el submuestreo llamadas *max-pooling*, con el objetivo de reducir el tamaño de los volúmenes de salida convolucionados en una relación de 4-1 vóxeles.
La segunda etapa se encarga de construir el mapa de segmentación semántica y se compone de capas de deconvolución e incremento de resolución (*up sampling*), haciendo uso de los vólumenes de salida obtenidos en cada etapa de contracción, los cuales se concatenan a capas de la etapa de expansión de manera ordenada.
Todo el proceso mencionado se realiza con cierto grado de error en cada capa. Posteriormente este valor se utiliza para mejorar el sistema y disminuir el error mediante entrenamiento. Por lo tanto, el sistema aprende y optimiza el proceso.

1.6. Aceleramiento por GPU

La unidad central de procesamiento (CPU) es un elemento computacional diseñado para efectuar operaciones lógicas y matemáticas eficientemente, por lo que es esencial para los procesamientos digitales de imágenes como recuperación, filtrado, morfología matemática, entre otras. Pero existe el problema en el que la carga gráfica sobre la CPU es muy grande, tal es el caso de un filtro espacial formado por procesos iterativos convergentes, o bien, del modelado, entrenamiento, validación y prueba de una red neuronal profunda. Dado que el poder computacional que exigen tareas de esta magnitud no es suficiente por parte de la CPU y que en realidad no está específicamente diseñada para estas tareas, se plantea el uso de la unidad de procesamiento gráfico (GPU), la cual es un coprocesador dedicado al procesamiento de gráficos y operaciones en paralelo con números de punto flotante.

La computación acelerada por GPU permite asignar a ésta las tareas de la aplicación donde la computación es más intensiva y requiere un esfuerzo computacional muy alto, mientras que el resto de tareas se ejecutan de forma regular en la CPU. Una ventaja del uso de la GPU es que una CPU tiene pocos núcleos diseñados y optimizados para el procesamiento en serie secuencial, mientras que la GPU cuenta con una arquitectura especificamente diseñada en paralelo que contiene miles de núcleos más eficientes y que se diseñaron especialmente para resolver varias tareas al mismo tiempo.

Es por ello que en este proyecto se plantea el uso de una GPU para el modelado, entrenamiento y puesta en marcha del sistema, esto porque el filtrado espacial tiene inmerso

múltiples operaciones vectoriales recursivas en espacios multidimensionales, mientras que el modelo, entrenamiento y prueba de la red profunda representa también una gran carga computacional por sí sola [17]. En resumen, el uso de la GPU para el desarrollo de este trabajo será fundamental tomando en cuenta la inmensa exigencia computacional para el fin presentado.

1.6.1. TensorFlow-GPU

TensorFlow [18] es una biblioteca de software de código abierto disponible en *Python*, dónde es posible diseñar e implementar grandes cálculos numéricos con enfoque en aplicaciones de aprendizaje maquinal y aprendizaje profundo. TensorFlow permite a los desarrolladores crear grafos de flujo de datos, es decir, estructuras que describen cómo se comportan los datos a través de una serie de nodos de procesamiento. Cada nodo en el grafo representa una operación matemática y cada conexión o borde entre nodos es una matriz de datos multidimensional o bien, un tensor. En la Figura 1.12 se muestra un ejemplo en el cual esta biblioteca permite que un modelo de inteligencia artificial disponible se describa como un grafo dirigido de operaciones conectadas, específicamente se aplica la función de activación ReLu, la multiplicación matricial y la suma de entradas, pesos y sesgos. Esta biblioteca tiene mayor enfoque en el entrenamiento e inferencia sobre redes neuronales profundas, por lo que los grafos que representan las conexiones en arquitecturas de redes de este tipo contienen estructuras con miles de nodos e interconexiones, esto supone un gasto computacional formidable.
Las aplicaciones de TensorFlow se pueden ejecutar en distintos entornos que soporten su ejecución, tales como una máquina local, un clúster en la nube, dispositivos iOS y Android o en CPU, pero dado que los cálculos requeridos en modelos de IA requieren un gran gasto computacional, TensorFlow optó por adaptar su funcionamientos a entornos que tienen tarjeta gráfica, formando Tensorflow-GPU, con la idea de utilizar más núcleos de forma simultánea, esto se traduce en paralelizar operaciones de grafos en plataformas habilitadas para GPU, usando dispositivos portátiles, computadoras de escritorio y servidores de alta gama.

Considerando GPUs y CPUs convencionales enfrentando una misma tarea, TensorFlow-GPU opera de forma más eficiente en comparación a su versión en CPU, aunque es importante mencionar que la magnitud de esta ventaja depende fuertemente de la gama y modelo de la GPU. En resumen, esta biblioteca acelerada por GPU provoca que la ejecución reduzca su tiempo en operaciones matriciales y tensoriales, ampliamente utilizadas

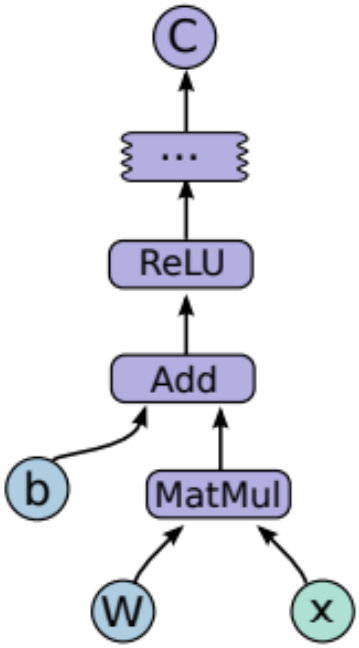


Figura 1.12: *Ejemplo de modelo de grafo implementado con TensorFlow, que simula el funcionamiento de una neurona en una red neuronal artificial [18].*

en aprendizaje maquinal y aprendizaje profundo.

1.6.2. CUDA

CUDA *(Compute Unified Device Architecture)* es una plataforma de computación paralela y un modelo de programación, que permite incrementos descomunales en el rendimiento de tareas computacionales, al aprovechar la potencia de la unidad de procesamiento de gráficos (GPU) [19]. Esta plataforma se orienta en la metodología SIMD *(Single Instruction, Multiple Data)*, la cual aplica una instrucción que codifica una operación aplicable simultáneamente a varios operandos, ésta metodología maneja operaciones de enteros o flotantes, pero con mejor rendimiento en las últimas mencionadas. El uso de instrucciones SIMD representa una mejora importante en la ejecución de código, con mayor relevancia cuando una rutina contiene una gran cantidad de ciclos, ya que se reduce significativamente el número de instrucciones.

La Figura 1.13 muestra los niveles anidados de hilos, bloques de hilos y mallas de bloques de hilos, además de los niveles correspondientes de memoria: local, compartida y memorias globales por hilo, por bloque de hilos y por compartición de datos de aplicacio-

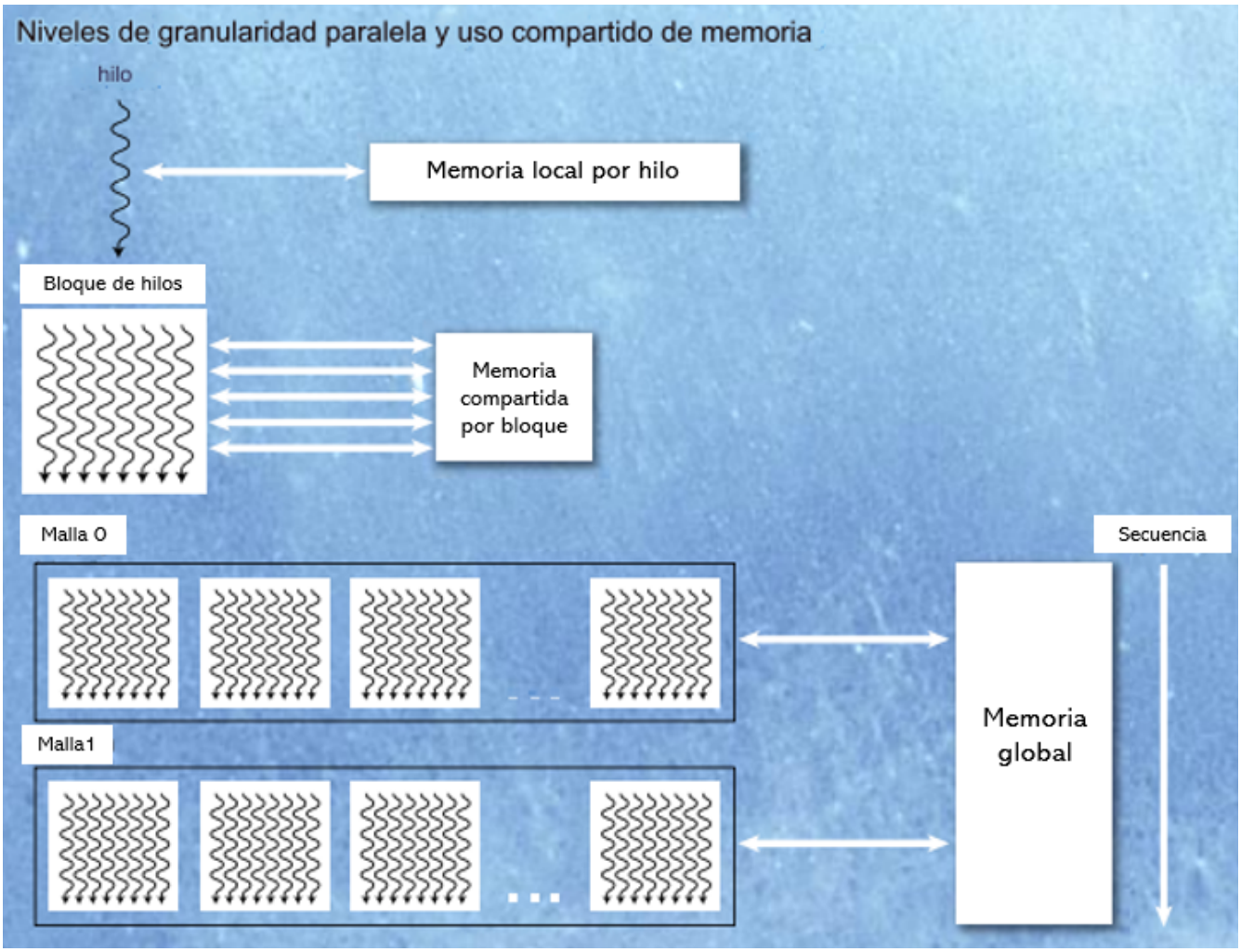


Figura 1.13: *Estructura de niveles de granularidad paralela y uso compartido de memoria [20].*

nes. La importancia de esta organización radica en el aprovechamiento de recursos, ya que el hilo, como unidad básica de operación, se sitúa como componente fundamental de un módulo de trabajo y al contar con múltiples estructuras similares que pueden ser programadas bajo una misma tarea, permiten la paralelización de un mismo proceso a múltiples operandos, por lo que esta disposición de medios motiva a dar esta aplicación a las tarjetas gráficas.

Los núcleos CUDA son unidades de procesamiento paralelo que se encuentran en las GPUs NVIDIA. Éstos permiten que las GPU contribuyan al procesamiento de datos junto con las aplicaciones de renderizado al realizar cálculos y ejecutando tareas. Además, estos núcleos están diseñados específicamente para la programación CUDA, lo que per-

mite a los desarrolladores aprovechar las GPUs para ejecuciones de programación de uso general (incluyendo la simple renderización) en las cuales se aceleran ciertas tareas en paralelo donde una CPU convencional es menos eficiente.

Python desempeña un papel clave dentro del ecosistema de aplicaciones de ciencia, ingeniería, análisis de datos y aprendizaje profundo, por lo que NVIDIA estableció soporte al ecosistema Python para aprovechar el rendimiento paralelo y acelerado de las GPUs, ofreciendo bibliotecas, herramientas y aplicaciones estandarizadas con dicho soporte. Se estableció la unificación del ecosistema Python-CUDA con un único conjunto estándar de interfaces de bajo nivel, proporcionando cobertura completa y acceso a las API del entorno CUDA desde Python, esto se traduce en un flujo de implementación menos compleja y una mejor comunicación con los recursos del hardware. La metodología para el desarrollo de una aplicación se ilustra en la Figura 1.14. El primer paso es codificar las instrucciones en Python, compilar el código en el entorno GPU después de traducirlo automáticamente a CUDA C++, transferir recursos de la memoria RAM a la VRAM en GPU en caso que sean necesarios, esperar el procesamiento paralelizado en la GPU y por último recuperar los resultados desde la GPU para un posterior procedimiento. La ventaja está en la automatización de los procesos mencionados, siendo así que el desarrollador se encarga de programar la aplicación, recursos y el entorno desde un lenguaje de alto nivel.

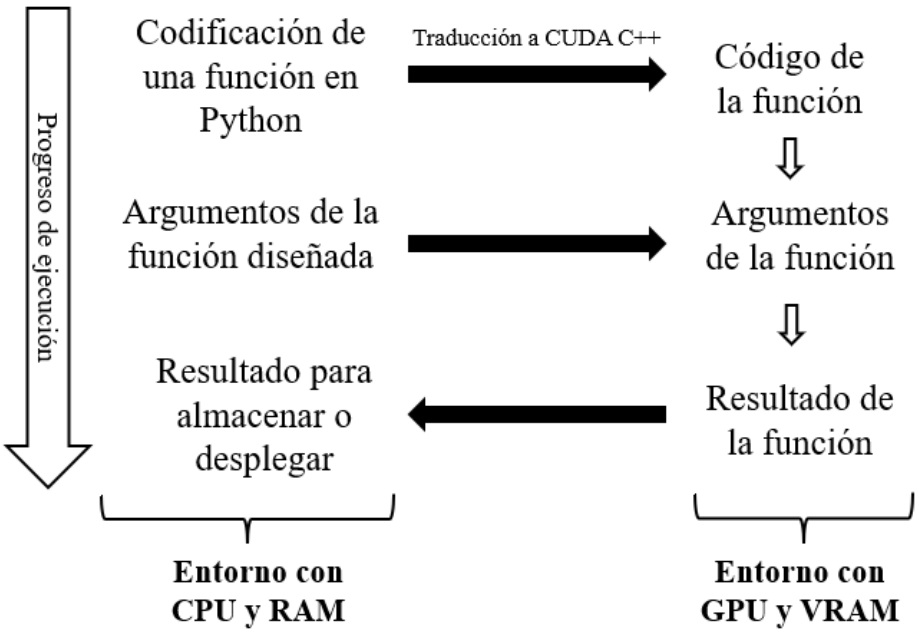


Figura 1.14: *Diagrama de flujo de datos y funciones entre entornos con CPU y GPU.*

Capítulo 2

Sobre este trabajo

2.1. Planteamiento del problema

La segmentación es un componente del procesamiento digital de imágenes que permite extraer y manipular regiones homogéneas según un criterio dado. El análisis de las regiones segmentadas es relevante cuando conlleva la detección o la valoración de padecimientos clínicos; en la mayoría de los casos el tiempo de procesamiento es crucial hasta el punto de ser crítico en situaciones previas a una intervención médica importante, por lo cual la ejecución de este análisis no sólo debe cumplir con la calidad esperada, sino también considerar su rapidez.
Si bien existe un gran número de algoritmos para la segmentación de imágenes, en general son diseñados con propósitos específicos. El caso de la segmentación de IRM no es la excepción. En este proyecto se plantea en contraste a lo mencionado, un abordaje general de la segmentación de volúmenes completos de IRM ”guiada por los datos”, es decir, utilizando un sistema de filtrado espacial acoplado a una red neuronal profunda, siendo ambos métodos del dominio del aprendizaje maquinal.

2.2. Pregunta de Investigación

¿Cómo impacta el filtro espacial de volúmenes, formado por el algoritmo del corrimiento de media potenciado por el mapa de confianza de bordes, cuando es acoplado a una red convolucional de arquitectura U-NET 3D, que tiene como objetivo clasificar materia gris, materia blanca, líquido cefalorraquídeo, cráneo y fondo, en volúmenes de resonancia magnética cerebral?

2.3. Hipótesis

El acoplamiento del filtro espacial formado por el algoritmo corrimiento de media potenciado con el mapa de confianza de bordes, previo a una red convolucional de arquitectura U-NET 3D, forma un segmentador de volúmenes de resonancia magnética cerebral que facilita el proceso de entrenamiento además de reportar una precisión superior, en comparación a un segmentador de las mismas características pero sin el uso del filtrado espacial.

2.4. Objetivos

2.4.1. Objetivo general

Elaborar un sistema de segmentación de volúmenes de resonancia magnética cerebral que clasifique materia gris, materia blanca, líquido cefalorraquídeo, cráneo y fondo, utilizando un bloque de filtrado espacial y una red neuronal profunda operando directamente en el espacio de volumen 3D.

2.4.2. Objetivos específicos

1. Extender la solución 2D del trabajo previo a una implementación 3D tanto del corrimiento de media como de la red U-Net.

2. Implementar el sistema de segmentación en un medio computacional acelerado por GPU.

3. Evaluar el sistema utilizando validación cruzada para establecer su precisión e intervalo de confianza.

2.5. Justificación

El procesamiento digital de imágenes tiene una gran importancia en el campo de la medicina, ya que sus aplicaciones abarcan desde el mejoramiento de una imagen médica por contaminación de ruido externo, hasta el uso de morfología matemática para contabilizar bacterias en una muestra de laboratorio. Aunque generalmente, se utilizan herramientas establecidas con un propósito específico y con una implementación de la misma naturaleza, esto es problemático en cuestión de tiempo y recursos, ya que es necesario el ajuste manual de parámetros o procesos, sobretodo cuando el procesamiento de imágenes médicas retrasa diagnósticos, investigaciónes, cirugías, consultas, entre otros. Por esta razón, el ingeniero biomédico tiene que hacer uso de sus conocimientos multidisciplinares de tipo teóricos y prácticos, para dar soluciones efectivas a situaciones cuyos problemas no son solamente de precisión, sino de tiempo y adaptabilidad a los datos.

El rol de la inteligencia artificial en la Ingeniería Biomédica es relevante pues tiene herramientas que ayudan a la predición o clasificación de datos biomédicos con alta precisión, esto es un gran paso en múltiples investigaciones, sin contar las aplicaciones directas en áreas hospitalarias. La riqueza de esta ciencia radica en tener sistemas guíados por los datos, es decir, que aprenden automáticamente tendencias y patrones directamente del comportamiento de los datos, sin el uso de párametros iniciales o ajustes manuales sobre la ejecución. En tiempos recientes, el uso de tarjetas gráficas computacionales han logrado mermar la conocida problemática en los tiempos de entrenamiento y ejecución de los algoritmos de este campo, lo que termina por consolidar su gran aprovechamiento, siempre y cuando se tome en cuenta el costo ecónomico de las GPUs *vs* su rendimiento.

