ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Desarrollo de algoritmos de inteligencia artificial para la segmentación y clasificación de hiperintensidades en sustancia blanca cerebral obtenidas de imágenes de Resonancia Magnética.

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Ciencias Computacionales

Presentado por:
Sucre Kevin Cando Garces

GUAYAQUIL - ECUADOR
Año: 2018

DEDICATORIA

El presente proyecto lo quiero dedicar a mis padres, a mi hermana y mis familiares que han hecho que este trabajo sea posible.

AGRADECIMIENTOS

Mi más sincero agradecimiento a mis padres, mi tutor, mis maestros por ayudarme durante el proceso de elaboración y guiarme en el desarrollo investigativo

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; Sucre Kevin Cando Garces y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Kevin Cando

RESUMEN

Las hiperintensidades de sustancia blanca cerebral suelen estar asociadas a enfermedades cerebro vasculares debido al rol ocupado en el decaimiento cognitivo y perdida de funciones motrices en la vejez. Estas lesiones son visibles en imágenes obtenidas de secuencias FLAIR por Resonancia Magnética. El proceso de segmentación automática de estas lesiones es dependiente de las características de la máquina lo que se traduce en una falta de robustez en los algoritmos de segmentación tradicionales.

El método desarrollado en este trabajo parte de un conjunto de datos de imágenes FLAIR y T1 proporcionados por el concurso MICAI-2017 obtenidos por máquinas de diferentes características. También se proporcionó el "ground truth" consistente en la segmentación manual de esas lesiones. El método consistió en entrenar un modelo de aprendizaje profundo usando la red U-Net 3D. La red neuronal convolucional aquí propuesta consistió en 28 capas y 22 millones de parámetros entrenables.

Finalmente, el resultado de este proceso de segmentación se comprobó con el "ground truth" mediante cinco coeficientes de comparación. Los resultados mostraron que se logró una efectividad del 81% en el aprendizaje usando el coeficiente de DICE para hacer la comparación, sin embargo, también se pudo observar que el modelo identifica con gran grado de precisión las lesiones con información en el eje z. El método propuesto resultó robusto y simple al momento de segmentar imágenes tridimensionales, sin necesidad de tener que segmentar las imágenes tridimensionales a un modelo bidimensional.

Palabras Clave: WMH, CNN, U-Net3D, Modelos de aprendizaje

ABSTRACT

White matter hyperintensities are usually associated with cerebrovascular diseases due to the role they occupied in cognitive decay and loss of motor functions in old ages. These lesions are visible in images obtained from FLAIR sequences by Magnetic Resonance. The process of automatic segmentation of these injuries is dependent on the characteristics of the machine, which translates into a lack of robustness in the traditional segmentation algorithms. The method developed in this work is based on a data set of FLAIR and T1 images provided by the MICAI-2017 contest obtained by machines with different characteristics. The "ground truth" consists in the manual segmentation of these injuries which was also provided by the dataset. The method consisted in training a deep learning model using the U-Net 3D network. The convolutional neuronal network proposed here had 28 layers and 22 million trainable parameters. Finally, the result of this segmentation process was verified with the ground truth by comparison of five coefficients. The results showed an effectiveness of 81% which was achieved using the DICE coefficient to make the comparison, however, it could also be observed that the model identifies injuries with information in the z-axis with great precision. The proposed method was robust and simple when segmenting three-dimensional images, without having to segment the three-dimensional images to a two-dimensional model.

Keywords: Learning models, CNNs, U-Net3D, WMH, automatic segmentation.

ÍNDICE GENERAL

ABREVIATURAS

AVD Average Volume Difference
AWS Amazon Web Services
CNN …Convolutional Neural Network
CNTK Cognitive Toolkit
CPU Central Processing Unit
DSC DICE Similarity Coefficient
ESPOL Escuela Superior Politécnica del Litoral
FLAIR Fluid attenuated inversion recovery
FSL FMRIB Software Library
GPU Graphical Processing Unit
H95 Hausdoff Distance
ITK Insight Segmentation and Registration Toolkit
MLP ...Multi Layer Perceptron
Nifty Neuroimaging Informatics Technology Initiative
ReLU …Rectified Linear Unit
RM Resonancia Magnética
USD United State Dollars
WMH Hiperintensidades en sustancia blanca cerebral

Commented [FP1]: Favor, revisar, quedan algunas posteriormente ordenar alfabeticamente.

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

CAPÍTULO 1

1. INTRODUCCIÓN

Las hipertensidades de sustancia blanca cerebral (WMH por sus siglas en ingles), están asociadas al deterioro cognitivo (Lampe et al., 2017). Estas son relevantes debido al volumen que pueden presentar en adultos sanos sin demencia, lo cual puede tener efecto sobre las funciones motoras, de memoria o ejecución de funciones. Las enfermedades de vaso sanguíneo pequeño se refieren a un grupo de procesos patológicos los cuales afectan pequeñas arterias, arteriolas, vénulas y capilares del cerebro; las consecuencias de estas enfermedades son lesiones ubicadas en estructuras subcorticales tales como: infartos lacunares, lesiones por WMH, microsangrados y hemorragias grandes. Esto tiene una gran importancia en enfermedades cerebro vasculares debido al rol ocupado en el decaimiento cognitivo y perdida de funciones motrices en la vejez (Pantoni, 2010).

La cuantificación del volumen, ubicación y forma de la WMH es de importancia clave en los estudios de investigación clínica y es probable que encuentre su camino en la práctica clínica; para apoyar el diagnóstico, el pronóstico y la monitorización del tratamiento de la demencia y otras enfermedades neurodegenerativas. Se ha observado que la calificación visual de WMH tiene limitaciones importantes (Caligiuri et al., 2015) y, por lo tanto, se prefiere una segmentación más detallada de WMH. Se han desarrollado diversas técnicas automatizadas de segmentación de WMH, para proporcionar mediciones cuantitativas y reemplazar procedimientos de delineación dependientes del observador que consumen mucho tiempo.

Una revisión sobre técnicas automatizadas de segmentación de WMH reveló una cuestión clave: es difícil comparar varias técnicas (Caligiuri et al., 2015). Cada técnica de segmentación se evalúa en una verdad de terreno diferente (diferente número de sujetos, diferentes expertos, diferentes protocolos) y utilizando diferentes criterios de evaluación.

Es así como un equipo liderado por UMC Utrecht, VU Amsterdam, and NUHS Singapore, (http://wmh.isi.uu.nl) plantearon un reto en el 2017, en el cual buscan encontrar diferentes técnicas de segmentación que sean más eficientes. El trabajo actual

se basa en esos resultados mostrados por (Li et al., 2018) para poder establecer un antecedente sobre el cual se decidió actuar.

Los resultados de este trabajo se utilizarán en el laboratorio de Bioingeniera de ESPOL con el objetivo de presentar un punto de partida para el desarrollo de un software que sea capaz de generar la segmentación automática de lesiones en sustancia blanca (WMH), robusto e independiente de las condiciones particulares de cada experimento, capaz de alcanzar la más alta eficiencia.