Se conocen métodos que resuelven el problema de identificar tejidos en estudios de resonancia magnética cerebral, muchos de ellos usan distintos algoritmos del campo del procesamiento digital de imágenes, entre ellos están el filtrado, la morfología matemática y la umbralización, mientras que el campo de la inteligencia artificial también ha realizado sistemas para dar solución a este objetivo, haciendo uso de redes neuronales convencionales, redes neuronales convolucionales e incluso una combinación de éstas. En este trabajo se toman tres procesamientos conocidos por funcionar efectivamente cuando son aplicados, por el lado del procesamiento digital de imágenes se toma el mapa de confianza de bordes y por parte de la inteligencia artificial, el corrimiento de media y la red convolucional. El objetivo es hacer uso de los tres procesamientos mencionados de forma acoplada para generar un sistema que logre segmentar materia gris, materia blanca, líquido cefalorraquídeo, cráneo y fondo, directamente de vólumenes completos

de RM cerebral. Esto pretende resolver distintas problemáticas, tales como el aprovechamiento efectivo de los recursos computacionales, el rendimiento del proceso y la calidad de los resultados.

Capítulo 3

Antecedentes

3.1. El papel del aprendizaje profundo en procesamiento de imágenes médicas

Sarker (2021) [14] conjunta de manera categórica las técnicas, aplicaciones y directivas del aprendizaje profundo, destacando su importancia en años recientes. Se plantea como la rama del aprendizaje automático y la inteligencia artificial considerada como la tecnología central de la Cuarta Revolución Industrial. Asentando que su más grande capacidad es la de aprender directamente de los datos, con aplicaciones en la atención médica, reconocimiento visual, análisis de texto, ciberseguridad y muchas más. Sin embargo, la problemática surge en la construcción de un modelo de aprendizaje profundo apropiado, debido a la naturaleza dinámica, las variaciones en los problemas y datos del mundo real. Además, se resalta que la elección e implementación de un algoritmo para resolver un problema, generalmente, es una tarea arbitraria y sin consciencia en el contexto de los datos y objetivo, esto se debe a la falta de comprensión básica y fundamental de los métodos de aprendizaje profundo, lo que termina por convertir a los sistemas de aprendizaje automático en cajas negras que obstaculizan el desarrollo. El autor presenta una visión estructurada y completa de un gran compendio de técnicas de aprendizaje profundo, considerando varios tipos de aplicacion en el mundo real. Además de clasificar el contexto de los algoritmos de este campo según sus características, como técnicas super-

visadas o no supervisadas, discriminativo o generativo, así como el aprendizaje híbrido y otros relevantes. En general, este artículo expone una visión general del modelado de técnicas de aprendizaje profundo que pueden usarse como guía de referencia tanto para objetivos académicos o profesionales en la industria.

Akkus et al. (2017) [21] presentaron de forma clara y precisa elementos complejos del tema de interés, aprendizaje profundo, ya que da un ejemplo de la aplicación de segmentación basada en esta estructura. Comienza por mencionar las diferentes técnicas y tipos de imagenología exploradas, así como sus ventajas sobre las demás. En el caso de la resonancia magnética, se mencionan conceptos fundamentales, usos comúnes de las imágenes y las aplicaciones actuales de aprendizaje profundo que estudian esta imagenología. También menciona la estructura, conceptos e ideas básicas del aprendizaje profundo, partiendo desde las redes neuronales convencionales. Por último, muestra ventajas de la segmentación al utilizar aprendizaje profundo comparado con otras técnicas de procesamiento digital de imágenes más populares, mediante tablas y parámetros de una extensa biblioteca de referencias. Este artículo impacta en el presente proyecto ya que da un contexto general de aplicaciones de aprendizaje profundo en preprocesamiento, procesamiento, detección de enfermedades o trastornos y sobre todo, segmentación de imágenes.

De igual manera, Lundervold y Lundervold (2018) [15] resaltaron la importancia que puede tener el aprendizaje profundo empleado en imagenología médica y específicamente en procesamiento de imágenes de resonancia magnética, además de informar los avances y estado del arte hasta el momento. A lo largo del texto se menciona cómo se aplica el aprendizaje profundo en el campo de la imagenología médica, algunos ejemplos son: adquisición y recuperación de imágenes, segmentación de imágenes y la predicción de una enfermedad o trastorno. Este artículo es de gran ayuda, ya que funge como punto de comienzo en este proyecto, sienta las bases sobre lo que se ha hecho, se está haciendo y en un futuro se logrará en el campo del aprendizaje profundo enfocado al entorno biomédico.

El trabajo de Aswathy y Chandra (2022) [22] es un ejemplo de cómo el aprendizaje profundo tiene un papel significativo en la detección de infecciones en imágenes de tomografía computarizada (TC). Han desarrollado sistema de redes convolucionales 3D U-Net en cascada de dos etapas, esto para segmentar el área contaminada en los pulmones. La primer red 3D U-Net extrae el parénquima pulmonar de la entrada, la cual es un volumen de la TC después de un preprocesamiento. Dado que el volumen de TC es pequeño, se aplica el posprocesamiento adecuado a la parénquima pulmonar e ingresan estos vo-

lúmenes en la segunda red 3D UNet. Esta última tiene como función extraer el volumen 3D infectado. Con este método, los médicos pueden ingresar el volumen completo de TC del paciente y obtener como resultado el área contaminada sin tener que etiquetar el parénquima pulmonar para cada nuevo paciente. Para segmentación del parénquima pulmonar, el método propuesto obtuvo una sensibilidad del 93.47%, especificidad del 98.64%, una precisión del 98.07% y un coeficiente de Dice del 92.46%. Por otro lado, obtuvieron una sensibilidad del 83.33%, una especificidad del 99.84%, una precisión del 99.20% y una puntuación de Dice del 82% para la segmentación de la infección pulmonar. La importancia de este trabajo está en el extenso aprovechamiento del modelo de red convolucional U-Net, ya que resalta que los resultados de su correcto uso son favorables independientemente del objetivo de segmentación y reafirma la confiabilidad de esta arquitectura de red en la segmentación de imágenes médicas.

Chen (2016) [23] comienza enunciando la popularidad de los últimos años por usar unidades residuales para entrenar redes neuronales profundas, enfatizando la presencia de este avance en tareas de reconocimiento de imágenes 2D, tales como la detección y segmentación de objetos o estucturas. El objetivo de su trabajo es afrontar los retos de segmentar, con esta metodología, datos volumétricos como lo son imágenes de resonancia magnética cerebral en distintas modalidades . La propuesta primordial es la creación de la arquitectura de red conocida como VoxResNet, que tiene como característica principal el aprovechamiento de la combinación de características de bajo nivel de la imagen, información de tamaño y el contexto de alto nivel. Al final del estudio, las métricas calculadas superan el rendimiento de 37 competidores previamente establecidos, lo que instaura un precendente acerca de un método relativamente nuevo en la segmentación de imágenes de resonancia magnética cerebral, siendo así un referente en el desarrollo de este trabajo.

3.2. Aceleración por GPU

El procesamiento digital de imágenes es una tarea relacionada con grandes tiempos de ejecución y de una importante demanda computacional, especialmente cuando se trabaja con imágenes médicas. Existen varias aplicaciones de imágenes médicas en las que se exige que el procesamiento de imágenes se lleve a cabo de manera rápida y eficiente, garantizando la calidad para el posterior proceso realizado por investigadores o médicos, dependiendo el caso. El uso de unidades de GPUs en el procesamiento de imágenes mé-

dicas ha aumentado recientemente debido a su alta eficiencia y capacidad de paralelizar tareas, lo cual minimiza el tiempo de espera.
Talukder (2022) [17] explora los diversos algoritmos de procesamiento de imágenes basados en GPU utilizados para imágenes médicas con comparaciones cuantitativas basadas en el rendimiento con sus respectivas contrapartes basadas en CPU. Específicamente se analiza la segmentación de GPU impulsada por NVIDIA Clara dentro de la aplicación 3D Slicer para demostrar las capacidades de segmentación de GPU enfrentando al reto de clasificar tejidos imágenes de resonancia magnética cerebral con un tumor cerebral de tamaño considerable.

El artículo realizado por Oza y Joshi [24] es relevante, ya que integra la aceleración por GPU y el filtrado de imágenes médicas, estos temas son fundamentales para el desarrollo de este trabajo, aún más cuando se aplican de manera conjunta. Los autores consideran a la eliminación de ruido en imágenes médicas como extremadamente crítica por la retención de las estructuras de los tejidos corporales y la importancia de definir los contornos para diagnóstico adecuado. El filtro bilateral espacial no lineal propuesto les proporciona una solución de eliminación de ruido eficiente, pero con la problemática del bajo rendimiento cuando se aumenta el tamaño y la resolución de la imagen.
Los autores enfrentan este común desafío en el procesamiento de grandes volúmenes de datos con el uso de una GPU para paralelizar el procesamiento, el propósito de su trabajo de investigación es lograr una ejecución de este filtrado en tiempo real utilizando una GPU compatible con CUDA para eliminar ruido que contamina imágenes médicas de diferente tamaño y resolución. Al terminar, se realiza una comparación de rendimiento entre el entorno con GPU contra el entorno con CPU. Para el desarrollo e implementación de este proyecto se usan kits de herramientas para la implementación de dominios secuenciales en Matlab y OpenCv. La plataforma de implementación consta de una GPU GTX830 de Nvidia alojada en un entorno con una Intel i5 de doble núcleo a 2.2 GHz. Se observó que la implementación GPU del filtro bilateral completó la tarea de eliminación de ruido en 1.175 ms, lo cual resulta en 613 veces más rápida que su versión de CPU basada en Matlab y 89 veces más rápida que su versión de CPU basada en OpenCV para una imagen de entrada con resolución de 64 Megapíxeles. Además, para el rendimiento de la GPU CUDA, se logra una ocupación promedio del 85 %, lo que indica una paralelización efectiva.

3.3. Segmentación de imágenes médicas cerebrales

Jiménez-Alanis et al. [25] presentan un trabajo enfocado en la segmentación de imágenes de resonancia magnética cerebral bajo una metodología que aplica una estimación de densidad no paramétrica, utilizando el algoritmo del corrimiento de media en el dominio de rango espacial conjunto. La conocida problemática acerca de los límites entre clases se enfrenta con la inclusión de un mapa de confianza de bordes, el cual se traduce en el grado de confianza de que cierto pixel sea parte de una frontera entre regiones adyacentes, con ello es posible construir un gráfico de adyacencia con las regiones etiquetadas, después se analiza y se poda para fusionar regiones adyacentes.
Posteriormente, se toma el criterio de llevar a cabo una normalización espacial entre los datos de la imagen y los mapas de probabilidad estándar, para poder asignar regiones de la imagen a un tipo de tejido cerebral, tal que para cada estructura se aplica un criterio de probabilidad posterior máxima. El método diseñado por Jiménez-Alanis y equipo, se aplicó a imágenes sintéticas y reales, utilizando el mismo conjunto de parámetros durante todo el proceso para cada tipo de dato, respetando la idea de ser un método automatizado. La combinación de segmentación regional y la detección de bordes demostró una alta precisión, con identificaciones adecuadas de los grupos sin importar el nivel de ruido y el sesgo. Comparando su desempeño en la segmentación de materia gris, sustancia blanca y fondo, las segmentaciones del modelo creado *vs* las segmentaciones de referencia, obtuvieron índices de Tanimoto promedio de 0.90 a 0.99 para datos sintéticos y de 0.59 a 0.99 para datos reales.

El trabajo de Islam [26] et al. aborda la segmentación de imágenes multimodales como un gran recurso utilizado en muchas aplicaciones médicas. En los últimos años se han introducido numerosos métodos para resolver el problema de la segmentación de imágenes médicas multimodales, usando filtrado, umbralización, aprendizaje maquinal, aprendizaje profundo e incluso de forma manual. Los autores proponen una solución para la tarea de segmentación de tumores cerebrales con la siguiente estructura: se comienza introduciendo un método de mejora en un conjunto de datos de imágenes por resonancia magnética existente mediante la generación de imágenes de tomografía computarizada sintéticas, para después, analizar un proceso de optimización sistemática de una arquitectura de red neuronal convolucional que utiliza este conjunto de datos mejorado, con el fin de adaptarlo a la tarea establecida, mostrando que el método propuesto supera a métodos similares existentes.
Por otro lado, los autores utilizan conjuntos de datos disponibles públicamente, lo que

reafirma la idea de que el uso de datos de libre acceso permite desarrollar modelos de aprendizaje profundo de alta precisión para posteriores aplicaciones con datos particulares reales, como los generados cotidianamente en hospitales, enfrentando así el constante debate del uso de imágenes de repositorios con presuntamente poca variabilidad que pudiera dificultar la adaptación a otros datos no conocidos previamente.

Finalmente, el presente trabajo funge como expansión al realizado previamente en [5], proyecto en el que se logró realizar segmentaciones 2D sobre imágenes de resonancia magnética cerebral utilizando un filtro espacial, conformado por la combinación entre un mapa de confianza de bordes [25] y el algoritmo corrimiento de media, además de una red neuronal convolucional, bajo la arquitectura de red U-Net. El promedio total de la precisión en el procedimiento de validación cruzada fue mayor al 96 %, además de demostrar que el uso del módulo de filtrado espacial propuesto entrega procesos de entrenamiento en los cuales se tienen progresiones parsimoniosas del valor de la función de costo y de la función precisión época a época. Uno de los retos del presente proyecto, es analizar si la arquitectura de este sistema de segmentación es viable dado el anterior, ya que se mantiene la estructura de filtrado espacial previo a la red neuronal, pero utilizando la extrapolación del mapa de confianza de bordes y el corrimiento de media a un espacio 3D, mientras que la red profunda opta por una arquitectura de red U-Net pero con operaciones de convolución, max-pooling y concatenación, de naturaleza 3D, entre otras. Uno de los mayores retos abordados en este trabajo en relación a [5], es la implementación de clasificar un tipo de tejido más, el cráneo. El aumento de una etiqueta al proceso de clasificación hace que este problema aumente su complejidad en gran manera, a pesar de sólo aumentar una clase más. Esto se debe a que los bancos de filtros, parámetros internos, ajuste y entrenamiento en la red neuronal aumentan en gran escala, lo que produce también un aumento de exigencia computacional considerable.

Capítulo 4

Metodología

4.1. Materiales

El desarrollo de este trabajo se realiza en tres entornos distintos dependiendo la parte del proyecto desarrollado, todos utilizando el lenguaje de programación *Python*. Las especificaciones de las GPUs usadas se muestran en la Tabla 4.1, los entornos finales utilizados fueron:

1. Entorno local: distribución Anaconda, utilizando el software de desarrollo Jupyter Notebook. RAM: 8GB, GPU: NVIDIA GTX1650.
2. Entorno virtual 1: plataforma de desarrollo Colab de Google [27]. RAM: 12 GB, GPU: NVIDIA T4.
3. Entorno virtual 2: plataforma de desarrollo Colab Pro de Google. RAM: 12 GB, GPU: NVIDIA A100SXM.

La implementación de aplicaciones con TensorFlow-GPU y CUDA, requiere que las bibliotecas de *Python* a usarse, se encuentren en versiones compatibles con éstas, así como la versión de este lenguaje. Las versiones de las bibliotecas implicadas en el desarrollo del segmentador para los distintos entornos de desarrollo se muestran en el Anexo.

GPUs usadas			
Modelo	Gama	Cantidad de CUDA Cores	VRAM
GTX1650	Baja	896	4GB
Tesla T4	Media	2560	16 GB
A100SXM	Alta	6912	40 GB

Tabla 4.1: *Especificaciones de GPUs usadas.*

4.2. Conjunto de volúmenes

Los volúmenes utilizados para desarrollo, entrenamiento y pruebas del sistema fueron obtenidos de *BrainWeb: 20 Anatomical Models of 20 Normal Brains*, en la Figura 4.1 se muestran 20 rebanadas con el mismo índice de orden de los distintos volúmenes utilizados [28]. Esta base de datos proporciona volúmenes de resonancia magnética cerebral simulados en la modalidad T1 con secuencia SFLASH con TR = 22 ms, TE = 9.2 ms, ángulo de giro = 30 grados y tamaño de vóxel isotrópico de 1 mm., que resultan en volúmenes de dimensión 181x256x256, además de disponer de los correspondientes volúmenes segmentados, como es necesario para la clasificación en métodos de aprendizaje profundo supervisados.
El uso de CUDA y Tensorflow-GPU implican ciertas restricciones, una de ellas es el uso de datos con tamaños múltiplos de 32, ya que los bloques en la estructura de la GPU se componen de 32 hilos (mostrado en la Figura 1.13), por lo tanto, la asignación de un hilo a cierto proceso sobre un operando implica tomar en cuenta esta medida. Por lo tanto, los volúmenes se recortan al tamaño 160x256x256, eliminando los cortes extremos que no contienen tejido relevante para su clasificación, en la mayoría de dichos cortes no hay presencia de tejido.

4.3. Estructura de procesamiento

El sistema de segmentación se compone de dos bloques concatenados, el filtrado espacial y la segmentación con red profunda, la Figura 4.2 muestra el diagrama metodológico para el desarrollo de este trabajo. El volumen de resonancia magnética cerebral ingresa sin transformaciones a los datos, a excepción de la eliminación de los cortes como se menciona anteriormente. El primer bloque se refiere al procesamiento de los volúmenes con el filtrado espacial propuesto, utilizando el algoritmo del corrimiento de media con participación del mapa de confianza de bordes, esto con el objetivo en que la entrada del

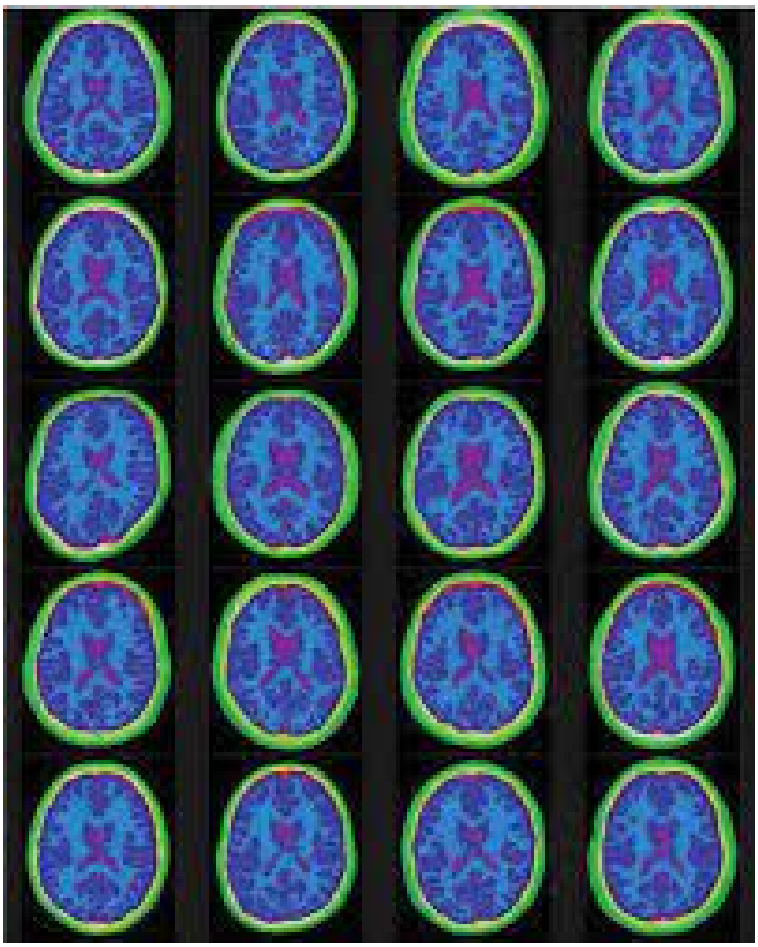


Figura 4.1: *Rebanadas de los 20 volúmenes de RM cerebral utilizados [28].*

sistema de aprendizaje profundo esté en condiciones óptimas para el procedimiento de clasificación. El segundo bloque consta de una red profunda convolucional de arquitectura U-NET 3D, que se encarga de hacer la clasificación de tejidos de los volúmenes, a la salida de ésta resultan 5 mapas de probabilidad posterior que terminan por formar el volumen segmentado final. En el diagrama mostrado en la Figura 4.2 se aprecia la influencia de la GPU en la aceleración de este proceso, en el corrimiento de media se optó por paralelizar las operaciones correspondientes a la búsqueda de medias locales utilizando CUDA, mientras que la red profunda tiene soporte en Tensorflow-GPU al minimizar el tiempo de entrenamiento y posterior clasificación utilizando una mayor cantidad de núcleos en comparación a su versión en CPU. Es importante destacar que el mapa de confianza de bordes es desarrollado y aplicado sobre un entorno sin GPU, la razón de esta decisión se explica más adelante.

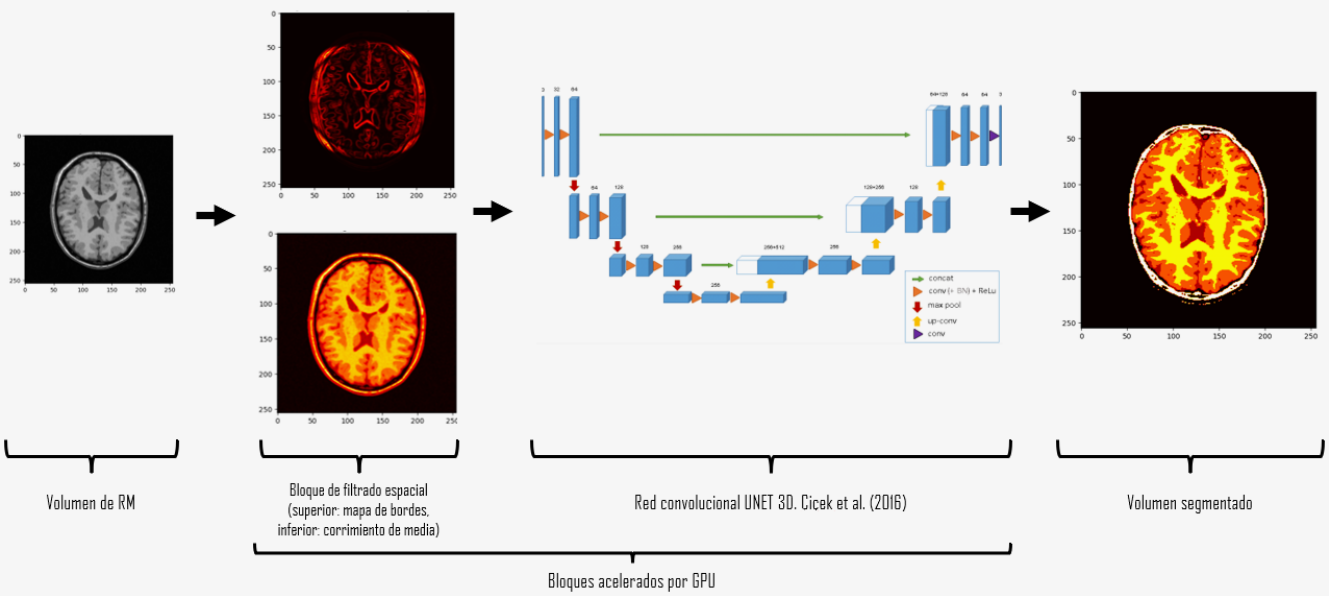


Figura 4.2: *Diagrama general de los métodos realizados.*

4.4. Filtrado espacial

El bloque de filtrado espacial previo a la red profunda tiene como intención eliminar ruido notorio y artefactos de los volúmenes, lo que permite que el volumen de entrada de la red facilite el entrenamiento a la misma, ya que contiene menos vóxeles con distorsiones sobre zonas homogéneas. Este filtro se compone del trabajo conjunto de un mapa de confianza de bordes y el algoritmo de corrimiento de media, dos procedimientos conocidos en el campo del procesamiento digital de imágenes.

4.4.1. Mapa de confianza de bordes

El objetivo del mapa de confianza de bordes es identificar cuáles son los vóxeles en los que aparecen transiciones abruptas de la escala de grises, lo cual se traduce en bordes o fronteras entre diferentes tipos de tejido humano a identificar. El primer paso fue aplicar filtros de Sobel con el objetivo de encontrar los gradientes de cada vóxel del volumen, esto da un panorama acerca de las zonas de alta transición de niveles de gris en las direcciones de altura, anchura y profundidad (dx, dy y dz). La Figura 4.3 expone el resultado de aplicar el operador Sobel, donde en cada vóxel se genera la aproximación al gradiente de la función de intensidad del volumen, los primeros dos planos mostrados están orientados de la misma forma pero el resultado del filtro Sobel en la dirección z se muestra en un plano distinto a los otros para mostrar su efecto.

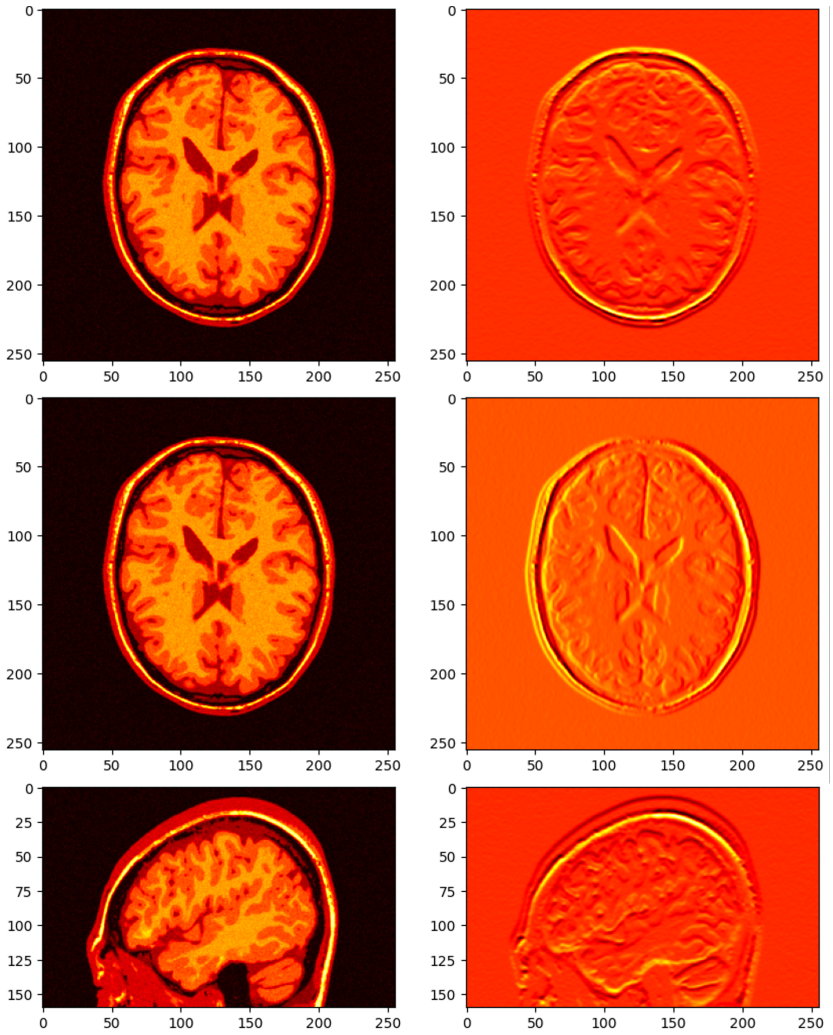


Figura 4.3: *Rebanada del resultado de aplicar el filtro de Sobel a un volumen en las distintas direcciones: par superior sobre eje sagital (x), par de enmedio sobre eje coronal (y), par inferior sobre eje axial (z). El plano de las rebanadas mostradas no corresponde a las orientaciones del filtro aplicado, se muestran en orientaciones que facilitan su visualización*

El siguiente paso para generar el mapa de bordes se compone de un algoritmo cuyo fin es calcular la correlación de ventanas 3D de tamaño definido del volumen original con modelos de borde ideal orientados en la dirección del gradiente. Este proceso se asimila a la aplicación de un kernel de filtrado de una imagen o volumen, ya que recorre todos los puntos de un elemento de forma ordenada, generando un nuevo punto en un elemento del mismo tamaño del original después de realizar alguna transformación o cálculo. El mapa de bordes será un nuevo volumen cuyos valores corresponden a la correlación calculada, por lo tanto, un vóxel del mapa con valor 1.0 corresponde a un vóxel de la imagen original para el que se tiene seguridad que es un borde. En tanto que un valor 0.0 en el mapa, implica que el vóxel en cuestión habita en una región homogénea de un tejido.
De forma ordenada se siguen los siguientes pasos:

1. Para cada voxel del volumen original, se calcula su vector gradiente a partir de los volúmenes de salida de los filtros Sobel (dx, dy y dz), tomando en cuenta una ventana 3D de tamaño definido.

2. El vector gradiente se utiliza para construir un modelo de borde ideal, el resultado es una separación espacial entre el mínimo y el máximo valor de gris y en donde el mismo vector de gradiente orienta la máxima transición de nivel de gris posible. Este proceso evalúa la dirección del vector formado del centro de la ventana con los voxeles restantes de la misma ventana, esto mediante productos vectoriales. Se muestra un ejemplo de borde ideal para un vóxel arbitrario en la figura 4.4.

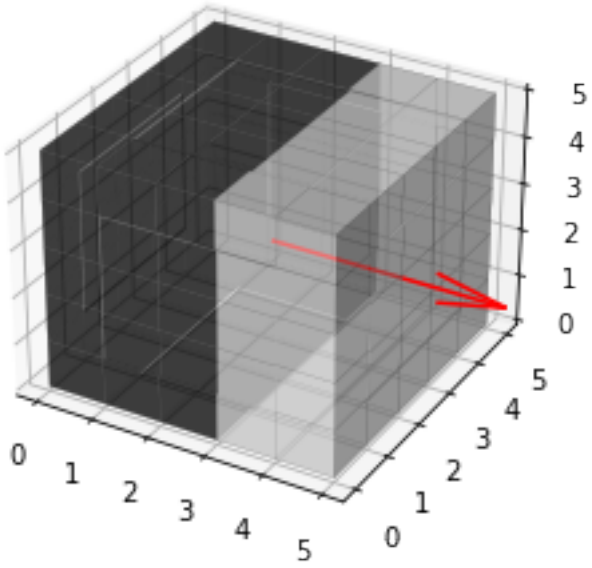


Figura 4.4: *Modelo 3D de borde ideal para un vóxel arbitrario. La flecha indica la dirección del vector gradiente.*

3. Se calcula el valor de correlación entre la ventana 3D del volumen original centrada en el vóxel en cuestión y el modelo de borde ideal generado, para después asignar el valor a un nuevo vóxel del nuevo volumen del resultado, en la misma posición que el centro de la ventana inicial.

4. Para tener un mapa con coeficientes de correlación más estrictos, se multiplica el mapa de confianza de bordes obtenido por la magnitud del gradiente del volumen original, el resultado de este cálculo es un mapa de confianza que establece valores de correlación con mayor diferenciación, lo que permite distinguir con mayor facilidad cuál vóxel es borde y cuál no.

El resultado de este procedimiento es un volumen del mismo tamaño del original, el cual es usado para el resto del filtrado espacial, posteriormente se explica la contribución de este algoritmo.

Como se menciona en la sección 4.3, a diferencia de los demás procedimientos, el mapa de bordes no cuenta con aceleración de GPU, debido a que la VRAM de este entorno es un recurso finito y limitado, además de contar con el problema considerable de tener nula liberación de sus recursos después de su uso, esto plantea la necesidad de evitar si-

tuaciones de riesgo que pudieran llenar la memoria y detener la ejecución del programa. La medida usada para el cuidado de los recursos fue destinar el uso de la GPU a procesamientos que necesiten, en nuestro criterio, una reducción significativa de tiempo de ejecución, por lo que la decisión final fue usar la aceleración en el corrimiento de media y el entrenamiento, pruebas y evaluación de red convolucional. En el caso del mapa de confianza de bordes, se estima un tiempo de ejecución considerablemente menor que en los demás procedimientos.

4.4.2. Corrimiento de media

El corrimiento de media es una técnica no paramétrica del dominio del aprendizaje maquinal utilizada en el análisis de un espacio de características con densidades de probabilidad multivariadas y multimodales. El uso de esta técnica se basa en la aplicación recursiva de un método para encontrar el punto estacionario más cercano de la función de densidad, esto resulta en la identificación de las modas de la misma función.

El primer paso fue llevar al volumen a un espacio 4D generado con las coordenadas espaciales 3D (altura, anchura, profundidad) y la intensidad de gris de cada vóxel. Para cada punto de este espacio se evalúa cuales otros puntos se encuentran dentro de una vecindad cercana, para esto se establece un radio espacial de longitud definida que logra determinar qué puntos se encuentran a una distancia menor a la del radio, la elección de este hiperparámetro es una tarea con muchas implicaciones, ya que su longitud define la hiperesfera que engloba los puntos cercanos y esto tiene consecuencias en el comportamiento del vector gradiente encargado de la trayectoria de cada punto hacia su moda local, por la ecuación 1.3 se puede concluir que en zonas en las cuales hay pocos puntos, el vector gradiente tiene una magnitud grande, por lo tanto, avanza en trayectorias grandes, esto beneficia el curso del vector gradiente, ya que estas zonas con puntos limitados son de poco interés para el análisis del espacio de características, su consecuencia es que en estos casos el vector gradiente se desplaza con mayor proporción en la búsqueda de su moda local. Por el contrario, cuando la hiperesfera se sitúa en una zona del espacio con alta densidad de puntos, la magnitud del vector es pequeña y esto provoca el avance en pequeñas trayectorias, haciendo un análisis mas refinado de las zonas de interés con mayor población de características.

Continuando, una vez identificados los puntos, se realiza un promedio espacial y su

resultante se convierte en el nuevo centro de la vecindad. Hasta este punto, todo el proceso mencionado se lleva a cabo en cada punto de la nube de datos hasta terminar con la primera iteración, lo que supone que la complejidad computacional del procedimiento está íntimamente relacionada a la cantidad de datos, ya que al menos para el caso del cálculo de distancia de un punto vs otro se utilizan dos operaciones vectoriales para definir si se cumple el criterio de vecindad, esto hace que el cálculo de la distancia de todos vs todos resulte en un número de operaciones definido por n_o en: $n_o = 2n(n-1)$, siendo n el número de puntos.

Por último, al ser un método iterativo, el proceso se repite hasta cumplir con el criterio de convergencia establecido previamente. En este trabajo dicho criterio fue repetir el proceso en una cantidad finita de iteraciones, por lo que se efectuaron los ciclos de actualización de media 5 veces, esto termina por tener una cantidad de modas notablemente menor a la del inicio. Una vez terminado el proceso y ya con el espacio inicial transformado en uno poblado de modas, se asigna el valor de gris de éstas a los valores de gris del espacio 4D inicial según su trayectoria de convergencia, para después reestructurar ordenadamente los puntos y volver a formar el volumen.

El resultado de este algoritmo es un volumen con menos modas, esto provoca que vóxeles con niveles de gris de alta variabilidad con sus vecinos se transformen en vóxeles con valores más parecidos en la adyacencia, por lo tanto, se entregan volúmenes más suaves, limpios y diferenciados entre presuntos tejidos por clasificar posteriormente.

4.4.3. Corrimiento de media acelerado con CUDA

El corrimiento de media es un algoritmo conocido por su efectividad así como su robusta ejecución, teniendo un tiempo largo de procesamiento dependiente de la cantidad de puntos (datos) en el espacio, por lo que se plantea el uso de un método de aceleración usando GPU. CUDA es una plataforma de desarrollo de computación paralela en sistemas con tarjetas gráficas NVIDIA, el aprovechamiento de los recursos de esta plataforma permite usar instrucciones sobre distintos operandos al mismo tiempo, por lo que es factible implementar el corrimiento de media en esta plataforma, aunado a esto, Python proporciona envoltorios Cython/Python para el controlador CUDA y las API de tiempo de ejecución, así que es posible aprovechar la computación paralela GPU usando Python para lograr resultados y precisión más rápidos.

El abordaje a la implementación en CUDA es que se necesita realizar en cada punto de un espacio 4D, distintas operaciones de cáracter matricial y vectorial así como númericas,

para encontrar, después de n iteraciones, las modas del espacio. Sabiendo las limitaciones de CUDA-Python en el uso de ciertas bibliotecas que facilitan operaciones matriciales, se diseña un programa que calcule de forma explícita la distancia de un punto del espacio vs todos los puntos del espacio mediante la fórmula en la ecuación 4.1, donde d^2 es la distancia al cuadrado, t_d es la matriz que contiene las 4 coordenadas de todos los puntos y x_i el vector del punto actual. Ya que la distancia cuadrada es proporcional a la distancia, sirve como medio de comparación. Lo siguiente es encontrar los puntos de los cuales la distancia cuadrada al punto actual es menor a una distancia establecida, siendo esta cantidad la correspondiente al radio de la hiperesfera mencionada antes. Este último procedimiento implica una lenta comparación entre datos númericos, lo que también suma tiempo de ejecución.

$$d^2 = |t_d - x_i|^2 = t_d \cdot t_d + x_i \cdot x_i^T - 2t_d \cdot x_i^T \tag{4.1}$$

Como ha sido mencionado, para este proyecto se utilizan volúmenes de tamaño 256 x 256 x 160, es decir, se tienen más de 10 millones de puntos en la nube de datos del espacio 4D generado, tomando en cuenta la metodología, para cada punto se realiza el cálculo de más de 10 millones de conjuntos de operaciones y sentencias, que en consecuencia se realizan más de $10 * 10^{12}$ conjuntos de cálculos para un volumen, entre ellos el cálculo de distancia, la comparativa entre distancias de puntos, el promedio espacial, etc. Esto anticipa un enorme reto para una CPU.
Por lo tanto, se diseña un programa para que el entorno CUDA paralelice este proceso, tomando un punto del espacio y realizando todas las operaciones pertinentes de este punto en un proceso que se sitúa en un sólo hilo de procesamiento situado en un bloque de estrucutura de la GPU, esto es, todas las operaciones de cada punto del volumen se llevarán a cabo de manera paralela, logrando una gran ganancia de tiempo de procesamiento del corrimiento de media. En el Anexo de este escrito, se muestran los códigos del mapa de confianza de bordes, el corrimiento de media acelerado, la implementación y arquitectura de la red convolucional y por último, la validación cruzada.

Es necesario aclarar que las GPU usadas no tienen tantos hilos de procesamiento como vóxeles en un volumen, por lo cual CUDA adapta su ejecución automáticamente para trabajar por bloques de instrucciones pero con coordinación en sus procesos, esto implica que los hilos se sincronicen de forma ímplicita para evitar cruces de información que afecten el desempeño entre hilos con información procesada fuera de orden.