1.1 Descripción del problema

En el 2015, en el Laboratorio de Bioingeniería de la ESPOL se desarrolló un programa que permitía segmentar de manera automática las hiperintensidades de sustancia blanca cerebral (Chancay et al., 2015); el mismo que resolvía el desafío propuesto UMC Utrecht, VU Amsterdam, and NUHS Singapore en 2017. Sin embargo, a la hora de probar con imágenes FLAIR obtenidas por otros centros (como los proporcionados por el concurso), los algoritmos carecen de robustez para obtener un buen resultado. La naturaleza de los datos, basados en otros protocolos de adquisición de imágenes y otras marcas de equipos, hizo que el algoritmo redujera significativamente el rendimiento al momento de hacer la segmentación automática.

El concurso realizado en 2017 tenía un dataset robusto constituido por imágenes de diferentes equipos conocidos, y al momento de probar la robustez del algoritmo también incluía equipos de otras marcas existentes en el mercado. El principal problema del algoritmo realizado en el laboratorio de bioingeniería de ESPOL fue la naturaleza de los datos. En el trabajo publicado por (Chancay et al., 2015) se dice que se trabajó con un dataset de una sola máquina de resonancia magnética (Siemens, TrioTim, 3 Tesla, con bobina de 12 canales), en el cual reclutaron 18 pacientes de Parkinson con media de edad = 57.44 años y 18 sujetos sanos pareados por edad. Los criterios de inclusión de dicho grupo de sujetos fueron: no padecer ninguna enfermedad y no estar tomando medicación alguna.

Este problema de segmentación se debía a que el algoritmo desarrollado tomaba como punto de segmentación los valores observados por un umbral (threshold) de las

frecuencias de la imagen, el cual depende mucho del conjunto de prueba a analizar, del protocolo de adquisición de la imagen y de las características del equipo como son: marca, modelo, potencia del campo magnético, tipo de bobina, etc. Esto sugiere un sesgo del algoritmo, lo cual evita que permita segmentar de manera correcta las lesiones de imágenes FLAIR y T1 realizadas en otras máquinas dónde cambian las variables descritas anteriormente.

1.2 Justificación del problema

Un hallazgo común asociado a personas de edad avanzada, demencia, esclerosis múltiple, accidentes cerebro vasculares y los trastornos motores son las lesiones en Sustancia Blanca (WMH por sus siglas en inglés) (Maniega et al., 2015). Además, las WMH han sido sistemáticamente relacionados con factores de riesgo vascular y deterioro funcional en las actividades cotidianas razones que en realidad su significación clínica es cada vez más importante (Wardlaw et al., 2013). Las lesiones en sustancia blanca se visualizan mejor con las secuencias FLAIR (Fluid Attenuated Inversion Recovery) en Resonancia Magnética (MR) que tienen la ventaja de la supresión de líquido cefalorraquídeo, permitiendo que las lesiones presenten un alto contraste (Caligiuri et al., 2015).

Con fines clínicos, las imágenes FLAIR se evalúan principalmente mediante inspección visual, sin embargo, esto no es suficiente para cuantificar y caracterizar el comportamiento fisiológico de diferentes enfermedades (Maniega et al., 2015). Se conoce que este tipo de lesiones en sustancia blanca se presentan tanto en sujetos sanos a partir de los 60 años y sin ninguna manifestación clínica, o en pacientes con diversos tipos de patologías (Lampe et al., 2017). En este sentido, se han desarrollado varios métodos automatizados y semiautomatizados para segmentar y cuantificar las WMHs usando diferentes técnicas, proporcionando información sobre el tamaño, forma y la ubicación principalmente. Estos trabajos están más enfocados en los procedimientos matemáticos para la segmentación ya sea automática o semi-automática de las lesiones.(Dalca, Balakrishnan, Guttag, & Sabuncu, 2018; Orbes-Arteaga et al., 2018; Pandey, Vasan, & Ramakrishnan, 2018; Pantoni, 2010; Respino et al., 2018; Wardlaw et al., 2013)

1.3 Objetivos

1.3.1 Objetivo General

Determinar si un método de aprendizaje profundo es capaz de realizar el proceso de segmentación automática de lesiones en sustancia blanca cerebral en imágenes de resonancia magnética.

1.3.2 Objetivos Específicos

- Realizar un método de preprocesado de las imágenes para aumentar el conjunto de datos de entrenamiento.
- Elaborar un modelo de aprendizaje profundo basado en la red U-Net 3D
- Validar la efectividad del método por medio de diferentes coeficientes de comparación.

1.4 Marco teórico

1.4.1 Redes Neuronales

Una red neuronal está compuesta de muchas neuronas artificiales, en el campo de las redes neuronales se inspiró originalmente en el objetivo de modelar los sistemas neuronales biológicos, pero desde entonces se ha ramificado en diferentes direcciones y se ha convertido en una cuestión de ingeniería con el fin de lograr buenos resultados en las tareas de aprendizaje automático. Una representación gráfica de esta red se muestra en la Ilustración 1-1.

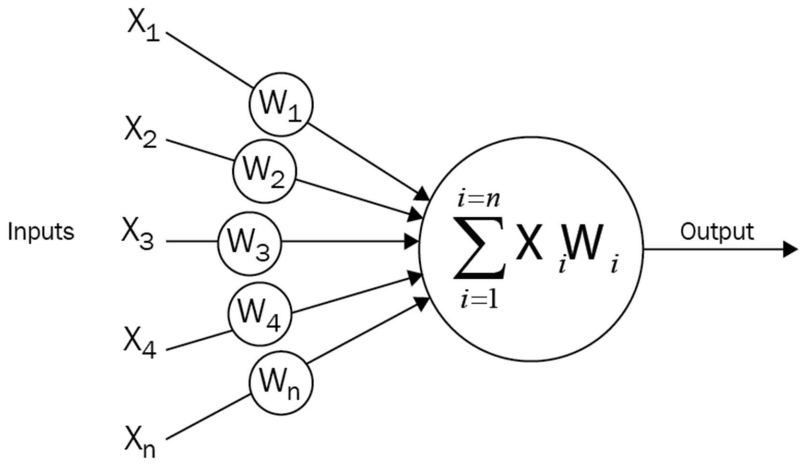

Ilustración 1-1 Modelo de una red neuronal

Una neurona artificial es una función que toma una entrada y produce una salida. La cantidad de neuronas que se utilizan depende de la tarea que se realice. Podría ser tan bajo como dos o tantos como varios miles. Existen numerosas formas de conectar neuronas artificiales para crear una CNN. Una de estas topologías que se usa comúnmente se conoce como red de feed-forward:

Cada neurona recibe entradas de otras neuronas. El efecto de cada línea de entrada en la neurona está controlado por el peso, el cual puede ser positivo o negativo. Toda la red neuronal aprende a realizar cálculos útiles para reconocer objetos mediante la comprensión del lenguaje. Ahora, podemos conectar esas neuronas en una red conocida como red de feed-forward. Esto significa que las neuronas en cada capa alimentan su salida a la siguiente capa hasta que obtengamos una salida final.

1.4.2 Redes Neuronales Multicapa

Una red neuronal multicapa es aquella que presenta más de una capa e interconexiones entre sí. Uno de los modelos más simples a usar es una red neuronal de dos capas, en donde se tiene una capa de entrada, una capa intermedia y una capa de salida. La capa intermedia se suele llamar capa oculta. La Ilustración 1-2 muestra un

modelo de red neuronal de dos capas. Si se tiene más de una capa esta se llama red neuronal de aprendizaje profundo o ***"deep neural network"***.

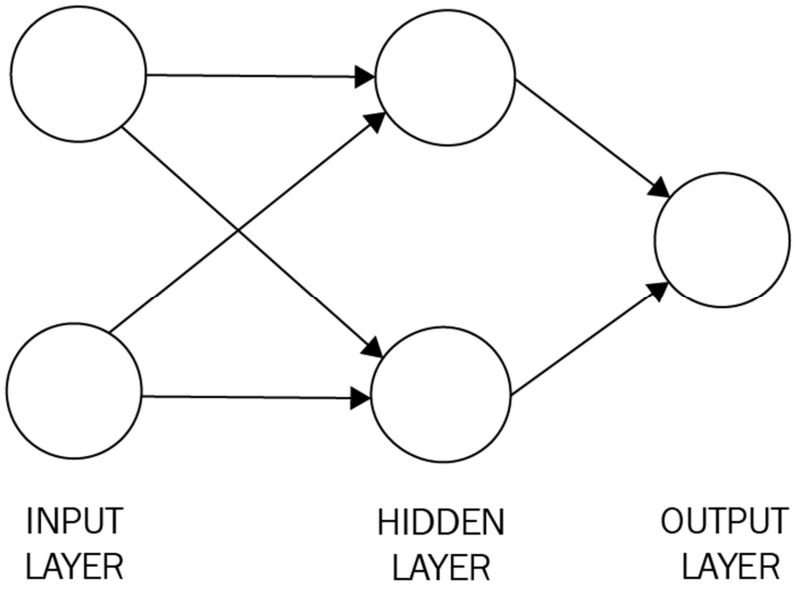

Ilustración 1-2 Modelo multicapas de una red neuronal

La entrada y salida de cada neurona en la capa oculta está conectada a cada neurona en la siguiente capa. Puede haber cualquier número de neuronas en cada capa dependiendo del problema. El ejemplo simple es el popular reconocimiento de dígitos escritos a mano que detecta un número, digamos 5. Esta red aceptará una imagen de 5 y arrojará 1 o 0. En donde 1 es para indicar que la imagen de hecho es un 5 y 0 de lo contrario. Una vez que se crea la red, debe ser entrenada. Podemos inicializar con pesos aleatorios y luego alimentar las muestras de entrada conocidas como el conjunto de datos de capacitación. Para cada muestra de entrada, verificamos la salida, calculamos la tasa de error y luego ajustamos los pesos de manera que cada vez que ve 5, emite 1 y para todo lo demás emite un cero. Este tipo de entrenamiento se denomina aprendizaje supervisado y el método de ajuste de los pesos se llama retropropagación o "***backpropagation"***.

Al construir modelos de redes neuronales artificiales, una de las consideraciones principales es cómo elegir las funciones de activación para capas ocultas y de salida. Las tres funciones de activación más comúnmente utilizadas son la función sigmoidea, la función de tangente hiperbólica y la unidad lineal rectificada (ReLU). La belleza de la función sigmoidea es que su derivada se evalúa en z y simplemente es z multiplicada por 1-menos z. Eso se presenta en la Ecuación 1-1:

$$^{dy}/_{dx} = \sigma(x)(1 - \sigma(x))$$

Ecuación 1-1 Ecuación de la función sigmoidea.

Esto nos ayuda a calcular de manera eficiente los gradientes utilizados en las redes neuronales. Si las activaciones de avance de la función logística para una capa determinada se mantienen en la memoria, los gradientes para esa capa en particular se pueden evaluar con la ayuda de una simple multiplicación y sustracción en lugar de implementar y reevaluar la función sigmoidea, ya que requiere exponenciación adicional La Ilustración 1-3 nos muestra la función de activación ReLU, que es cero cuando x <0 y luego lineal con pendiente 1 cuando x> 0.

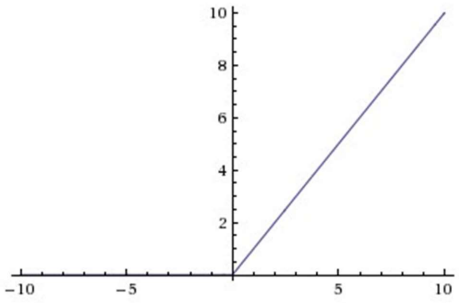

Ilustración 1-3 Función de activación de ReLU

1.4.3 Redes Neuronales Convolucionales

Las redes neuronales convolucionales, CNNs, o ConvNets, son bastante similares a las redes neuronales regulares. Están formados por neuronas con pesos se aprenden de los datos. Cada neurona recibe algunas entradas y realiza un producto escalar. Tienen una función de pérdida en la última capa completamente conectada. Y pueden usar una

función no lineal. Como se muestra en la sección 1.4.2, una red neuronal regular recibe datos de entrada como un único vector y pasa a través de una serie de capas ocultas. Cada capa oculta consiste en un conjunto de neuronas, en el que cada neurona es totalmente conectada a todas las otras neuronas en la capa anterior. Dentro de una sola capa, cada neurona es completamente independiente y no comparten ninguna conexión. La última capa totalmente conectada, también llamada capa de salida, contiene puntuaciones por clase en el caso de un problema de clasificación de imágenes. En general, hay tres capas principales en una ConvNet simple. Son la capa de convolución, la capa de agrupamiento y la capa completamente conectada.

Entonces, la principal diferencia en una CNN es que generalmente toma imágenes como entrada, esto nos permite codificar algunas propiedades en la red, reduciendo así el número de parámetros. En el caso de los datos de imágenes del mundo real, los CNN funcionan mejor que los Perceptrones de Capas Múltiples (MLP por sus siglas en ingles). Hay diferentes motivos para esto: para alimentar una imagen a un MLP, se convierte a la matriz de entrada en un vector numérico simple sin estructura espacial. Lo cual no tiene conocimiento de que estos números estén dispuestos espacialmente. Entonces, las CNN se crean por esta misma razón; es decir, para elucidar los patrones en datos multidimensionales. A diferencia de las MLP, las CNN entienden el hecho de que los píxeles de la imagen que están más cerca unos de otros están más relacionados que los píxeles que están más separados:

Las CNN difieren de las MLP en los tipos de capas ocultas que se pueden incluir en el modelo. Una ConvNet organiza sus neuronas en tres dimensiones: ancho, alto y profundidad. Cada capa transforma su volumen de entrada 3D en un volumen de salida 3D de neuronas usando funciones de activación.

1.4.4 Agrupamiento de datos

El agrupamiento de datos es una técnica común usada en machine learning para reducir la dimensionalidad, es decir supongamos que tengamos un conjunto de datos de imágenes que tengan un número de pixeles de 28 x 28, esto se representará como una matriz de 1 dimensión de 784 píxeles.