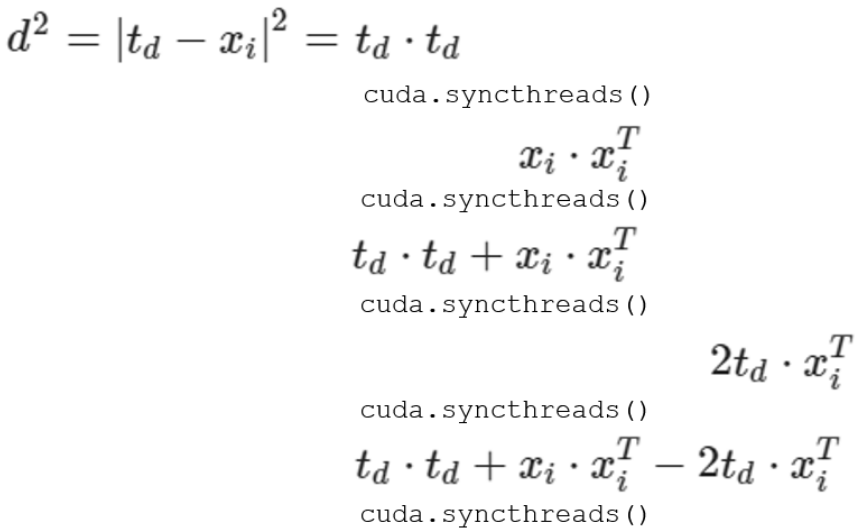


Figura 4.5: *Diagrama de sincronización de hilos en el cálculo de la distancia entre puntos.*

El gran riesgo de tratar con datos de gran volumen implica medidas en el curso de la ejecución, tal como la mencionada sincronización de los hilos, en la implementación realizada se aplica esta medida después de cada operación matricial de gran escala. La Figura 4.5 ilustra un diagrama del orden de sincronización de hilos en el cálculo de la distancia entre puntos, necesaria en el corrimiento de media. El comando *cuda.syncthreads()* es la instrucción aplicada para que las tareas asignadas en cada hilo culminen completamente de manera paralela y coordinada, esto para continuar con la siguiente instrucción evitando que existan problemas de sincronización en los que información necesaria para cálculos de un hilo se encuentre desactualizada por el procesamiento hecho en otros hilos. Aunque el uso de CUDA consta de restricciones, la comparación del rendimiento de pocos núcleos en una CPU contra la cantidad de éstos en una GPU es ampliamente superior, en secciones posteriores se aborda esta comparativa.

4.4.4. Filtro espacial completo, corrimiento de media acelerado y mapa de confianza de bordes

Para formar el filtro espacial planteado, es necesario que el mapa de confianza intervenga en el cálculo de las nuevas medias para el algoritmo de Mean Shift. Esto ocurre cuando los puntos dentro de la vecindad establecida se promedian, el cambio radica en que es un promedio ponderado por el coeficiente determinado por el mapa de bordes, tal como se muestra en la ecuación 4.2, donde S_{WM} es el promedio espacial ponderado, S_h

la hiperesfera, n_x la cantidad de puntos dentro de la hiperesfera, X_i cada punto dentro de la hiperesfera y B_i el coeficiente de confianza de borde respectivo a cada punto.
Con esto se logra que los vóxeles respectivos de este espacio situados en los bordes, se aproximen a un valor despreciable, el cual favorece a que las modas no tengan gran influencia de valores muy dispersos como lo son puntos cerca del límite de la hiperesfera planteada, es decir, las altas transiciones de gris en un volumen (bordes) y por lo tanto, las modas se sitúen en zonas cercanas a la homogeneidad. Además, este hecho tiene como ventaja que el vector gradiente reduzca las iteraciones necesarias para llegar a su trayectoria, debido a que los puntos pertenecientes a bordes desviarían su dirección hacia la moda local al contaminar el cúmulo de puntos contenidos dentro de la hiperesfera.

$$S_{WM} = \frac{\sum_{x_i \in S_h(x)} X_i(1 - B_i)}{n_x} \tag{4.2}$$

4.5. Red neuronal profunda U-NET 3D

La red U-NET 3D es un modelo de red neuronal convolucional dedicado comúnmente a tareas de visión artificial y específicamente para resolver problemas de segmentación semántica.

Básicamente, es una red con una estructura de contracción y otra de expansión, que utiliza capas sucesivas de convolución y *pooling* en la primera parte, con capas de sobremuestreo de la segunda parte que logran aumentar la resolución en la salida de la red. Las capas de contracción se encargan de extraer características de alta resolución, para después conectarse con las capas de interpolación para la salida, así, en las capas de expansión se dispone de un mayor número de mapas de características, que le permiten mantener la información de localización.

La arquitectura original presentada por [16] y mostrada en la Figura 1.11, presenta bancos de 32, 64, 128, 256, 256, 128 y 64 filtros por cada nivel de la estructura. Para este trabajo la arquitectura se definió con bancos de 32, 64, 128, 256, 512, 256, 128, 64, 32 filtros. La modificación planteada, mostrada en la Figura 4.6, se realizó con la intención de aprovechar las ventajas de cada una de las dos partes de la arquitectura, es decir, extrayendo más características de alta resolución y recabando mayor información de la localización. Con conciencia en que el esfuerzo computacional que exige aumentar los bancos de filtros para cada nivel y capa, aumenta considerablemente. La Figura 4.6 muestra en cada operación el número de filtros aplicados, tal como se menciona anteriormente, además

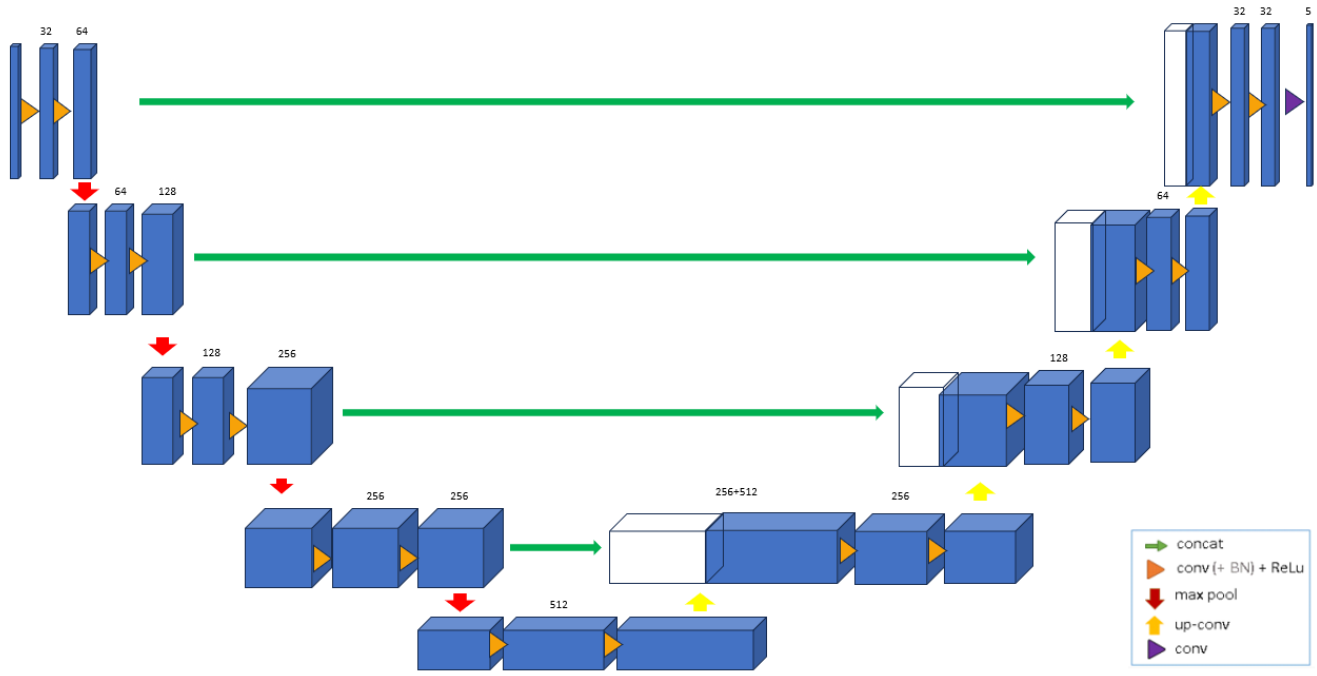


Figura 4.6: *Arquitectura de red U-NET 3D modificada y usada en este proyecto.*

de exponer la concatenación de salidas de la parte de contracción con la parte de expansión según el nivel.

Como puede observarse en la Figura 4.6, la última convolución de la red entrega 5 canales, debido a que se planea que el resultado final de la misma sea la clasificación de 5 mapas probabilísticos 3D, uno por cada etiqueta clasificada, donde es posible identificar, con la probabilidad posterior dada por el Teorema de Bayes, a qué tipo de tejido es más probable que pertenezca cada vóxel del volúmen.

Por último, se genera el volumen etiquetado por tejidos a partir de los mapas probabilísticos generados a la salida de la red. En cada uno de estos 5 volúmenes de probabilidad posterior, correspondientes a cada clase, se conoce la probabilidad de que cada vóxel pertenezca a una etiqueta. La decisión se toma al comparar el mismo vóxel en todos los mapas y encontrar cuál es el mapa con la mayor probabilidad de correspondencia, repitiendo este proceso en cada vóxel. Esto es posible con la aplicación de la función *arg max*, la cual encuentra el argumento o argumentos (*arg*) para la función objetivo que devuelve el valor máximo (*max*) de esta última.

4.5.1. Entrenamiento

El entrenamiento de las redes neuronales es el proceso más lento y complicado computacionalmente de su implementación, debido a distintos factores como el modelo propuesto, el tamaño de la entrada, los hiperparámetros establecidos, el volumen de datos, la cantidad de épocas por cumplir, etc. Por ello, para este proyecto que plantea usar una red profunda de alta complejidad, un volumen completo de RM cerebral de 160x256x256 vóxeles como entrada y 20 volúmenes como el conjunto de desarrollo, se ha planteado su operación utilizando el aceleramiento por GPU. Con el modelo construido, el proceso de entrenamiento optimiza la entropía cruzada de las categorías (función de pérdida) utilizando el algoritmo Adam [29].

Como se ha revisado en otras secciones, la cantidad de datos a entrenar es inmensa, además de tener la gran dificultad de la limpieza de la VRAM en las GPUs, lo cual impide que el proceso de entrenamiento termine, ya que los párametros internos optimizados durante el proceso se almacenan en cada época, esta acumulación termina por exceder el límite de memoria disponible, lo que interrumpe la ejecución de forma abrupta sin posibilidades de recuperar los parámetros de la última época. Esto sumado a que se necesita una gran cantidad de memoria VRAM utilizada en un modelo de esta especie por la gran cantidad de datos de entrada como lo son 20 volúmenes de RM, provoca que sea necesario adquirir tarjetas gráficas de amplia VRAM (muy costosas) o encontrar un método que permita entrenar el modelo con el recurso disponible. Cabe destacar que a pesar de contar en cierto momento con la tarjeta A100SXM de NVIDIA con VRAM suficiente para vencer esta problemática, se establece el desafío de lograr el entrenamiento de la red en cualquier GPU disponible, es decir, usando las GPUs NVIDIA GTX1650 y Tesla T4 con la misma implementación sin importar el entorno usado.
Po lo tanto, para este proyecto se plantea un entrenamiento a trozos, es decir, dividir ordenadamente los volúmenes de entrenamiento y prueba en 5 partes, para entrenar la red con la primera parte, guardar el modelo completo entrenado en formato *.h5*, cargarlo en una nueva ejecución del kernel y volverlo a entrenar con la segunda parte de la partición. Este proceso se repite de forma correcta y ordenada para así terminar con el entrenamiento de la quinta parte. El objetivo es que en cada etapa de entrenamiento, los coeficientes de los kernels internos de la red se adapten a volúmenes de la misma naturaleza pero con morfologías distintas. En cada iteración de este proceso se recaban métricas de evaluación, como el valor de la precisión y las curvas de aprendizaje, con motivo de establecer un calificador del segmentador final.

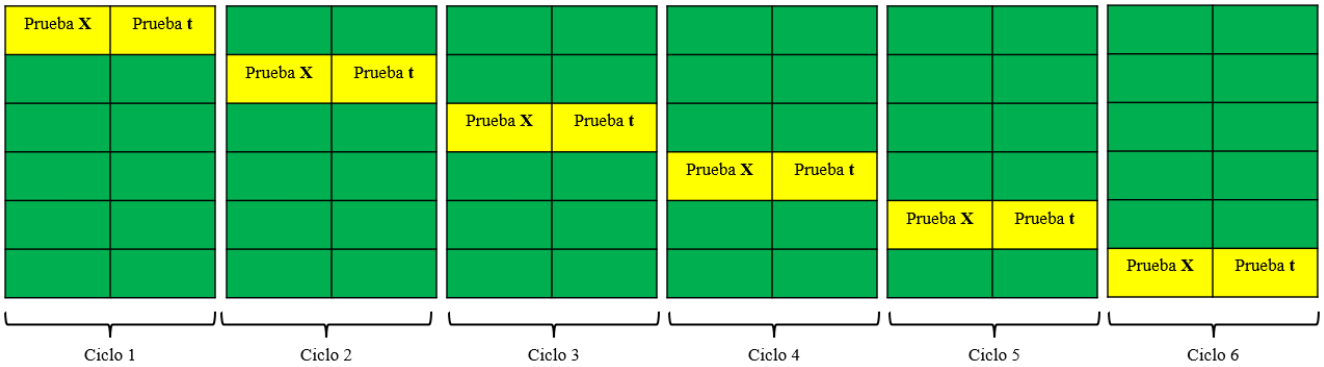


Figura 4.7: *Diagrama de la validación cruzada de 6 vías. En color amarillo de cada ciclo se muestra el lote que funge como conjunto de prueba (datos de entrada X y correspondientes etiquetados t), mientras que en color verde se muestran los lotes que fungen como conjunto de entrenamiento.*

4.5.2. Validación cruzada y conjunto de prueba

El algoritmo de validación cruzada es una técnica usada en el aprendizaje automático para evaluar la variabilidad de un conjunto de datos y la confiabilidad de cualquier modelo entrenado con ellos. Comienza con la mezcla aleatoria de los datos y con la definición del parámetro k, el cual define el número de particiones o vías del set de datos, así como el número de iteraciones de entrenamiento y prueba que se usan al construir el modelo.
Una vez definido el valor de k, se inicia cada proceso de entrenamiento y pruebas. Así que por un total de k iteraciones se repiten los siguientes pasos:

1. Se toma una de las k particiones y se aparta del modelo.
2. Se toman las $k-1$ particiones restantes y con ellas se entrena el modelo.
3. Se almacenan las curvas de precisión y error arrojadas durante el entrenamiento para su análisis al término del proceso.
4. Una vez entrenado el modelo, se aplica directamente al set reservado, se evalúa y las métricas resultantes se almacenan para posterior análisis.
5. Se repiten los pasos anteriores pero intercalando la participación del set de pruebas con cada uno de los sets. La Figura 4.7 da una mejor apreciación de la mecánica de intercambio de roles en cada vía.

Ya terminadas las 6 iteraciones, se tienen k medidas de desempeño para los sets de entrenamiento y validación usados en cada iteración. Por lo que el desempeño final del modelo es el promedio de los desempeños recabados.

Se utilizan las funciones predeterminadas de la biblioteca de *Python*, *Keras* [30], para observar y evaluar el funcionamiento de la red.
Los 20 volúmenes son divididos de forma alternada y aleatoria en 90% para el uso en validación cruzada y el 10% para pruebas y evaluación. Posteriormente, el apartado de volúmenes de validación cruzada es nuevamente dividido en 6 lotes, 5 de ellos fungen como entrenamiento y uno como prueba y evaluación, en cada ciclo. Para calificar la calidad del clasificador diseñado, se obtienen las curvas de aprendizaje (error y precisión), intervalo de confianza, métricas de evaluación y las matrices de confusión en la prueba de validación cruzada de 6 vías.

Como evaluación del modelo final de operación, se entrena la red con el conjunto de entrenamiento completo, es decir, el 90% de volúmenes utilizados para la evaluación con la validación cruza. Para posteriormente, aplicar el modelo al conjunto de prueba, es decir, el 10% de los volúmenes separado incialmente. Por último, se calculan los valores de evaluación de la clasificación y se reportan, afirmando así el compromiso de calificar su desempeño al utilizar datos desconocidos en la entrada del sistema.

Sobre el código del cálculo del mapa de confianza de bordes situado en el Anexo. Se muestra la creación de un nuevo volumen que guardará el resultado del procesamiento, además, se crea en cada caso, la ventana volumétrica que define el borde ideal según la ventana actual, para después realizar el cálculo de correlación entre ellas y determinar el nuevo valor del mapa de bordes. El resultado final es el producto del mapa de bordes calculado con la magnitud de los gradientes de Sobel previamente calculados.

Respecto al corrimiento de media acelerado. El código expuesto en el Anexo, compacta su implementación con el afán de reducir en extremo la cantidad de variables a utilizar debido al cuidado de la VRAM de la GPU. La metodología a seguir es transferir previamente los datos a paralelizar al dominio de la memoria de la GPU, procesarlos y finalmente regresarlos al dominio de la CPU para seguir con el procesamiento. Se comienza el código definiendo 4 variables que se encargan de almacenar resultados escalares, ya que CUDA no permite utilizar funciones para hacer cálculos vectoriales directos y se necesitan para calcular operaciones de este tipo tal como la multiplicación matricial y el promedio espacial. Se destaca que la especificación del número de hilos a usarse, así como de bloques, es un factor poco relevante en el procesamiento.

El código relacionado con la red convolucional, comienza con la implementación de

la arquitectura, empatando con las especificaciones mencionadas anteriormente en la figura 4.6. Por último, se especifica el inicio del entrenamiento ya con los parámetos establecidos como el optimizador Adam y la entropía cruzada categórica como función de pérdida.

Por último, se muestra en el Anexo, el código respectivo a la validación cruzada implementada en la tarjeta gráfica de alta gama. Primeramente se específica el número de vías a usarse y se sincroniza con las particiones del conjunto entrenamiento. En cada iteración se evalúa la precisión para almacenarse, además de rescatar las curvas de aprendizaje, las cuales funcionan como pruebas del rendimiento de este sistema.