Al agrupar los datos, se pierde información sobre la estructura 2D de la imagen; Sin embargo, nuestros datos se simplifican. Con la ayuda de esto, todos nuestros datos de entrenamiento pueden estar contenidos en una matriz de formas (60,000, 784), donde la primera dimensión representa el número de imágenes de entrenamiento y la segunda representa el número de píxeles en cada imagen.

1.4.5 Backpropagation

En esta sección, se explicará la propagación inversa o backpropagation. Esta es una forma de calcular gradientes usando la regla de cadena. Comprender este proceso y sus sutilezas es fundamental para que se pueda comprender, desarrollar, diseñar y depurar de manera efectiva las redes neuronales.
En general, dada una función *f (x)*, donde x es un vector de entradas, queremos calcular el gradiente de f en x denotado por $\nabla(f(x))$. Esto se debe a que, en el caso de las redes neuronales, la función f es básicamente una función de pérdida (L) y la entrada x es la combinación de pesos y datos de entrenamiento.

El gradiente se toma en los parámetros de peso porque los datos de entrenamiento generalmente son fijos y los parámetros son variables sobre las que tenemos control. Usualmente calculamos el gradiente de los parámetros para que podamos usarlo para actualizar parámetros. Esto nos indica que, la propagación hacia atrás consiste en:

- Realizar de una operación de avance
- Comparar la salida del modelo con la salida deseada
- Cálculo del error
- Ejecutar de la operación de acondicionamiento de la red (*feedforward*) hacia atrás (*backpropagation*) para distribuir el error a cada uno de los pesos.
- Actualizar los pesos, para obtener un mejor modelo
- Continuar hasta que tengamos un modelo óptimo.

En la mayoría de los casos, la propagación hacia atrás se implementa en un framework, como TensorFlow. Sin embargo, no siempre es cierto que, simplemente agregando un número arbitrario de capas ocultas, la propagación hacia atrás funcionará

mágicamente en el conjunto de datos. De hecho, si la inicialización de los pesos es descuidada, estas funciones de no linealidad pueden saturarse y dejar de aprender. Eso significa que la pérdida de entrenamiento será plana y se negará a disminuir. Esto se conoce como el problema del gradiente de fuga.

Si la matriz de peso W, es inicializada con valores demasiado grandes, la salida de la matriz también podría tener un rango muy grande, que a su vez hará que todas las salidas en el vector z sean casi binarias: 1 o 0. Sin embargo, si este es el caso, entonces, *z * (1-z)*, que es el gradiente local de la no linealidad sigmoidea, se convertirá en cero (desaparecerá) para ambos casos, lo que hará que el gradiente tanto para x como para W también sea cero. El resto del pase hacia atrás también saldrá cero a partir de este punto en adelante a causa de la multiplicación en la regla de la cadena.

Otra función de activación no lineal es ReLU, que muestra el umbral de las neuronas en cero como se muestra a continuación. El pase hacia adelante y hacia atrás para una capa completamente conectada que utiliza ReLU en el núcleo incluye:

- *z = np.maximum (0, np.dot (W, x))* #Representando pase hacia adelante
- *dW = np.outer (z> 0, x)* #Representando el pase hacia atrás: gradiente local para W

Si esto se observa por un tiempo, se verá que si una neurona se sujeta a cero en el pase directo (es decir, z = 0, no dispara), entonces sus pesos obtendrán un gradiente cero. Esto puede llevar a lo que se llama el problema de ReLU muerto. Esto significa que, si una neurona ReLU desafortunadamente se inicializa de tal manera que nunca dispara, o si los pesos de una neurona alguna vez se eliminan con una gran actualización durante el entrenamiento en este régimen, en tales casos esta neurona permanecerá muerta permanentemente. A veces, incluso puedes reenviar todo el conjunto de entrenamiento a través de una red entrenada y finalmente darte cuenta de que una gran fracción (alrededor del 40%) de tus neuronas fue cero todo el tiempo.

El presente trabajo pretende establecer una nueva metodología de segmentación de lesiones por hiperintensidades de sustancia blanca cerebral, usando la arquitectura propuesta por Çiçek, Abdulkadir, Lienkamp, Brox, and Ronneberger (2016) para

aprendizaje profundo y siguiendo la metodología usada por Li, Jiang, et al. (2018), se logró segmentar lesiones con poca área axial pero con bastante volumen en el eje z, en una maquina con recursos computacionales moderados. Obteniendo una efectividad del 81% en el reconocimiento y clasificación de estas lesiones.

CAPÍTULO 2

2. METODOLOGÍA

2.1 Participantes

Para la segmentación de las lesiones en sustancia blanca cerebral (WMH) partimos de imágenes cerebrales FLAIR (Fluid Attenuated Inversion Recovery) obtenidas por resonancia magnética e imágenes estructurales potenciadas en T1. Estas imágenes pertenecen a un conjunto de datos proporcionadas por el Concurso del MICCAI 2017 WMH Challenge.

Para el cual se usaron imágenes adquiridas en tres escáneres diferentes de tres centros de estudios diferentes ubicados en Holanda, Ámsterdam y Singapur. En este concurso el conjunto de datos constaba de un total de 60 sujetos con imágenes de resonancia magnética FLAIR y T1 de tres escáneres diferentes junto con sus máscaras binarias, las cuales consisten en mapas binarios de datos que contienen la información de las lesiones segmentadas por manualmente, lo mismo que se conoce como *"ground truth"*.

Como se muestra en la Tabla 2-1, existe una gran diferencia en las configuraciones de adquisición; en los tamaños de vóxel particulares de las imágenes capturadas difieren significativamente entre los tres escáneres. Para cada sujeto, una imagen 3D ponderada en T1 y una división múltiple en 2D.

Dataset	Tamaño del Voxel (m^3)	Maquina Usada	Dimensiones de la imagen FLAIR	Dimensiones de la imagen T1
Utrech	0.96x0.95x3.00	3T Philips Achieva	240x240x48	240x240x48
Singapore	1.00x1.00x3.00	3T Siemens Trio Tim	252x232x48	256x232x48
GE3T	0.98x0.98x1.20	3T GE Signa	132x256x83	132x256x83

Tabla 2-1 Características del dataset de imágenes proporcionadas por el concurso MICCAI-2017

Dado que el estándar de referencia manual está definido en la imagen FLAIR, se generó una versión en 2D de varias secciones de la imagen T1 volviendo a muestrear la imagen 3D ponderada en T1 para que coincida con la del FLAIR. El conjunto de datos a usar en el trabajo actual consta de un total de 60 sujetos con imágenes de resonancia magnética FLAIR y T1 de tres escáneres diferentes. Logrando así que en la etapa de entrenamiento se va a usar el método de dejar un paciente afuera y entrenar con él resto.

Se usaron diferentes librerías para poder hacer el análisis y segmentación de las lesiones, todas estas fueron usadas en el ambiente de Python 2.7 y Anaconda como *package manager.* Las librerías usadas se muestran en la Tabla 2-2, junto con una breve descripción.

Librería	Versión	Descripción
Numpy	1.15	Presentada por Walt, Colbert, and Varoquaux (2011), permite hacer cálculos complejos usando estructuras numéricas de otros ambientes de programación más simples.
Tensorflow	1.9.0	Es una librería *open source* permite hacer cálculos computacionales complejos mediante diagramas de flujo de datos, usados para el aprendizaje profundo como lo muestra Abadi et al. (2016)
Tensorflow-gpu	1.9.	Parecida a Tensorflow esta se diferencia de que usa el GPU, para realizar los cálculos y el aprendizaje.
Keras	2.2.2	Es un API de alto nivel para redes neuronales, escrita en Python y capaz de correr usando Tensorflow, CNTK o Threano. Introducida por Chollet (2015), es un *framework* que permite simplificar las operaciones del aprendizaje profundo agregando una capa de abstracción entre el modelo y el procesamiento.
Nibabel	2.3.0	Esta librería permite hacer procesamiento de imágenes tridimensionales de resonancia magnética (Brett et al., 2016)

Nilearn	0.4.2	Tal como lo muestra Estève (2015), esta librería permite hacer minado de datos en imágenes cerebrales.
OpenCV	3.4.1	Cuando Bradski and Kaehler (2000), introdujeron OpenCV hicieron una librería que permitiera manipular imágenes, la misma fue usada para manipular imágenes en 2D para luego convertirse en 3D
Matplotlib	2.2.3	Esta librería permite hacer graficos 2D estadisticos se usó con la finalidad de poder visualizar los resultados obtenidos. (Hunter, 2007)
Scikit-learn	0.19.1	Este *framework* es uno de los más usados para aprendizaje profundo, presentado por Pedregosa et al. (2011) permite hacer de manera simple separaciones de dataset en entrenamiento y pruebas.
Scipy	1.1.0	Es una librería open source que permite el uso de Python como ambiente científico. (Jones, Oliphant, & Peterson, 2014)
SimpleITK	1.1.0	Librería para el manejo de imágenes de resonancia magnética, (Lowekamp, Chen, Ibáñez, & Blezek, 2013)
Cudann	7.1.2	Se uso para poder aprovechar la GPU de la máquina, tal como lo muestra Moreso (2015), se puede usar el GPU de las tarjetas gráficas NVIDIA

Tabla 2-2 Librerías usadas en la implementación del programa

2.1.1 Tensorflow

TensorFlow es un framework basado en Python, el cual se basa en grafos para realizar los cálculos. Cada estructura en TensorFlow, se representa como un grafo computacional, logrando así que los cálculos se realicen en paralelo. Los datos no se

almacenan como enteros, punto flotante, cadenas de caracteres u otros tipos de datos primitivos; sino que estos valores son encapsulados en un objeto llamado tensor. Esto permite tener un conjunto de valores primitivos agrupados en un arreglo de cualquier número de dimensiones. El rango de un tensor indica el número de dimensiones.

El programa principal de TensorFlow se basa en la idea de un grafo computacional. Esto es un grafo dirigido que consta de dos acciones principales:

- Construir el modelo computacional del grafo.
- Ejecutar el modelo computacional del grafo:

Cada grafo computacional se ejecuta dentro de una sesión; la cual es un entorno de tiempo de ejecución para el grafo computacional. Este asigna la CPU o GPU y mantiene el estado del tiempo de ejecución de TensorFlow.

2.1.2 Keras

Keras es una API de redes neuronales profundas de alto nivel en Python que se ejecuta en la parte superior de TensorFlow, CNTK o Theano.

Aquí hay algunos conceptos básicos que necesita saber para trabajar con Keras. TensorFlow es una biblioteca de aprendizaje profundo para computación numérica e inteligencia artificial. Es de código abierto y usa gráficos de flujo de datos para el cálculo numérico. Las operaciones matemáticas están representadas por nodos y matrices de datos multidimensionales; es decir, los tensores están representados por los bordes del grafo. Este marco es extremadamente técnico y, por lo tanto, probablemente sea difícil para los analistas de datos, científicos, entre otros. Por esto motivo aparece Keras, el cual hace que la codificación de redes neuronales sea simple. También funciona perfectamente en máquinas CPU y GPU.

Los modelos son la estructura de datos básicos de Keras. El modelo secuencial, que consiste en una pila lineal de capas, siendo este el tipo de modelo más simple. Para facilitar el análisis este proporciona funciones comunes, como fit, evaluate y compile. Una capa de Keras, es como una capa de red neuronal. Esto implica que hay capas conectadas, ***"max pool layers"*** y capas de activación. Se puede agregar una capa al modelo utilizando la función add de un modelo.

2.2 Módulos del programa:

Dentro de la metodología planteada se realizó un programa que sea robusto a los cambios de los parámetros de configuración que se pueden dar en máquinas de resonancia magnética, aunque es una meta utópica, se puede lograr probando con diferentes datasets. Teniendo en cuenta que se requieren más sujetos de prueba para poder hacer el algoritmo más robusto. Se dividirá el programa en tres módulos principales:

- Módulo de Preprocesamiento
- Módulo de Segmentación
- Módulo de Postprocesamiento

Hay que resaltar que el módulo de segmentación va a ser hecho en Python 2.7 usando la librería Tensorflow y Keras.

2.3 Módulo de Preprocesado

Las imágenes requieren de un tratamiento previo antes de poder ser usadas para entrenar a la red neuronal. Si estas fueran ingresadas tal como se obtienen de los escáneres de resonancia magnética se tendría una cantidad baja de detección de lesiones cerebrales. Hace menos de 5 años se usaban bastantes técnicas de preprocesado para poder lograr hacer la segmentación, esto se daba porque en técnicas como la que muestra Chancay et al. (2015), en donde se usa segmentación por umbral.

Las CNN por naturaleza son buenas para poder identificar y clasificar características de la imagen sin procesar, sin embargo, tal como lo muestra la Ilustración 2-1, se requiere que se potencie la imagen T1, la imagen FLAIR y que también se haga una corrección de bias para potenciar la sustancia blanca cerebral, esto fue realizado previamente y vino junto con el dataset de entrenamiento. Con esta finalidad se harán los métodos mostrados en este capítulo.

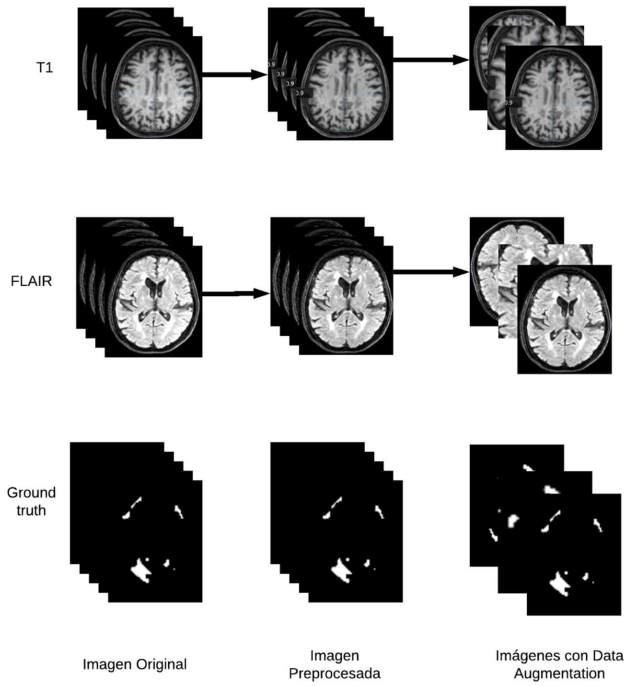

Ilustración 2-1 Preprocesamiento de las imágenes antes de ser usadas para el entrenamiento de la red

2.3.1 Imagen original

Las imágenes originales obtenidas del dataset provisto, presentan la característica de no ser del mismo tamaño. Esto implica que más cortes axiales, sagitales o coronales dependiendo del caso. Para poder hacer un agrupamiento de las imágenes y poderlas ingresar a un módulo de aprendizaje autónomo, estas deberán tener las mismas dimensiones caso contrario no se podrá entrenar a la red.