Capítulo 5

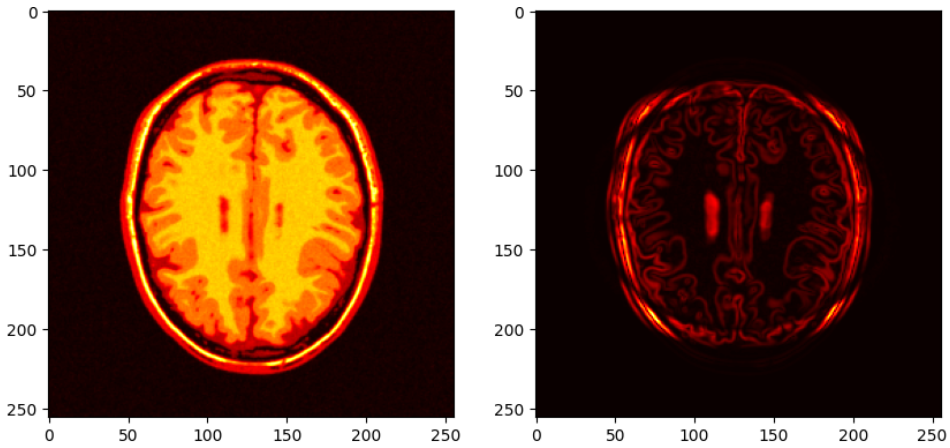
Resultados

En la presente sección, los resultados por mostrarse se dividen en dos partes. La primera de ellas se refiere a los resultados cualitativos de cada proceso de la metodología, se pretende mostrar la ejecución del sistema en condiciones reales de operación, donde no sería posible tener un medio de contraste que permita realizar una comparación y posterior evaluación númerica. Mientras que los resultados cuantitativos completan el conjunto de soluciones mostradas previamente y permiten un análisis de la calidad de los resultados.

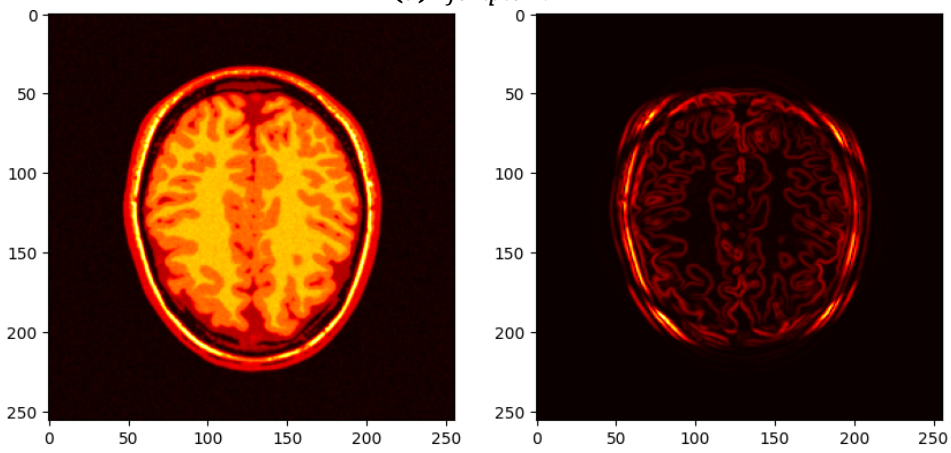
5.1. Resultados cualitativos

5.1.1. Mapa de confianza de bordes

Como primer resultado, la Figura 5.1 muestra dos cortes correspondientes a la aplicación del mapa de confianza de bordes a un volumen arbitrario. Se observa que las fronteras entre estructuras identificadas en volúmenes son definidas con valores de gris muy distintos de zonas homogéneas, esto permite con mayor facilidad la diferenciación entre la clasificación de cada voxel.



(a) *Ejemplo 1.*



(b) *Ejemplo 2.*

Figura 5.1: *Ejemplos de cortes del resultado del mapa de confianza de bordes de un volumen. Izquierda: rebanada de volumen de entrada. Derecha: rebanada de volumen de salida del mapa de confianza de bordes.*

5.1.2. Filtrado espacial

Una vez que se genera el mapa de confianza de bordes del volumen de entrada, se aplica el método del corrimiento de media (ponderado con el mapa de bordes) para filtrar los datos de entrada. Las Figuras 5.2 y 5.3 se tratan de ejemplos de cortes de un volumen arbitrario antes y después de aplicar el filtrado espacial, en ellos se resalta el efecto del filtro espacial sobre un volumen.

A primera vista, es posible observar que el filtro aporta mayor suavidad intra-tejido y mayor contraste inter-tejidos, esta observación depende de la habilidad visual del usuario, por lo que además de presentar un corte del volumen de entrada y su correspondiente corte filtrado en un mapa de color de escalas de grises, se presenta su análogo

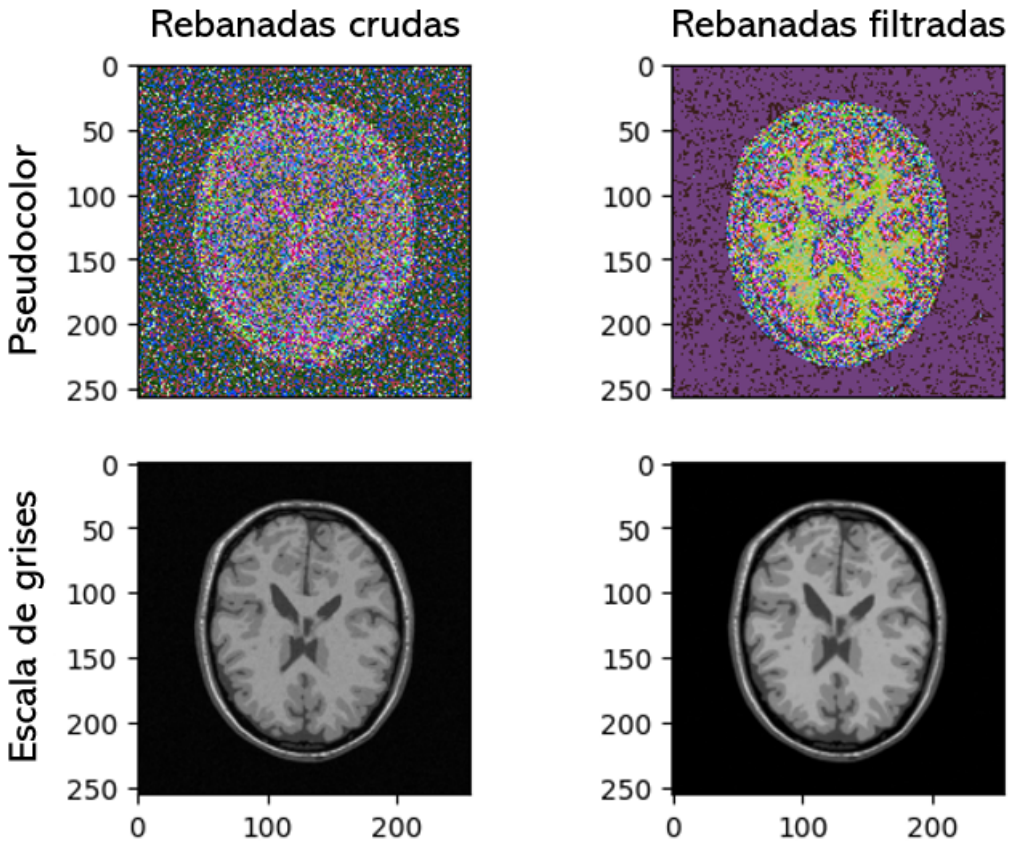


Figura 5.2: *Rebanada ejemplo del resultado del filtro espacial sobre un volumen. Cuadrícula de combinaciones: eje vertical respectivo al mapa de color, pseudocolor aleatorio o escala de grises. Eje horizontal respectivo a tipo de rebanada, cruda o filtrada respectivamente.*

en pseudocolor de rango dinámico aleatorio. El objetivo de estas imágenes es apreciar la identificación de las modas locales encontradas durante el proceso del corrimiento de media, donde es posible hacer una diferenciación de menor escala entre tejidos que posteriormente se logran clasificar con la red neuronal, ya que si el rango dinámico del volumen no sigue una progresión de color continua, es posible apreciar la diferencia de transiciones incluso en cambios de pequeña magnitud, tal como se plantea cuando los puntos cambien su valor en dirección de sus modas locales, aún cuando el cambio sea relativamente pequeño.

5.1.3. Red convolucional

Después de filtrar los volúmenes con el método de filtrado propuesto, son ingresados a la red neuronal entrenada. Según lo mencionado en secciones pasadas, la entrada de la

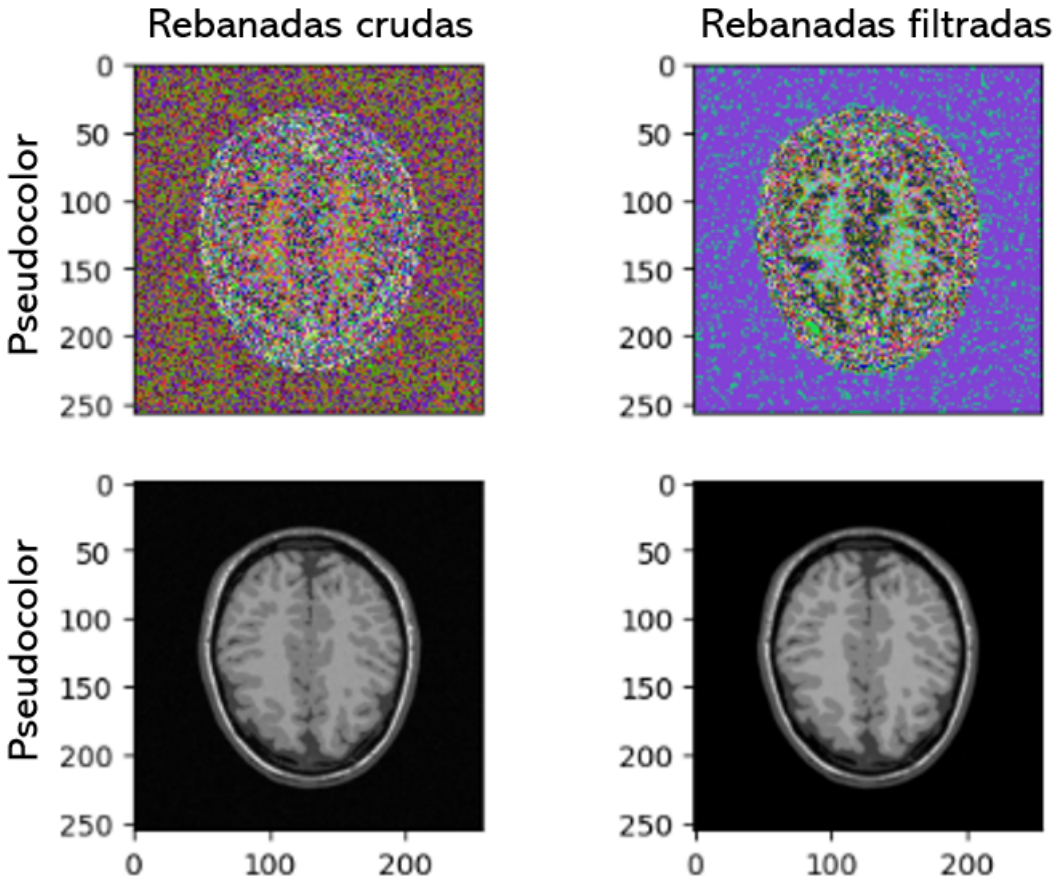
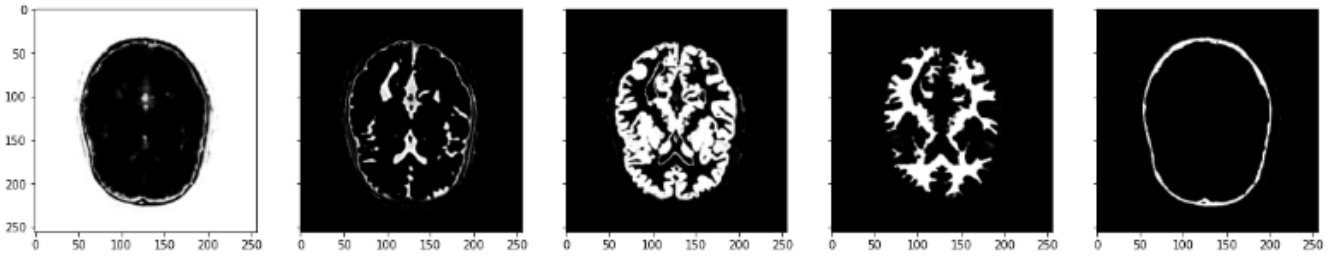


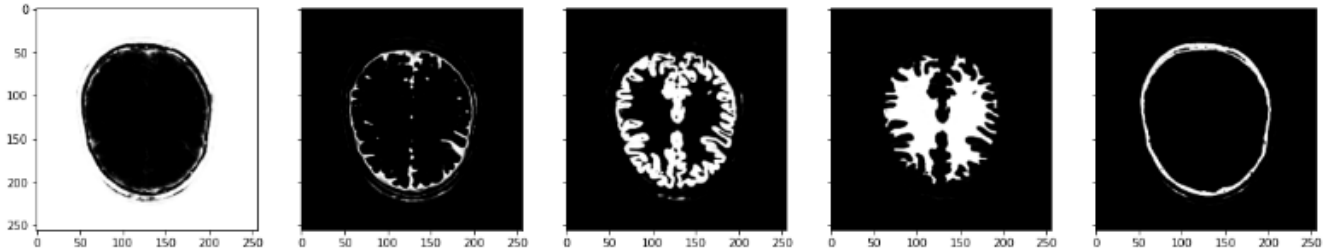
Figura 5.3: *Rebanada ejemplo del resultado del filtro espacial sobre un volumen. Cuadrícula de combinaciones: eje vertical respectivo al mapa de color, pseudocolor aleatorio o escala de grises. Eje horizontal respectivo a tipo de rebanada, cruda o filtrada respectivamente.*

red es un volumen de RM de 160x256x256 vóxeles y la salida se compone de 5 volúmenes de 160x256x256, correspondientes a 5 mapas de probabilidad posterior que denotan en un índice con rango de 0 a 1, cuál es la probabilidad de que cada voxel pertenezca a uno de los 5 tejidos (materia gris, materia blanca, líquido cefalorraquídeo, cráneo y fondo) predefinidos.
La Figura 5.4 expone dos ejemplos de mapas de probabilidad posterior como resultado de la aplicación de la red convolucional.

El último paso para culminar la segmentación de los volúmenes es la transformación de los 5 mapas de probabilidad posterior a un volumen etiquetado como resultado de la segmentación. En la Figura 5.5 se muestran dos ejemplos de la aplicación de la función *arg max* sobre los mapas de probabilidad, lo que resulta en volúmenes etiquetados con 5 distintos valores en su rango. Además, la Figura 5.5 muestra la comparación visual entre



(a) *Ejemplo 1.*



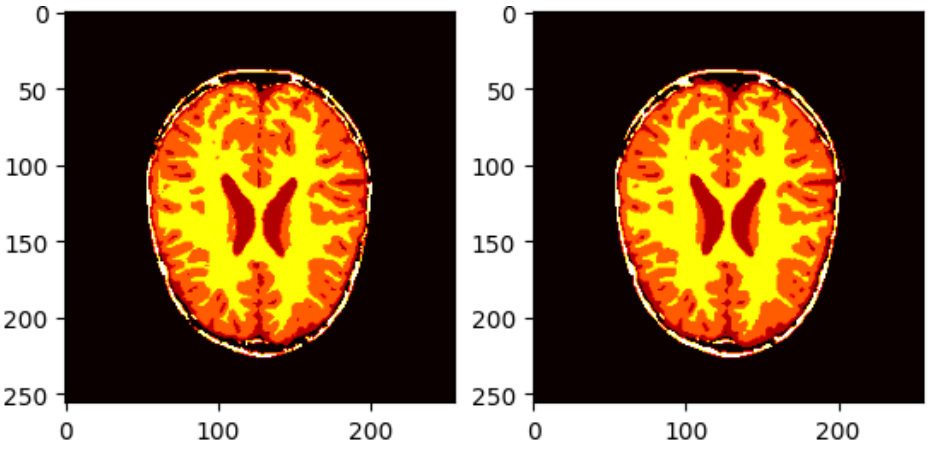
(b) *Ejemplo 2.*

Figura 5.4: *Ejemplos de cortes de mapas de probabilidad posterior como resultados de la clasificación de un volumen utilizando la red neuronal. Izquierda: fondo, centro izquierda: líquido cefalorraquídeo, centro: materia gris, centro derecha: materia blanca, derecha: cráneo. Imágenes en escala de grises con rango dinámico de 0 a 1 en relación de la probabilidad de pertenecer a una clase determinada.*

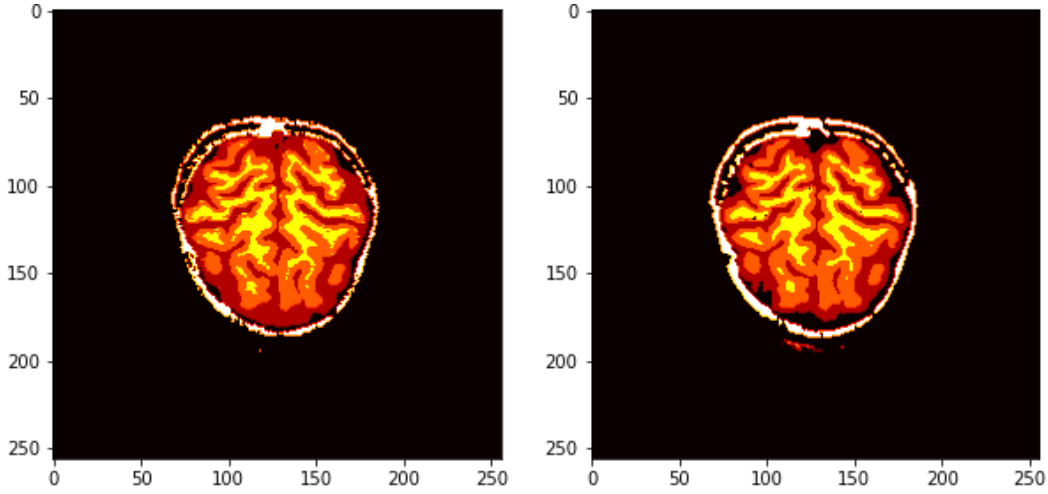
el etiquetado realizado en este trabajo y el etiquetado hecho por el conjunto de prueba de *BrainWeb*.

Ahora, se muestran en la Figura 5.6, distintos ejemplos de cortes de la aplicación del segmentador diseñado. Los mapas de probabilidad posterior por tejido estructuran la segmentación final entregada, se puede observar que a pesar de tener presencia de tejido encontrado (probabilidad posterior $p_p > 0$) en un mapa distinto al correcto, la probabilidad máxima encontrada con *arg max* toma la decisión de etiquetado con mayor probabilidad, según el conjunto de verdad fundamental. Se muestran nuevamente comparaciones entre el etiquetado propio y la verdad fundamental.

Para terminar esta sección, se muestran ejemplos de la renderización de mapas probabilísticos resultantes de la segmentación, es decir, la representación espacial de los tejidos segmentados en este proyecto. En las imágenes 5.7 y 5.8, se muestran los ejemplos específicos de materia blanca y materia gris como resultado de la salida de la red convolucional (con previo filtrado espacial).