Para esto se tomaron 6 grupos de imágenes por sujeto, las imágenes FLAIR y T1 originales, el ground truth proveniente del conjunto de datos. Adicionalmente las imágenes FLAIR y T1 con el cráneo removido de la imagen, a estas en primera instancia solo se les removió el cráneo usando la función Bet2 de la librería FSL (Smith et al., 2004), tratando de alterar en lo menos posible la imagen original.

2.3.2 Preprocesamiento

Se buscó hacer un método de suavizado pasando un kernel Gaussiano por la imagen preprocesada dada por el concurso de tal manera que se reduzca el ruido que se puede haber introducido en la imagen.

Debido a que el recurso computacional es limitado, una imagen con una alta resolución puede ser difícil de procesar usando Deep Learning, debido a que el rendimiento va a estar fuertemente ligado al hardware computacional. Por esto se decidió usar imágenes de 128x128 de tal manera que el volumen tridimensional sea mucho más corto de analizar.

Para esto se usó la función de resize de OpenCV (Bradski & Kaehler, 2000), usando una interpolación cubica, reduciendo la imagen de entrada. Luego de esto las imágenes se concatenan una tras otra con la finalidad de poder hacer un stream continuo de datos para el modelo de aprendizaje el cual en el caso de la imagen T1 es convertido a float32 y en el caso de la FLAIR se deja igual.

Con este arreglo de imágenes tridimensionales se usa la función save de Numpy (Walt et al., 2011) para guardar cada dataset de cada máquina en un archivo diferente, el formato de estos archivos genera una extensión .”npy” el cual es un formato binario que comprime el dataset y puede ser solo interpretado por Numpy.

2.3.3 Data augmentation

El aumento del dataset, se hizo por cada imagen dentro del conjunto de imágenes, para cada uno se hicieron tres transformaciones: rotación, shearing y zoom. Tal como se muestra en la Ilustración 2-2.

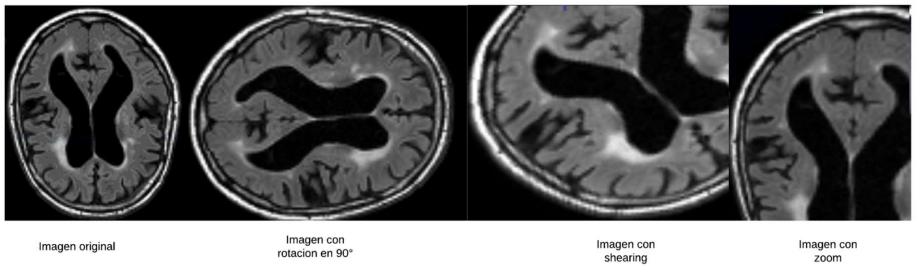

Ilustración 2-2 Imágenes después de hacer el aumento de la data

Para la rotación de las imágenes tridimensionales se tomó como centro las coordenadas medias en x, y; y se definió una matriz de rotación 2D en 90° grados. Luego de esto se usó la función wrapAffine de OpenCV (Bradski & Kaehler, 2000) tomando como entrada la imagen redimensionada.

Luego se hizo el shearing para lo cual se establecieron puntos fijos, estos servirán para poder preservar el volumen de la figura sólida y también cambiar las áreas de las figuras planas, después de hacer la transformación.

La última transformación fue hacer el zoom de la imagen ya redimensionada, logrando una imagen con más detalle en la sección a segmentar.

Este proceso de preprocesamiento se repite para cada imagen que se quiera agregar al sistema, incluida la máscara del ground truth.

2.4 Módulo de Segmentación de WMH:

Teniendo las imágenes preprocesadas en un espacio estándar, se tiene que realizar la segmentación de las lesiones de sustancia blanca. Con esta finalidad se hace el siguiente proceso

2.4.1 Escoger el tamaño de la muestra de prueba y entrenamiento

Usando el método de train_split proporcionado por scikit learn (Pedregosa et al., 2011), se estima el tamaño de la muestra de prueba y entrenamiento. Para esto se sigue la metodología provista por (Li, Jiang, et al., 2018), en el cual se deja un paciente afuera como conjunto de prueba y se entrena con todo el dataset sobrante. Esto implica repetir

el proceso de aprendizaje n veces, donde n significa el número de sujetos. Para cada iteración se toma n-1 sujetos para aprendizaje y 1 sujeto para prueba.

Se uso una configuración de 99% del dataset para aprendizaje y 1% para pruebas siendo este proceso rotativo.

2.4.2 Segmentación usando U-Net 3D

Para la segmentación automática, se usó una red neuronal convolucional, de tal manera que se pueda extraer la mayor cantidad de información de las imágenes FLAIR y T1. El conjunto de entrada se encuentra concatenado haciendo un arreglo de imágenes tridimensionales, el cual incluye las imágenes sin aumentar y las imágenes aumentadas; cada una con su respectiva mascara de segmentación.

Con la finalidad de tener un resultado más robusto y menos sesgado al momento de encontrar y clasificar lesiones, se hizo un shuffle del dataset de entrada de tal manera que las imágenes combinadas junto con sus máscaras sean organizadas de manera aleatoria.

Debido al limitante computacional, se hizo que todas las imágenes del dataset de entrenamiento y pruebas también tuvieran una dimensión extra para que pueden trabajar con la arquitectura presentada, esto implica; que se hicieron tensores de dimensión 128x128x48x1. Sin embargo, la arquitectura tiene un límite de 18 capas y 22 millones de parámetros entrenables, esto implico que se hace una reducción en el eje z también de tal manera que los tensores a usar en la red son de la forma 128x128x16x1. Teniendo así solo 16 cortes en el eje axial.

Se construyo la red neuronal usando la arquitectura presentada por (Çiçek et al., 2016); el cual hace una representación de U-Net 3D. En la Ilustración 2-3, se puede apreciar que la red consiste en una parte convolucional hacia abajo (lado izquierdo) y una parte convolucional hacia arriba (lado derecho).

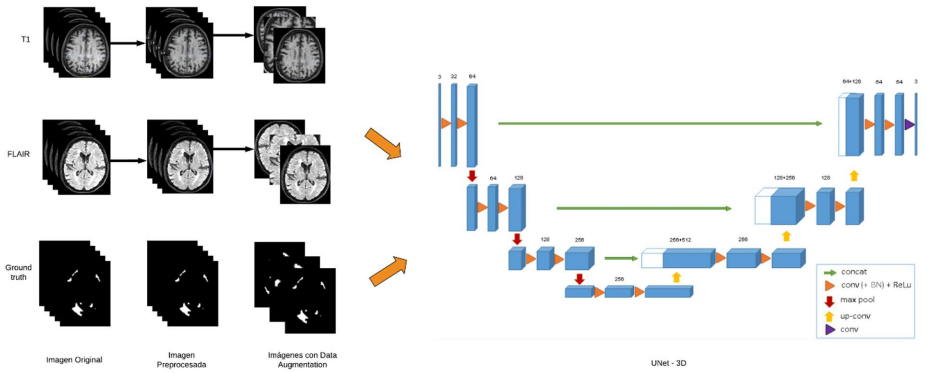

Ilustración 2-3 Arquitectura de la red U-Net 3D

El lado izquierdo tiene como objetivo extraer las características para ir clasificando a cada voxel en regiones WMH y no WMH. Y el lado derecho tiene como objetivo localizar las regiones WMH de manera más precisa.

La parte convolucional de abajo- consiste en dos capas de convolución 3 × 3, cada una seguida por una unidad lineal rectificada (ReLU) y una capa de agrupación máxima 2 × 2 para muestreo descendente. Para las dos primeras capas convolucionales, se utilizó un núcleo de tamaño 5 × 5 para manejar las diferentes transformaciones.

Cada paso en la parte convolucional hacia arriba involucra un muestreo ascendente de características seguidas de una capa convolucional 3 x 3 que reduce el número de canales de atributos y dos 3 x 3 capas de convolución seguidas de una capa ReLU. La concatenación se realizó entre convolucionales hacia abajo y hacia arriba como se muestra en la Ilustración 2-3 usando la línea verde. La inicialización aleatoria se utilizó para inicializar los pesos del modelo.

También el modelo se entrena usando dice loss como función de perdida, y como parámetro de métricas el factor f1, el cual por cada iteración va indicando cuanto es el valor de la función de costo (en este caso dice loss), cada epoch a usar va a ser repetido varias veces por la metodología de (Li, Jiang, et al., 2018) la cual consistia en dejar un paciente afuera de la parte de aprendizaje.

Para este enteramiento se usó 50 epochs y un batch_size de 1 por cada iteración teniendo un total de 60 iteraciones. A esto se le agregó un modelo de EarlyStopping de tal manera que si no se mejora los valores de la función de perdida el modelo decide si sigue entrenando o no. En el caso que ya no se desee seguir entrenando se agregó un ModelCheckpoint para que se guarden los pesos de los arcos dado un EarlyStopping.

Dentro del aprendizaje se entrenó a la red en diferentes momentos con difrentes datasets de entrenamiento para todos se usó el ground truth respetivo este fue orden de entrenamiento:

1. Imágenes FLAIR preprocesadas.
2. Imágenes T1 preprocesadas.
3. Imágenes FLAIR sin cráneo sin aumento de datos.
4. Imágenes T1 sin cráneo sin aumento de datos.
5. Imágenes FLAIR sin cráneo con aumento de datos
6. Imágenes T1 sin cráneo con aumento de datos

2.5 Módulo de postprocesado

El postproceso incluye dos aspectos:

2.5.1 Recorte o relleno de la segmentación obtenida con respecto al tamaño original

Aquí se realizó una operación inversa a el paso descrito en la Sección 2.3.1, debido a que se necesita tener las mismas características que la imagen segmentada, se debe cortar a al ground truth obtenido para poder tener congruencia de manera axial.

2.5.2 Rotación de las imágenes

Con el fin de tener las imágenes luego de ser segmentadas en el mismo espacio se le realiza una transposición a la imagen para alternar los ejes y que quede en el formato nifti estándar.

2.5.3 Guardado de las imágenes

Debido a que las imágenes segmentadas tienen solo 16 cortes axiales, no se puede mezclar con el dataset original, por esto se debe tomar a la imagen original, ground truth y guardarlas luego de haber hecho el corte axial respectivo. Teniendo así un nuevo dataset, el cual forma un subconjunto del dataset original.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

3.1 Métricas de evaluación y puntaje

Para poder evaluar la similitud entre el resultado obtenido y el resultado deseado se han usado cinco indicadores diferentes que fueron utilizados por los organizadores del desafío para comparar y clasificar los métodos por diferentes equipos; esas métricas evalúan la segmentación rendimiento en diferentes aspectos.

Tal como nos muestra Li, Jiang, et al. (2018), necesitamos asumir que dado un mapa de segmentación (ground truth) ***G*** y un mapa de segmentación ***P*** generado por un algoritmo, las cinco métricas de evaluación se definen de la siguiente manera:

3.2 Coeficiente de similitud de DICE (DSC)

$$DSC = \frac{2(G \cap P)}{|G| + |P|}$$

Ecuación 3-1 Coeficiente de similitud DICE

Esto mide la superposición en porcentaje entre G y P.

3.3 Distancia de Hausdorff (percentil 95)

La distancia de Hausdorff se define como:

$$H(G,P) = max\{\sup_{x \in G} \inf_{y \in P} d(x,y), \sup_{y \in P} \inf_{x \in G} d(x,y)\}$$

Ecuación 3-2 Distancia de Hausdoff

donde d (x, y) denota la distancia de x e y, sup indica el supremo e inf para el infimo.

Esto mide hasta qué punto dos subconjuntos de una métrica el espacio es el uno del otro. Tal como se usa en este desafío, se modifica para obtener una versión robusta utilizando el percentil 95 en lugar del máximo (Centésimo percentil) distancia.

3.4 Diferencia de volumen promedio (en porcentaje)

Sea VG y VP el volumen de regiones de lesión en ***G*** y ***P***, respectivamente. Entonces, la diferencia de volumen promedio (AVD) en porcentaje se define como:

$$AVD = \frac{|V_G - V_P|}{V_G}$$

Ecuación 3-3 Diferencia de volumen promedio

3.5 Sensibilidad para lesiones individuales (Recall)

Sea NG ser el número de lesiones individuales delineadas en ***G***, y NP ser el número de lesiones detectadas correctamente después de comparar ***P*** con ***G***. La lesión individual se define como un componente conectado en 3D. Entonces el recall para las lesiones individuales se define como:

$$Recall = \frac{N_P}{N_G}$$

Ecuación 3-4 Sensibilidad para lesiones individuales

3.6 Resultados obtenidos en el dataset de prueba

En esta sección, se mostrará los resultados obtenidos por el desempeño del algoritmo de segmentación tanto en el conjunto de datos de entrenamiento, como en el conjunto de pruebas extendidas y se comparan con los resultados obtenidos por el ground truth. Los resultados de la segmentación se evaluaron usando las cinco métricas descritas en la sección anterior.

Los hiperparámetros U-Net 3D se configuraron de la siguiente manera: el tamaño de lote para computar la perdida de aprendizaje se estableció en 1; la tasa de aprendizaje se estableció en 0,00002; el número de épocas se estableció en 50. El número de modelos en el conjunto se estableció en 3.