(a) *Ejemplo 1.*



(b) *Ejemplo 2.*

Figura 5.5: *Ejemplos de comparación entre cortes del etiquetado proveniente de BrainWeb vs salida para un volumen de prueba. Izquierda: verdad fundamental, Derecha: salida de segmentación. Las relaciones de colores con las clases son: amarillo-materia blanca, rojo-LCR, naranja-materia gris, blanco-cráneo y negro-fondo.*

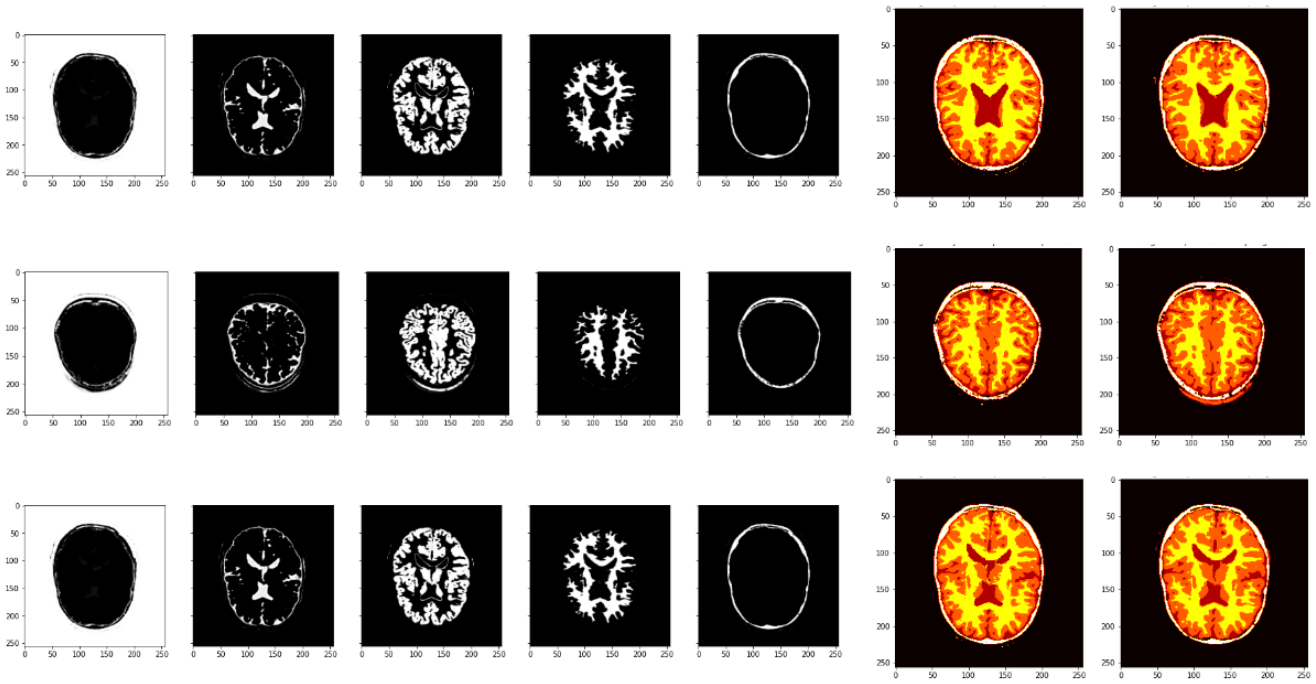


Figura 5.6: *Ejemplos de cortes de mapas de probabilidad posterior y su respectiva segmentación final. Por renglón; mapas probabilísticos, izquierda: fondo, centro izquierda: líquido cefalorraquídeo, centro: materia gris, centro derecha: materia blanca, derecha: cráneo. Figuras complementarias, izquierda: corte de volumen segmentado del conjunto de prueba (verdad fundamental), derecha: respectivo etiquetado por la red neuronal.*

5.2. Resultados cuantitativos

En esta sección se abordan resultados capaces de establecer un análisis formal y riguroso del desempeño de cada sección. Anteriormente se establece que la metodología de trabajo utilizada es filtrar los volúmenes con el corrimiento de media potenciado con el mapa de confianza de bordes, para después realizar la clasificación con la red UNET 3D. Para una evaluación completa y contrastante del segmentador, se implementa esta estructura de trabajo dos veces utilizando los mismos parámetros, a excepción de que uno de los dos segmentadores no hace uso del filtrado espacial, esto logra resaltar el efecto de este último.

5.2.1. Validación cruzada

El objetivo principal de la validación cruzada es recabar métricas que califiquen el rendimiento del clasificador con datos diversos. El primer resultado aborda el coeficien-

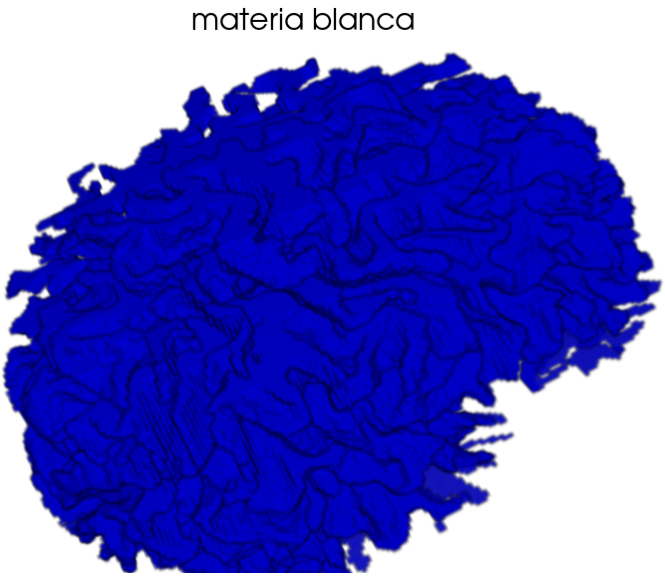


Figura 5.7: *Ejemplo de mapa probabilístico renderizado, perteneciente a materia blanca, resultante de la segmentación. Se aplicó un filtrado por tamaño minimo de componentes conectados.*

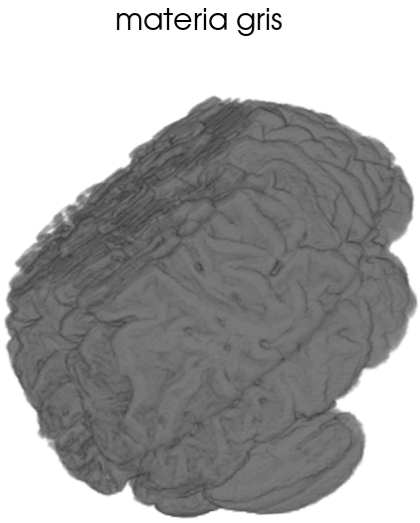


Figura 5.8: *Ejemplo de mapa probabilístico renderizado, perteneciente a materia gris, resultante de la segmentación. Se aplicó un filtrado por tamaño minimo de componentes conectados.*

Tabla 5.1: *Precisión de los modelos de segmentación con diferencia en el uso de filtrado espacial*

Filtrado espacial 4D	Precisión en Validación Cruzada						
	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Media (D. E.)
No usado	0.9408	0.9221	0.9549	0.9290	0.9384	0.9185	0.9339 (0.0123)
Usado	0.9590	0.9753	0.9503	0.9568	0.9692	0.9843	0.9657 (0.0117)

te de precisión, el cual es un indicador de la calidad de un modelo de IA y específicamente se refiere al número de predicciones correctas dividido por el número total de clasificaciones. La Tabla 5.1 muestra la precisión reportada en cada uno de los 6 ciclos de la validación cruzada de cada segmentador (sin y con filtrado espacial), además de las medias y desviaciones estándar para cada caso.

El siguiente resultado explora las curvas de entrenamiento, se tratan de gráficos que muestran cambios en el desempeño del aprendizaje a lo largo del tiempo en modelos de aprendizaje automático. Existen dos clases de curvas: de error, la cual monitorea la función de pérdida, para este caso la entropía cruzada categórica. Y de precisión, la cual da seguimiento durante el entrenamiento al indicador de precisión que compara la salida de la red con la verdad fundamental.
Para el caso de la validación cruzada, se tienen 6 pares de curvas de entrenamiento, dado que se utilizan 6 vías. Como una medida de cotejo más puntual, se calcula el promedio e intervalo de confianza de las 6 curvas. Este proceso se realiza para los dos segmentadores realizados. En la Figura 5.9 se exponen dichas curvas para cada implementación.

5.2.2. Rendimiento

Uno de los grandes retos de este trabajo fue aminorar los tiempos de procesamiento, desde el comienzo era sabido que los tiempos de ejecución serían grandes considerando el equipo de cómputo a usar. El algoritmo de corrimiento de media y la red neuronal exigen de sobremanera a la capacidad computacional debido a la búsqueda de modas de la distribución de puntos, y por otro lado, el entrenamiento y ejecución de la red convolucional U-NET 3D. Para estos dos procesos, se destina el recurso de aceleración por GPU, ambos con la intención de paralelizar procesos internos y reducir considerablemente el tiempo de término de cada tarea.
El uso de la tarjeta gráfica como acelerador de procesos establece el compromiso por en-

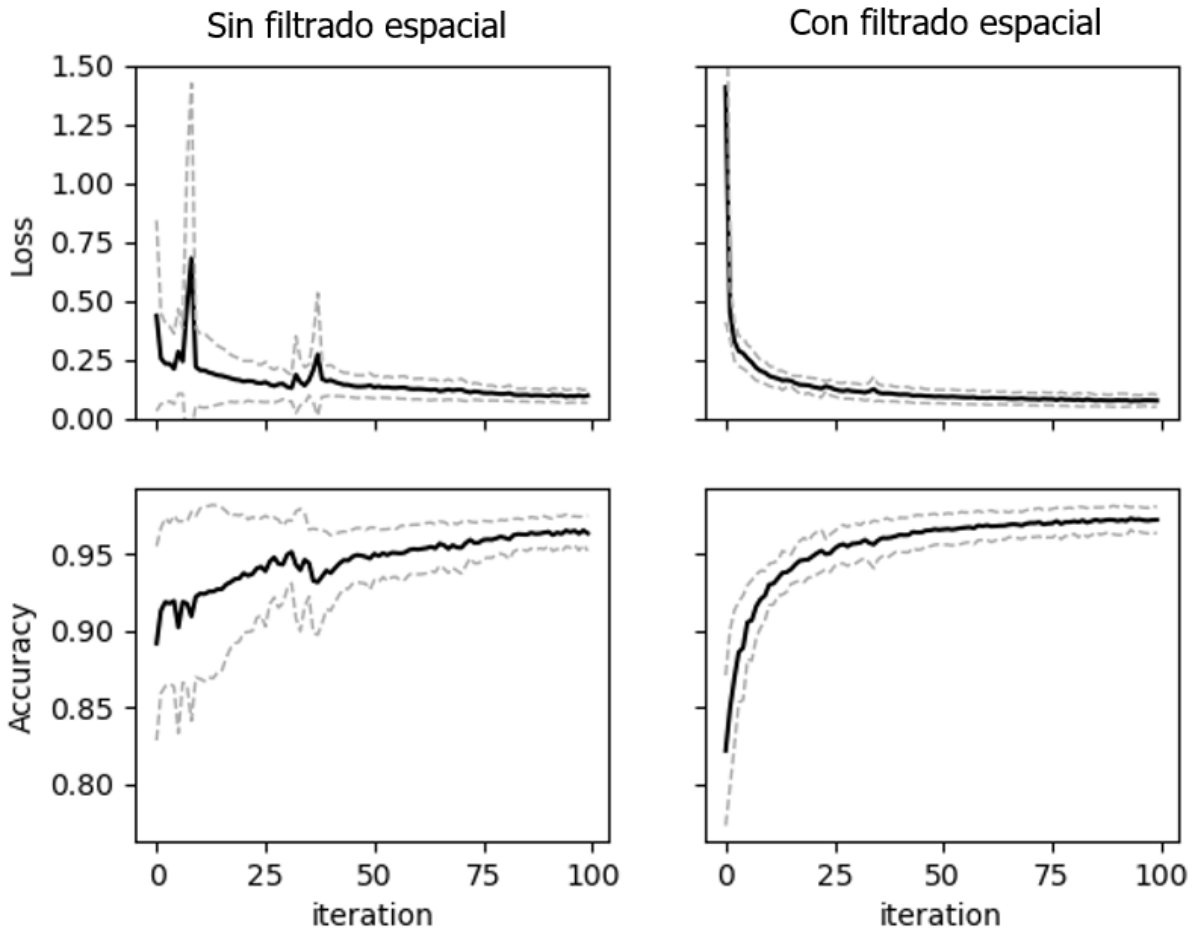


Figura 5.9: *Curvas de aprendizaje de las implementaciones del clasificador. Las líneas continuas son valores medios de validación cruzada y las líneas punteadas indican los intervalos de confianza correspondientes.*

tregar resultados de calidad con prioridad del tiempo de ejecución, en la Tabla 5.2, se presentan los tiempos de procesamiento de cada etapa de operación del segmentador diseñado con relación a los distintos entornos de ejecución disponibles para este trabajo. Se nota que el mapa de confianza de bordes fue utilizado únicamente con un entorno con soporte de CPU, debido al cuidado en la memoria VRAM de la GPU, tal como se menciona en secciones anteriores.

5.3. Modelo final

Dado que la validación cruzada que hace uso del filtrado espacial arroja métricas aceptables para este trabajo, se establece dicho modelo de arquitectura, función de pérdida,

Tabla 5.2: *Tiempos de ejecución de cada método al procesar un volumen dependiendo del entorno utilizado. n/a: no se realizó la implementación de dicho proceso en el entorno correspondiente.*

Tiempo de ejecución				
Proceso	CPU Ryzen 5	GTX1650	Tesla T4	A100SXM
Mapa de confianza de bordes	3 min	n/a	n/a	n/a
Corrimiento de media	23 hr	7 hr 28 min	3 hr 20 min	40 min
Predicción UNET 3D	6 min	4 min	20s	$<$ 1s

Tabla 5.3: *Indicadores de precisión para cada etapa de entrenamiento y media, del modelo final.*

	Precisión del modelo final					
	Submodelo 1	Submodelo 2	Submodelo 3	Submodelo 4	Submodelo 5	Media (D. E.)
Precisión	0.9673	0.9722	0.9730	0.9759	0.9896	0.9756 (0.0118)

optimizador y número de épocas de entrenamiento, como hiperparámetros oficiales para el modelo final de red.
La implementación del entrenamiento es el mismo, particionar los conjuntos de entrenamiento y pruebas en 5 partes. Para después entrenar el modelo con la primer parte del conjunto de entrenamiento, realizar las pruebas y evaluación con su conjunto análogo, guardar el modelo y métricas de evaluación y repetir el proceso hasta terminar las 5 partes.

La Tabla 5.3 expone la precisión en cada subproceso después de entrenar, probar y evaluar en cada uno de los 5 submodelos. El modelo final se presenta con la media y desviación estándar promedios de cada etapa. Se observa que en cada submodelo, la precisión arrojada en las pruebas supera el 0.96, además de reportar una desviación estándar del 1.18 %.

Capítulo 6

Discusión

6.1. Acerca del mapa de confianza de bordes

El mapa de confianza de bordes sólo puede analizarse desde el punto de vista cualitativo, ya que no hay alguna métrica para este trabajo que califique su rendimiento, pero si es posible ver el impacto sobre el filtrado espacial completo. En las Figuras 5.2 y 5.3, se observa en los cortes con mapa de color con pseudocolor aleatorio, que los tejidos que más adelante se clasifican con la red neuronal en el siguiente proceso, ya pueden diferenciarse con mayor facilidad, sobre todo resaltando las fronteras entre niveles de color.

Las Figuras 5.1a y 5.1b, contienen un efecto en la parte del mapa de bordes, en donde los extremos del corte pierden forma y continuidad. Este fenómeno se debe a que los volúmenes no son de dimensiones iguales en los ejes, es decir, su forma de 160x256x256 vóxeles, provoca que el cálculo del borde ideal en la ventana 3D para valores cerca a los extremos del volumen resulte con alteraciones por operar un kernel cúbico en un espacio de distinta forma.

Los volúmenes del mapa de borde ideal se normalizan a valores entre 0 y 1 debido al cálculo de la correlación entre ventanas, en los cortes mostrados en la Figura 5.1, se muestran datos de valores muy diferenciados entre ellos, con aproximaciones a valores

cercanos a 0 y 1, la consecuencia de esto es la distinción entre vóxeles definidos confiables como bordes y confiables como zonas homogéneas, situando así como extremos del rango dinámico los valores contenidos en el mapa de bordes. El cierre de esta idea es la nula aparición de vóxeles con valores cercanos a la mediana de los mismos, en consecuencia cada voxel del volumen es de únicamente dos naturalezas posibles: borde o no borde.

6.2. Acerca del filtro espacial

El primer resultado notorio en la implementación del filtro espacial es el impacto del mapa de bordes, los volúmenes provenientes destacan la frontera existente entre tejidos, aún cuando éstos no han sido formalmente etiquetados. Los bordes presentes en ejemplos como en las Figuras 5.2 y 5.3, tienen la facultad de diferenciar las estructuras visualmente, lo que hace pensar en el futuro impacto en etiquetado con la red convolucional, ya que desde ese punto logran facilitar la identificación de cada etiqueta relativamente.

En las mismas figuras, se puede percatar la existencia de más modas en volúmenes crudos en comparación a volúmenes filtrados. Los cortes en pseudocolor logran esclarecer el posible efecto visual en la continuación del rango dinámico, esto puede confundir visualmente el suavizado de zonas presuntamente homogéneas y el contraste entre distintos tejidos. La importancia de este resultado es observar la reducción en el número de modas encontradas, lo cual es una consecuencia del filtrado espacial propuesto. En los cortes en escala de grises, se aprecian los efectos visuales inmediatos de la aplicación de esta metodología, con la materia blanca mayormente diferenciada de la materia gris, incluso en zonas donde se traslapan de forma importante o bien, en zonas de poco espacio donde una clase se encuentra completamente rodeada por otra, como el caso del líquido cefalorraquídeo rodeado por la materia blanca.

En resumen, el filtro espacial contribuye a la segmentación de los volúmenes al realizar una diferenciación suave de los tejidos, esto le permitirá a la red neuronal terminar de identificar la presencia de 5 tejidos, además de etiquetar cada uno de ellos en los volúmenes.

6.3. Acerca de la red convolucional

La arquitectura de red U-NET 3D implementada, presenta como salida 5 mapas de probabilidad posterior en forma de volúmenes de 160x256x256 vóxeles cada uno, los cuales corresponden a los mapas que definen probabilísticamente a que tipo de tejido pertenece cada vóxel. Estos mapas tienen un rango dinámico entre 0 y 1, debido a la probabilidad calculada.
La imágenes presentadas en la Figura 5.4, tienen características importantes a analizar. La primera de ellas es que los valores de gris de cada mapa son muy cercanos a 0 y a 1, con nula presencia de valores situados entre este rango, que pudieran apreciarse en la imagen como vóxeles de tonos grises en la escala. El análisis de esta característica es una importante forma de evaluar objetivamente la clasificación, ya que para cualquier etiqueta, la probabilidad de que un elemento pertenezca a dicha clase es altamente probable o nula. En el caso que existieran problemas en la identificación de clases, existirían mapas con probabilidades cercanas a 0.5, es decir, con tonos grises, no negro y blanco.

Por otra parte, el caso de distinción más complejo dentro de las clases estudiadas, es el de materia gris y materia blanca, ya que la morfología de los cerebros en los estudios explorados, presentan una fuerte cercanía entre estos tejidos, además de una díficil distinción de ellos con enfásis en zonas como bordes, debido a su proximidad en el rango de niveles de gris. Esto aumenta la crucialidad de un método de resalte de los límites entre clases, tal como el mapa de confianza de bordes. En la ilustraciones de la figura 5.4, se nota que los mapas correspondientes a estos tejidos no presentan valores de probabilidad irelevantes en zonas de mapas fuera de su etiquetado, a pesar de los retos mencionados.

El cálculo de la segmentación total se lleva a cabo utilizando los mapas de probabilidad, se comparan al traslaparse y encontrar sus valores máximos respectivos por posición, para después asignar la clase de cada voxel según el mapa correspondiente al valor máximo detectado. Los ejemplos de cortes mostrados en la Figura 5.5 comparan el resultado del procedimiento resultado *vs* la verdad fundamental dictada por el conjunto de prueba de *BrainWeb*.
Aunque en este paso la comparación es visual, se aprecia el parecido entre el corte de salida del sistema y el corte de la verdad fundamental. La materia blanca marca una clara frontera con la materia gris que la contiene, siendo este el caso que mayor dificultad de contraste tiene. Esto aunado a que este parecido se mantiene en cortes con un alto valor (rebanadas cercanas a la mitad del plano en cuestión) donde se exponen todos los tejidos de interés. Para fortalecer esta idea, las imágenes 5.7 y 5.8, correspondientes a la

reconstrucción 3D, mostradas dan un panorama de información espacial, en el cual su composición espacial está totalmente ligada a la segmentación..

También puede apreciarse en la Figura 5.5b, que existe un error notable en la clasificación del fondo, ya una porción de éste fue clasificado como materia gris, posteriormente esto se contrasta con las métricas puntuales de precisión, ya que errores de esta índole afectan dicha métrica.

Por último, resalta el hecho de que en volúmenes segmentados por el sistema, la clasificacación de cráneo presenta una mayor continuidad en su morfología en comparación con la verdad fundamental, si bien el conjunto de prueba dicta la calidad del segmentador, es posible enunciar que el sistema logra suavizar cambios abruptos en el nivel de gris, además de permitir un análisis visual más útil para el diagnóstico o apoyo médico.

6.4. Validación cruzada

En esta sección se analizan los resultados cuantitativos a partir del método de validación cruzada de 6 vías, con enfásis en la comparación de dos sistemas, el primero que hace uso únicamente de la red convolucional y el segundo que realiza filtrado espacial al volumen previo a la operación de la red.
Los valores del indicador de precisión estimados a partir de la validación cruzada, tanto para las redes entrenadas con y sin volúmenes filtrados espacialmente, se resumen en la Tabla 5.1. Se puede ver en el valor de precisión de cada lote, así como en la media calculada, que la clasificación mejora cuando se utiliza el conjunto de datos filtrados espacialmente en el diseño del sistema, este resultado es contundente desde la perspectiva cruda de tener un segmentador que clasifica con mayor precisión cuando se usa previamente el filtrado espacial. Además, los valores de precisión reportados en cada lote del segmentador con filtrado espacial superan el 95 %, mientras que el segmentador que no hace uso de esta etapa tiene dificultades para clasificar distintos lotes de la validación, ya que reporta precisiones menores al 93 %, la consecuencia de este efecto es que ese sistema no demuestra adaptabilidad a datos con una mayor diversidad en sus morfologías.

El contraste de medias de los dos sistemas dicta la gran ventaja del uso del filtrado espacial, pero la desviación estándar también es un resultado favorecedor a este hecho, ya que en la metodología que hace uso de este recurso, la desviación estándar reduce su

valor, esto se traduce que la diversidad de volúmenes que procesa no implica una dificultad en la calidad de la clasificación ya que es demostrado en la validación cruzada que sin importar el lote de datos con el que se entrenó el sistema, la precisión se mantiene arriba del 95 %, esto empata con la reducción de la desviación estándar en el sentido de reducir la variabilidad de la calidad de su segmentación sin reestricciones en la morfología del volumen en cuestión.

Otros resultados de la validación cruzada son las curvas de aprendizaje, de las cuales es posible encontrar efectos en la morfología de las curvas que describan el proceso de entrenamiento.

En las curvas de aprendizaje en la figura 5.9 radica uno de los resultados más importantes del proyecto, éstas demuestran el efecto del filtrado espacial diseñado. Las curvas muestran los valores medios de error y precisión de la validación cruzada de 6 vías y los intervalos de confianza correspondientes.

Las curvas correspondientes al segmentador que no hace uso de filtrado espacial contienen sobretiros y transicisiones abruptas e inestables, su forma indica que el conjunto de datos de entrenamiento no fue representativo, esto significa que el conjunto de entrenamiento de cada set no proporcionó suficiente información para aprender el problema con el escenario operado, es decir, este sistema necesita una mayor cantidad de volúmenes para un entrenamiento con mejores resultados, esta cuestión dificulta una medida correctiva en este trabajo, ya que el conjunto de datos de *BrainWeb* se limita a 20 volúmenes de RM cerebral.

La diferencia entre los pares de gráficos (con filtrado y sin filtrado) se nota visualmente. Las métricas finales al concluir el entrenamiento favorecen mucho al segmentador que hace uso de datos filtrados espacialmente, debido a que el segmentador con filtrado tiene una mayor precisión y un menor índice de error, en la última época. Este resultado sería suficiente para establecer que la calidad del segmentador diseñado sustenta su estructura. Pero además, la morfología de las curvas de aprendizaje de este sistema muestra transiciones suaves y una continuidad estable, disminuyendo o aumentando gradualmente según el gráfico, con formas aproximadas a funciones logarítmicas. En contraste, el modelo construido sin filtrado presenta un cronograma de aprendizaje inestable y sin una tendencia clara, incluso con disminuciones intermitentes en la mejora del entrenamiento, presumiéndose de el sobreajuste de sus parámetros.

Los resultados de la validación cruzada en métricas númericas y gráficos de aprendi-

Tabla 6.1: *Matriz de confusión de la clasificación de materia gris, materia blanca y otro tejido, usando SPM en volúmenes del repositorio por Klauschen, Goldman, et al. BrainWeb [31].*

	Materia gris	Materia blanca	Cualquier otro tejido
Materia gris	0.891	0.068	0.041
Materia blanca	0.091	0.908	0.005
Cualquier otro tejido	0.008	0.000	0.991

zaje, cierran la idea de la ventaja del uso del filtrado espacial, en el entrenamiento del modelo así como en su calidad final.

La prueba final de evaluación de calidad del segmentador es cotejar si la métrica de precisión del modelo final se encuentra dentro del intervalo de confianza reportado por la validación cruzada, en el caso que no sea así, se entiende que no existe sustento de que la calidad del modelo final no haya sido producto de situaciones aleatorias que produjeron para ese caso resultados aceptables particulares. El rango de intervalo de confianza calculado durante la validación cruzada corresponde a:

$$[0.9540, 0.9775]$$

Con una precisión del modelo final de 0.9756.

Por último, se refuerza la calidad de la segmentación de este sistema al mostrar, en la Figura 6.1, la matriz de confusión correspondiente a la clasificación de materia gris, materia blanca y cualquier otro tejido, utilizando el software *Statistical Parametric Mapping*, en el trabajo de Klauschen, Goldman, et al. [31]). A su vez, el valor de precisión dada la matriz de confusión corresponde a 0.9445, lo cual anuncia un desempeño inferior en comparación al proyecto presentado. Aunado a esto, es importante resaltar la diferencia del número de etiquetas por clasificar en cada caso, ya que el ejercicio de segmentar 5 tejidos supone un esfuerzo mayor que segmentar solamente 3 de ellos, tomando en cuenta que se usan los mismos datos.

6.5. Rendimiento

Como ha sido mencionado anteriormente, este trabajo tiene un compromiso con la reducción del tiempo de ejecución de las distintas etapas, por lo cual el rendimiento de cada proceso fue puesto a prueba en distintos entornos, el análisis en esta sección se basa

en la comparación del rendimiento de cada método según el poder computacional con el que fue desarrollado.
La Tabla 5.2 muestra los tiempos de ejecución de los diferentes procesos dependiendo del entorno donde se ejecutaron. Estos tiempos de ejecución corresponden a los mejores alcanzables, dada la experimentación con diferentes entornos, GPUs, estrategias de asignación de tareas y complejidad, que impactan directamente en el tiempo de ejecución de la mayoría de los procesos.
Es notable la diferencia entre entornos, sobre todo en el corrimiento de media, el cual es el proceso que mayor esfuerzo demanda, dado que es un proceso iterativo que realiza una alta cantidad de operaciones con un gran número de operandos. El impacto del uso de una GPU *vs* una CPU es relevante, ya que según los resultados, la GPU de menor gama responde aproximadamente tres veces mejor que la CPU disponible, lo que ya implica una mejora considerable, cabe recalcar que la CPU utilizado se sitúa en el mercado actual como un procesador de mediana gama. Por otro lado, en la comparación entre GPUs destaca la razón actual de la búsqueda de mejores tarjetas gráficas para procesos relacionados con aprendizaje maquinal y aprendizaje profundo, ya que la GPU de alta gama rinde cerca de 11 veces más rápido que la GPU de menor gama disponible para este trabajo, un aprovechamiento muy importante del tiempo que hace diferencia en procesos largos. Comparando el rendimiento de la tarjeta GTX1650 *vs* la Tesla T4, se encuentra que la última actúa poco más del doble de rápido.

La ejecución de la clasificación por la red convolucional no conlleva un esfuerzo computacional relevante en contraste al entrenamiento realizado antes, aunque las métricas de tiempo reportadas enfatizan nuevamente el impacto de la aceleración por tarjeta gráfica. Si bien 6 minutos reportados como tiempo de procesamiento para la CPU en este proceso, no son relevantes en comparación al tiempo respectivo del corrimiento de media pero si difieren en sobremanera en confrontación con los entornos con GPU, el ejemplo más rotundo se marca en que la GPU de mayor gama disponible clasifica el volumen de entrada 360 veces más rápido de la CPU usada.

Capítulo 7

Conclusiones

Es sabido que las redes neuronales profundas tienen una alta capacidad de procesamiento de datos, aunque siempre es posible explorar variantes y alternativas que impactan en la calidad de las soluciones. Este es el caso del enfoque propuesto, lo cual tiene un impacto de diversas maneras en el diseño y rendimiento de la red 3D U-NET utilizada.
Hoy en día, el uso de aprendizaje profundo se hace de manera indiscriminada, en el mayor de los casos y sin importar el objetivo, se utiliza como herramienta de caja negra, sin consciencia si el modelo a usar y su operación es la mejor solución dado el problema en cuestión. Para este trabajo se abordó la estructura crítica de evaluar y analizar el impacto del filtrado espacial sobre una red neuronal convolucional U-NET 3D, con la hipótesis de que su efecto favorecería la calidad del entrenamiento y su posterior ejecución. Después de resultados y análisis, esta hipótesis se cumple con fundamento en que los resultados del filtrado espacial son volúmenes con zonas homogéneas más uniformes, limpias y suaves, paralelamente logran tener bordes mayormente demarcados y en conjunto alcanzan una mejor diferenciación entre tejidos dentro del volumen, lo que en resumen resulta en una aproximación a la clasificación de tejidos, esto permite que la red convolucional clasifique las etiquetas con mayor facilidad que si tomara los datos crudos con alto contenido en ruido.

Otro punto, es que realizar el cambio y optimización de una rutina por una implementación relativamente compleja, como la programación de bajo nivel de funciones

establecidas en bibliotecas no disponibles con CUDA, reduzca un 34,7 % de tiempo de ejecución en una tarea de gran impacto, sabiendo que el mismo hardware utilizado en un distinto paradigma pueda paralelizar tareas que involucran operaciones matriciales, recalcando que esta disminución se lleva a cabo con el uso de una GPU de gama baja. Esto representa una ventaja en términos económicos ya que los equipos computacionales de bajo costo relativo pueden ser aprovechados a niveles significantes sobre todo en tareas que implican muchos datos.
El uso de la GPU es notable, especialmente con los beneficios que brinda CUDA, aunque su implementación tiende a ser difícil ya que involucra operaciones primarias como multiplicaciones de matrices elemento por elemento, sincronización de hilos, cálculos de media con ciclos, entre otros. Su rendimiento superó ampliamente al de la CPU, sin embargo, cabe señalar que el valor económico de una GPU se refleja en su rendimiento, sobretodo en tareas tan extensas como aquellas que incluyen el algoritmo del corrimiento de media. La tarjeta gráfica A100SXM es 34 veces más rápida que el sistema con CPU probado y 11.25 veces más rápida que la GPU con los núcleos CUDA más bajos probados. Finalmente, la calidad del filtrado espacial que realizan los diferentes entornos es la misma, lo que hace que la comparación sea sólo en el tiempo.

Uno de los resultados más importantes encontrados son las curvas de aprendizaje, en el sistema de segmentación propuesto muestran que la aplicación previa del filtrado por corrimiento de media potenciado con el mapa de confianza de bordes, a la segmentación de la red, produce volúmenes que facilitan el curso de entrenamiento de ésta, con progresiones parsimoniosas del valor de las funciones de error y precisión de época a época. También, se observa que las curvas del sistema sin filtrado presentan sobretiros y transicisiones abruptas en su valor, se puede considerar que los datos de entrenamiento no fueron suficientes para la cantidad de épocas de entrenamiento, por lo que necesitarían una mayor cantidad de datos para un mejor rendimiento. Estas últimas ideas en conjunto hacen concluir que al utilizar esta estructura de segmentación se tendrán entrenamientos de mejor calidad con menos épocas y con mayor estabilidad, además de tener la certeza de evitar caídas en el rendimiento del entrenamiento. Asimismo, la estabilidad en la progresión de las curvas del modelo con filtrado, aún cuando se realizan entrenamientos a trozos para la generación del modelo final, hace crecer la idea de seguir alimentando esta red para futuras aplicaciones con la persuasión de que el entrenamiento seguirá presentando este rasgo crucial.

Bajo un panorama de reflexión completo, se realizó en análisis de las ventajas en la estructura propuesta para este segmentador, con alcance en la calidad del entrenamiento

de la red y la reducción en el tiempo de procesamiento, aunque el resultado final desde una perspectiva cruda, destaca la métrica de evaluación como la característica más importante de este sistema. Para reforzar la idea de calidad del segmentador de RM cerebral realizado, los valores de precisión de cada prueba establecidos en la validación cruzada superan el 95 % y el modelo final presentado tiene una precisión del 97.56 % (± 1.18 %).

A criterio propio, la cifra del modelo final supera las expectativas establecidas al inicio de este proyecto, culminando así la formación de un segmentador con una alta eficiencia de entrenamiento y operación, además de abonar al contexto del procesamiento de imágenes con aprendizaje profundo, al sentar la base de aplicar el filtrado espacial por el algoritmo del corrimiento de media potenciado por el mapa de bordes, a volúmenes de RM cerebral, previo a su ingreso a una red convolucional U-NET 3D.

Bibliografía

[1] J.O.S.H. Cleary y A.R. Guimarães. «Magnetic Resonance Imaging». En: *Pathobiology of Human Disease*. Ed. por Linda M. McManus y Richard N. Mitchell. San Diego: Academic Press, 2014, págs. 3987-4004. ISBN: 978-0-12-386457-4. DOI: https://doi.org/10.1016/B978-0-12-386456-7.07609-7. URL: https://www.sciencedirect.com/science/article/pii/B9780123864567076097.

[2] Ridgway John P. «Cardiovascular magnetic resonance physics for clinicians: part I». En: *Journal of Cardiovascular Magnetic Resonance* 12.1 (nov. de 2010), pág. 71. ISSN: 1532-429X. DOI: 10.1186/1532-429X-12-71. URL: http://www.jcmr-online.com/content/12/1/71.

[3] Rafael C. Gonzalez y Richard E. Woods. *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X. URL: http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X.

[4] Rashi Bist, Ritu Vijay y Shweta Singh. «Comparative Analysis of Fixed Valued Impulse Noise Removal Techniques for Image Enhancement». En: abr. de 2018, págs. 175-184. ISBN: 978-981-13-1809-2. DOI: 10.1007/978-981-13-1810-8_18.

[5] L. A. Rosas-Castillo, O. Yáñez-Suárez y V. Medina-Bañuelos. «Segmentación de imágenes de resonancia magnética utilizando filtrado espacial y aprendizaje profundo». En: *Memorias del Congreso Nacional de Ingeniería Biomédica* 8.1 (nov. de 2021), págs. 82-86. URL: https://memoriascnib.mx/index.php/memorias/article/view/897.

[6] Wei Chen, Xingbo Hu, Wen Chen et al. «Airborne LiDAR Remote Sensing for Individual Tree Forest Inventory Using Trunk Detection-Aided Mean Shift Clustering Techniques». En: *Remote Sensing* 10 (jul. de 2018), pág. 1078. DOI: 10.3390/rs10071078.

[7] Y. Lecun, L. Bottou, Y. Bengio et al. «Gradient-based learning applied to document recognition». En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324. DOI: 10.1109/5.726791.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky et al. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». En: *Journal of Machine Learning Research* 15.56 (2014), págs. 1929-1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren et al. «Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition». En: *Computer Vision – ECCV 2014*. Ed. por David Fleet, Tomas Pajdla, Bernt Schiele et al. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-10578-9.

[10] Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, págs. 1097-1105.

[11] François Chollet. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[12] Christian" "Szegedy, Wei" "Liu, Yangqing" "Jia et al. «Going deeper with convolutions». En: 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[14] Iqbal H. Sarker. «Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions». En: *SN Comput. Sci.* 2.6 (2021), pág. 420. DOI: 10.1007/s42979-021-00815-1.

[15] Alexander Selvikvåg Lundervold y Arvid Lundervold. «An overview of deep learning in medical imaging focusing on MRI». En: *Zeitschrift für Medizinische Physik, Volume 29, Issue 2, May 2019* (nov. de 2018). DOI: 10.1016/j.zemedi.2018.11.002. arXiv: 1811.10052 [cs.CV].

[16] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp et al. «3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation». En: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*. Springer International Publishing, 2016, págs. 424-432. DOI: 10.1007/978-3-319-46723-8_49. URL: https://doi.org/10.1007/978-3-319-46723-8_49.

[17] Sajedul Talukder. «GPU-based Medical Image Segmentation: Brain MRI Analysis Using 3DSlicer». En: *Artificial Intelligence Applications for Health Care*. CRC Press, feb. de 2022, págs. 109-121. DOI: 10.1201/9781003241409-6. URL: https://doi.org/10.1201/9781003241409-6.

[18] Martín Abadi, Ashish Agarwal, Paul Barham et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[19] NVIDIA, Péter Vingelmann y Frank H.P. Fitzek. *CUDA, release: 10.2.89*. 2020. URL: https://developer.nvidia.com/cuda-toolkit.

[20] John Nickolls, Ian Buck, Michael Garland et al. «Scalable Parallel Programming with CUDA». En: *Queue* 6.2 (2008), págs. 40-53. DOI: 10.1145/1365490.1365500.

[21] Zeynettin Akkus, Alfiia Galimzianova, Assaf Hoogi et al. «Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions». En: *J. Digit. Imaging* 30.4 (2017), págs. 449-459. DOI: 10.1007/s10278-017-9983-4.

[22] Aswathy A. L. y Vinod Chandra S. S. «Cascaded 3D U-Net architecture for segmenting the COVID-19 infection from lung CT volume». En: *Scientific Reports* 12.1 (feb. de 2022). DOI: 10.1038/s41598-022-06931-z. URL: https://doi.org/10.1038/s41598-022-06931-z.

[23] Hao Chen, Qi Dou, Lequan Yu et al. *VoxResNet: Deep Voxelwise Residual Networks for Volumetric Brain Segmentation*. 2016. arXiv: 1608.05895 [cs.CV].

[24] Shraddha Oza y Kalyani R. Joshi. «CUDA based Fast Bilateral Filter for Medical Imaging». En: Noida, India. Noida, India: IEEE, 2018, págs. 930-935. ISBN: 978-1-5386-3046-4. DOI: 10.1109/SPIN.2018.8474236.

[25] J. R. Jimenez-Alaniz, V. Medina-Banuelos y O. Yanez-Suarez. «Data-driven brain MRI segmentation supported on edge confidence and a priori tissue information». En: *IEEE Transactions on Medical Imaging* 25 (2006), págs. 74-83. ISSN: 0278-0062. DOI: 10.1109/tmi.2005.860999.

[26] Kh Tohidul Islam, Sudanthi Wijewickrema y Stephen O'Leary. «A Deep Learning Framework for Segmenting Brain Tumors Using MRI and Synthetically Generated CT Images.» En: *Sensors (Basel, Switzerland)* 22 (2 ene. de 2022). ISSN: 1424-8220. DOI: 10.3390/s22020523. epublish.

[27] Ekaba Bisong. «Google Colaboratory». En: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, 2019, págs. 59-64. ISBN: 978-1-4842-4470-8. DOI: 10.1007/978-1-4842-4470-8_7. URL: https://doi.org/10.1007/978-1-4842-4470-8_7.

[28] Berengere Aubert-Broche, M. Griffin, G. B. Pike et al. «Twenty New Digital Brain Phantoms for Creation of Validation Image Data Bases». En: *IEEE TMI* 25 (2006), págs. 1410-1416. ISSN: 0278-0062. DOI: 10.1109/tmi.2006.883453.

[29] Diederik P. Kingma y Jimmy Ba. «Adam: A Method for Stochastic Optimization». En: *arXiv e-prints*, arXiv:1412.6980 (dic. de 2014), arXiv:1412.6980. arXiv: 1412.6980 [cs.LG]. URL: https://ui.adsabs.harvard.edu/abs/2014arXiv1412.6980K.

[30] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[31] Frederick Klauschen, Aaron Goldman, Vincent Barra et al. «Evaluation of Automated Brain MR Image Segmentation and Volumetry Methods». En: *Human brain mapping* 30 (abr. de 2009), págs. 1310-27. DOI: 10.1002/hbm.20599.

[32] Mohammad Hesam Hesamian, Wenjing Jia, Xiangjian He et al. «Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges». En: *J Digit Imaging* 32 (2019), págs. 582-596. ISSN: 0897-1889. DOI: 10.1007/s10278-019-00227-x.

[33] Ali Hatamizadeh, Dong Yang, Holger Roth et al. «UNETR: Transformers for 3D Medical Image Segmentation». En: *CoRR* abs/2103.10504 (2021). arXiv: 2103.10504. URL: https://arxiv.org/abs/2103.10504.

[34] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

Anexo

Recursos de cómputo

El equipo de cómputo del entorno local utilizado fue una laptop Lenovo Ideapad Gaming 3 con procesador AMD Ryzen 5 5600H con Radeon Graphics, 8GB de RAM y tarjeta gráfica NVIDIA GeForce GTX 1650, sobre el cual se elaborará el sistema de segmentación en Python 3.9.12 con la distribución Anaconda. El aceleramiento por GPU utilizando Python y una tarjeta gráfica NVIDIA tiene ciertas reestricciones acerca de las versiones de CUDA y Tensorflow, entre otros, por lo que a continuación se especifícan las bibliotecas y versiones utilizadas:

- conda 4.12.0
- cudatoolkit 11.3.1
- cudnn 8.2.1.32
- keras 2.13.0
- matplotlib 3.5.1
- numpy 1.22.3
- opencv 4.5.1
- scikit-learn 1.0.2
- scipy 1.8.0

- tensorflow 2.6.0
- tensorflow-base 2.6.0
- tensorflow-estimator 2.6.0
- tensorflow-gpu 2.6.0

Análogamente, las versiones de bibliotecas utilizadas en Colab y ColabPro son las siguientes:

- keras 2.13.1
- matplotlib 3.7.1
- numba 0.56.4
- numpy 1.22.5
- opencvcontrib-python 4.8.0.76
- scikit-learn 1.2.2
- scipy 1.11.3
- tensorflow 2.13.0
- tensorflow-base 2.13.0
- tensorflow-estimator 2.13.0
- tensorflow-gpu 2.13.0

Código

Código del cálculo del mapa de confianza de bordes

```
@njit
def confianza(im,sobelx,sobely,sobelz):
  #Tamaño de ventana
  sizeven=5
  #Recipiente del mapa de bordes
  mapbor=np.zeros_like(im)
  medioven=int(sizeven/2)
```

```
  for fil in range(0,im.shape[2]-sizeven):
    for col in range(0,im.shape[1]-sizeven):
        for sli in range(0,im.shape[0]-sizeven):
            roiimg=im[sli:sli+sizeven,col:col+sizeven,fil:fil+sizeven]
            roidx=sobelx[sli:sli+sizeven,col:col+sizeven,fil:fil+sizeven]
            roidy=sobely[sli:sli+sizeven,col:col+sizeven,fil:fil+sizeven]
            roidz=sobelz[sli:sli+sizeven,col:col+sizeven,fil:fil+sizeven]
            # Arreglo de gradientes Sobel
            h=np.array([ roidz[medioven,medioven,medioven],roidx[medioven,medioven,medioven]
            , roidy[medioven,medioven,medioven] ])
            h = h/np.sqrt(h@h)
            roiborde=np.zeros_like(roiimg) # Máscara de borde
            for x in range(0,roiborde.shape[1]):
                for y in range(0,roiborde.shape[2]):
                    for z in range(0,roiborde.shape[0]):
                        guarda=np.array([medioven-z,medioven-x,medioven-
                          y],dtype='float64')@h.astype('float64') # Generación de kernel de borde ideal
                        # Criterio para delimitar relación con el vector del centro de la ventana
                        if guarda<0:
                          roiborde[z,x,y]=1
                        else:
                          roiborde[z,x,y]=0
            if np.sum(roiborde)==0:
                mapbor[sli+medioven,fil+medioven,col+medioven]=0
            else:
                mapbor[sli+medioven,fil+medioven,col+medioven]=np.sum(roiborde*roiimg)
                    /np.sum(roiborde*roiborde)
    ### Magnitud de gradientes
  g_amp=np.sqrt((sobelz**2) + (sobelx**2) + (sobely**2))
  g_amp=g_amp/np.max(g_amp) # Normalización
  # El resultado final es el producto de mapa de bordes y magnitud de gradiente
  pro=g_amp*mapbor
  pro=pro/np.max(pro)

  return pro
```

Código del algoritmo de corrimiento de media acelerado por CUDA

```
@cuda.jit
def ms(bor,td,salida,tdpow): #borravelgpu,tdgpu,salidagpu,tdpowgpu
  ### Variables :
  # Distancia euclideana por |td-xi|**2= (td*td).sum(axis=1) + xi@xi.T - 2*td@xi.T
  # td : matriz de coordenadas de todos los puntos
  # a : receptor del nuevo valor del voxel
  # xi2 : vector de coordenadas actuales del punto a medir distancias
  # en producto vectorial por si mismo xi@xi.T
  # tdxi : vector de multiplicación matricial de td@xi.T
  # tdpow : vector (calculado de manera externa de cuda ya que sólo se
  # necesita una vez) que contiene (td*td).sum(axis=1)
```

```
  #  UNA ITERACIÓN DE MS CUBRIENDO UNA HIPERESFERA GRANDE


  #Paralelización dada por calcular el nuevo valor de un voxel a la vez, n hilos = voxeles
  n=cuda.grid(1)
  w,z=td.shape
  if n<w:

    a = cuda.shared.array(shape=1, dtype=nb.float64)
    xi2 = cuda.shared.array(shape=1, dtype=nb.float64)
    con = cuda.shared.array(shape=1, dtype=nb.float64)
    tdxi1d=cuda.shared.array(shape=1, dtype=nb.float64)


    #Sincronización inicial del vector de coordenadas actual
    cuda.syncthreads()

    #Cálculo de xi@xi.T
    xi2 = 0   #quiza xi2[i,j]
    for k in range(4):#xi.T.shape[0]
          xi2 += td[n,k] * td[n,k]
    cuda.syncthreads()
    #Cálculo de td@xi.T

    a=0
    #Contador de cuántos puntos hay en la circunferencia marcada
    con=0
    cuda.syncthreads()

    #Cálculo de la expresión completa:  (td*td).sum(axis=1) + xi@xi.T  2*td@xi.T
    # En el caso que se detecte una distancia mayor al ancho de banda,
    #se suman las coordenadas correspondientes
    # ponderadas por el mapa de confianza y por último se dividen entre
    #el número de puntos sumados (promedio ponderado)
    for i in range(w):#td.shape[0]
          tdxi1d = 0
          for k in range(4):#xi.T.shape[0]
              tdxi1d += td[i, k] * td[n,k]
          if (tdpow[i] + xi2 +  2*tdxi1d) < 70:
            con+=1
            a= a + td[i,3]*(1bor[i]) #Se acumula la suma de los
            #niveles de grises de los puntos dentro de la esfera

    cuda.syncthreads()


    # El resultado de cada hilo es el cálculo de un nuevo valor de gris para cada vóxel (n)
    salida[n]=a / con

a=pro

## AJUSTES EN CPU
```

CÓDIGO

```
#Carga de volumen
borde=a/np.max(a) #valores de 0 a 1
borderecorte=borde
suave=255*volsuave/np.max(volsuave) #valores de 0 a 255
suaverecorte=suave

#Elaboración de matriz de espacio 3D (filas,columnas,intensidad) *td*
slic=np.arange(0,suaverecorte.shape[0],1)
fil=np.arange(0,suaverecorte.shape[1],1)
col=np.arange(0,suaverecorte.shape[2],1)
columnas,filas,slices=np.meshgrid(col,fil,slic) #Cruzado, columnas vertical y filas horizontal
td = np.vstack((slices.ravel(order='C'),filas.ravel(order='C'),columnas.ravel(order='C'),
#suaverecorte.ravel(order='C'))).T
print(suaverecorte.shape,borderecorte.shape,td.shape)

#Mapa de bordes enviado en 1D
borravel=borderecorte.ravel(order='C')

#Receptor de salida
salida=np.zeros_like(borravel)

############ USO DE HILOS Y BLOQUES ######################################
threadsperblock=128 #Normalmente de 128 a 512
blockspergrid_x=int(np.ceil(td.shape[0] /threadsperblock))
#blockspergrid_y=int(np.ceil(C.shape[1] /threadsperblock[1]))
blockspergrid=blockspergrid_x
print(blockspergrid,threadsperblock)
##########################################

##### ARRAYS A GPU #########
tdgpu=cuda.to_device(td)
#Se eleva al cuadrado td y se manda, para facilitar el proceso. Sólo se realiza una vez por volumen
tdpowgpu=cuda.to_device(np.sum(td[:,0:4]*td[:,0:4],axis=1))
print(tdpowgpu.shape)
borravelgpu=cuda.to_device(borravel)
salidagpu=cuda.to_device(salida)


### PUESTA EN MARCHA ######
start = time.perf_counter()
ms[blockspergrid,threadsperblock](borravelgpu,tdgpu,salidagpu,tdpowgpu)
cuda.synchronize()
end = time.perf_counter()
print("Elapsed (with compilation) = {}s".format((end  start)))

start = time.perf_counter()
resul=salidagpu.copy_to_host()
end = time.perf_counter()
print("Elapsed (with compilation) = {}s".format((end  start)))
```

Código de arquitectura, compilación y entrenamiento de red UNET 3D

```
vol_size=(32,256,256,1)
#Arquitectura de red UNET 3D
#4 clases sin craneo, 5 con craneo
num_clases=5
def u_net(num_clases):
  inputs=Input(vol_size) #entrada de red
  #Ruta de contracción
  con1=Conv3D(32,kernel_size=(3,3,3),activation='relu',padding='same')(inputs)
  con1=Conv3D(32,kernel_size=(3,3,3),activation='relu',padding='same')(con1)
  pool1=MaxPooling3D(pool_size=(2,2,2))(con1)

  con2=Conv3D(64,kernel_size=(3,3,3),activation='relu',padding='same')(pool1)
  con2=Conv3D(64,kernel_size=(3,3,3),activation='relu',padding='same')(con2)
  pool2=MaxPooling3D(pool_size=(2,2,2))(con2)

  con3=Conv3D(128,kernel_size=(3,3,3),activation='relu',padding='same')(pool2)
  con3=Conv3D(128,kernel_size=(3,3,3),activation='relu',padding='same')(con3)
  pool3=MaxPooling3D(pool_size=(2,2,2))(con3)

  con4=Conv3D(256,kernel_size=(3,3,3),activation='relu',padding='same')(pool3)
  con4=Conv3D(256,kernel_size=(3,3,3),activation='relu',padding='same')(con4)
  pool4=MaxPooling3D(pool_size=(2,2,2))(con4)

  con5=Conv3D(512,kernel_size=(3,3,3),activation='relu',padding='same')(pool4)
  con5=Conv3D(512,kernel_size=(3,3,3),activation='relu',padding='same')(con5)

  #Ruta de expansión
  up6 = concatenate([Conv3DTranspose(256, kernel_size=3, strides=(2,2,2),padding='same')(con5), con4],axis=4)
  con_up6=Conv3D(256,kernel_size=(3,3,3),activation='relu',padding='same')(up6)
  con_up6=Conv3D(256,kernel_size=(3,3,3),activation='relu',padding='same')(con_up6)

  up7 = concatenate([Conv3DTranspose(128, kernel_size=3, strides=(2,2,2),padding='same')(con_up6), con3],axis=4)
  con_up7=Conv3D(128,kernel_size=(3,3,3),activation='relu',padding='same')(up7)
  con_up7=Conv3D(128,kernel_size=(3,3,3),activation='relu',padding='same')(con_up7)

  up8 = concatenate([Conv3DTranspose(64, kernel_size=3, strides=(2,2,2),padding='same')(con_up7), con2],axis=4)
  con_up8=Conv3D(64,kernel_size=(3,3,3),activation='relu',padding='same')(up8)
  con_up8=Conv3D(64,kernel_size=(3,3,3),activation='relu',padding='same')(con_up8)

  #Capa final

  up9 = concatenate([Conv3DTranspose(32, kernel_size=3, strides=(2,2,2),padding='same')(con_up8), con1],axis=4)
  con_up9=Conv3D(32,kernel_size=(3,3,3),activation='relu',padding='same')(up9)
  con_up9=Conv3D(32,kernel_size=(3,3,3),activation='relu',padding='same')(con_up9)

  fm_out=Conv3D(num_clases,(1,1,1),activation='softmax',padding='same')(con_up9)#Mapas de salida

  #Modelo de red
  model = Model(inputs=[inputs], outputs=[fm_out])
```

```
  model.compile(optimizer="adam",metrics=['accuracy'],loss='sparse_categorical_crossentropy')
  return model

#Puesta en marcha
monitor_="val_accuracy"

model = u_net(num_clases)

#model_checkpoint = ModelCheckpoint('w.h5',monitor='val_accuracy',save_best_only=True)
#model_checkpoint = ModelCheckpoint(filepath='w.h5',monitor='val_accuracy',mode='max',save_best_only=True)
history=model.fit(X_train, y_train, batch_size=1,epochs=epocas,verbose=1)
```

Código de implementación de validación cruzada de 6 vías.

```
skf= KFold(n_splits = 6)
accs=[]
cms=[]
skf.split(X_train,y_train)
for trninx,tstinx in skf.split(X_train,y_train):
  print(trninx,tstinx.shape[0])
#OPCION POSIBLE HACER CADA FOLD Y TIRAR KERNEL GUARDANDO ACC
epocas=100
cs=0
for trninx,tstinx in skf.split(X_train,y_train): #le cuesta volver a comenezar el ciclo
  cs+=1
  print('\n KFOLD # : ',cs)
  print(X_train[trninx].shape,y_train[trninx].shape,y_train[tstinx].shape)
  model = u_net(num_clases)

  for parte in range(5):
    #X[:,parte*32:(parte*32)+32,:,:],t1[:,parte*32:(parte*32)+32,:,:]
    model.fit(X_train[trninx][:,parte*32:(parte*32)+32,:,:], y_train[trninx]
        [:,parte*32:(parte*32)+32,:,:], batch_size=1,epochs=epocas)

  volcompleto=np.empty((tstinx.shape[0],160,256,256,5))

  for i in range(5):
    X_test_=np.expand_dims(X_train[tstinx][:,i*32:(i*32)+32,:,:],4)
    print(X_test_.shape)
    #Prediccion
    #volpred = model.predict(X_test_, verbose=2) # Tamaño de volpred (4,160,256,256,5)
    #print(volpred.shape)
    volcompleto[:,i*32:(i*32)+32,:,:,:]=model.predict(X_test_, verbose=2)
    #Reformar el volumen de 160,256,256
  print('Volumen completo: ',volcompleto.shape)

  #mask = np.argmax(volcompleto, axis=1)
  #print(mask.shape)

  #print('\nMatriz de confusión\n',confusion_matrix(y_train[tstinx].ravel(), mask.ravel()))
  #print('Accurancy: ',accuracy_score(y_train[tstinx].ravel(), mask.ravel()))
```

```
    accs.append( accuracy_score(y_train[tstinx].ravel(),
      np.argmax(volcompleto, axis=1).ravel()) )
    #cms.append( confusion_matrix(y_train[tstinx].ravel(),
      np.argmax(volcompleto, axis=1).ravel()) )

    del volcompleto
    del model
    del X_test_

#Promedio y desviación estandar de precisiones
macc=np.mean(np.array(accs))
stdacc=np.std(np.array(accs))
```

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00155
Matrícula: 2213800903

Segmentación 3D acelerada por GPU de IRM cerebral utilizando filtrado espacial y aprendizaje profundo.

En la Ciudad de México, se presentaron a las 11:00 horas del día 16 del mes de diciembre del año 2024 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DR. JORGE LUIS PEREZ GONZALEZ
DR. EDUARDO BARBARA MORALES
DR. JUAN RAMON JIMENEZ ALANIZ

Bajo la Presidencia del primero y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS (INGENIERÍA BIOMÉDICA)

DE: LEONEL ADAN ROSAS CASTILLO

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

APROBAR

Acto continuo, el presidente del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

LEONEL ADAN ROSAS CASTILLO
ALUMNO

REVISÓ

MTRA. ROSALIA SERRANO DE LA PAZ
DIRECTORA DE SISTEMAS ESCOLARES

DIRECTOR DE LA DIVISION DE CBI

Román Linares Romero
DR. ROMAN LINARES ROMERO

PRESIDENTE

DR. JORGE LUIS PEREZ GONZALEZ

VOCAL

DR. EDUARDO BARBARA MORALES

SECRETARIO

Juan R. Jiménez A.
DR. JUAN RAMON JIMENEZ ALANIZ