Con esos valores de aprendizaje se realizó el proceso de entrenamiento para esto la Tabla 3-1 muestra el resultado de los 5 coeficientes utilizados para realizar la verificación de la efectividad, de estos los valores de H95 y AVD son mejores a medida

que el valor es más pequeño, siendo el mínimo 0. ara los otros coeficientes se tiene que entre más alto el valor mejor la segmentación siendo el máximo 1.

Centro	DSC	H95	AVD	Recall	F1
Utrecht (n = 20)	0,784	4,419	9,087	0,364	0,499
Singapore (n = 20)	0,733	4,552	17,921	0,281	0,396
GE3T (n = 20)	0,782	3,948	11,351	0,541	0,632
Promedio	0,767	4,306	12,786	0,395	0,509

Tabla 3-1 Resumen de los resultados obtenidos por el dataset de entrenamiento.

Los resultados obtenidos indican que el aprendizaje es certero al momento de hacer las clasificaciones de las lesiones, tal como lo podemos comprobar por cada conjunto de prueba, cabe recordar que cada centro presenta imágenes con características diferentes. Viendo la Tabla 3-1 podemos apreciar que el promedio de las lesiones nos da resultados aceptables, aunque no sean ideales.

De esto se puede ver que el coeficiente DICE se tiene un valor cercano a 0.78 lo indica un 78% por ciento de exactitud, para los 3 centros estudiados. Sin embargo, la distancia de Hausdorff nos muestra una mejor exactitud en el modelo a usar, se puede decir que el resultado parece ser bueno, pero va a depender de la maquina a usar.

En la Ilustración 3-1, se observa que, al hacer el diagrama de cajas por cada uno de los centros para cada uno de los parámetros medidos, se tiene que en los primeros gráficos de la parte superior una tendencia aparece, en la diferencia promedio de volumen se tiene que el centro de Singapore presenta una desviación estándar mucho más grande que el centro de Utrecht, sin embargo, en el gráfico de la distancia de Hausdorff, se tiene que son casi equivalentes.

Esto implica que las imágenes generadas por el método de segmentación generan un espacio volumétrico similar al del ground truth, con la excepción que al sacar la distancia entre volúmenes este fluctúa, lo cual puede ser resultado de las lesiones

encontradas en dichos sujetos de prueba tal como lo muestra la Ilustración 3-2, las lesiones más pequeñas pueden estar ocurriendo como falsos negativos.

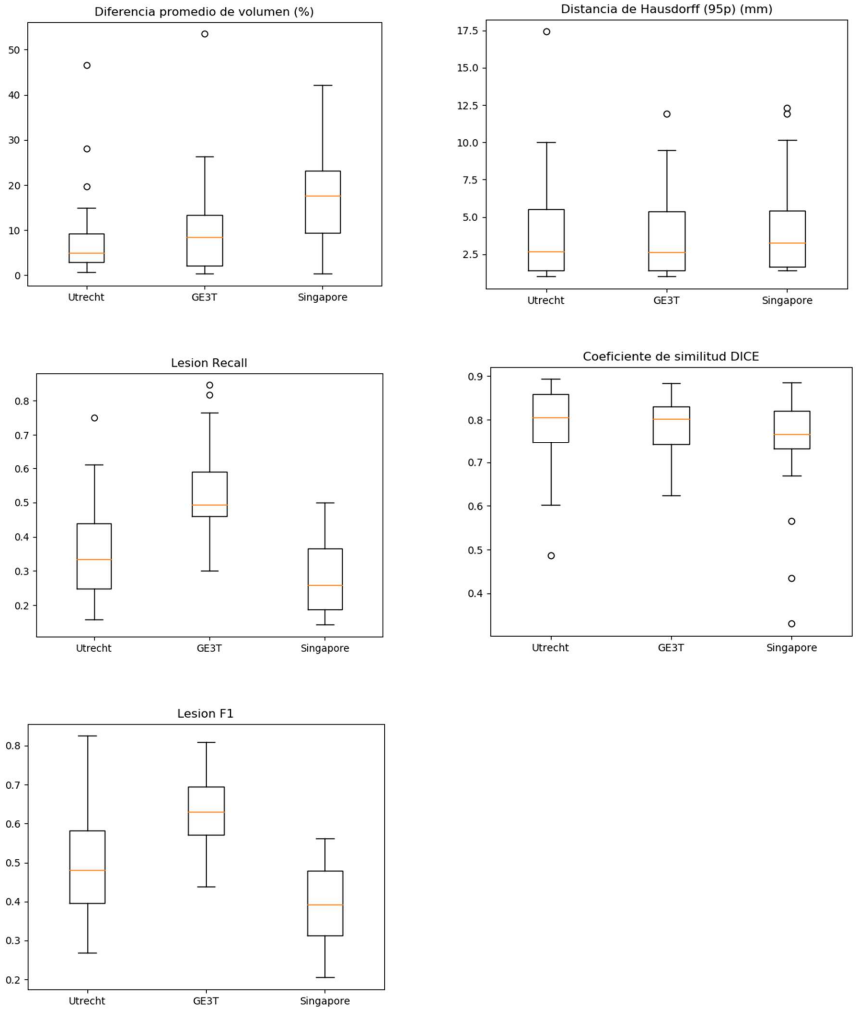

Ilustración 3-1 Diagramas de caja de los diferentes centros, en donde se mide un coeficiente diferente por gráfico.

Por otro, continuando con el análisis se puede apreciar también de la Ilustración 3-1, como el coeficiente de similitud DICE, es bastante homogéneo entre los diferentes

centros de estudio, así también como su desviación estándar, lo cual nos indica que nuestro modelo presenta una exactitud del casi 81%. Esta principal distinción se da al comparar las máscaras de manera manual como se da en la Ilustración 3-2.

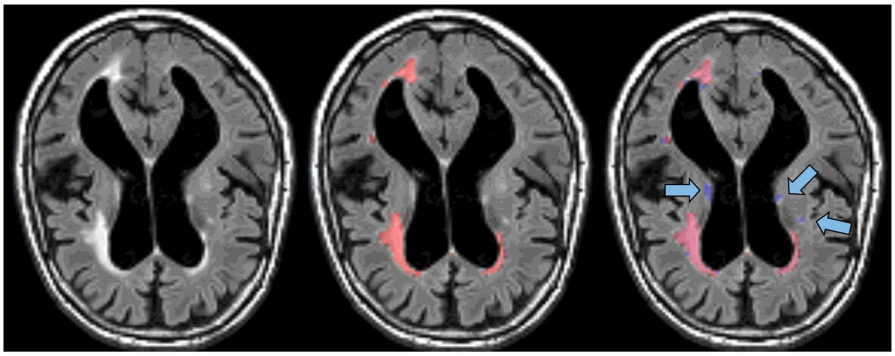

Ilustración 3-2 Comparación entre la imagen original, la máscara obtenida y el ground truth

Como se observa en la Ilustración 3-2 el principal inconveniente del método desarrollado es detectar las lesiones pequeñas en espacio plano como lo muestran los análisis de diagrama de caja, debido a que muchas lesiones pueden tener un volumen pequeño o representan un pequeño espacio en el eje z, puede hacer que el modelo 3D no lo detecte como una lesión. Lo cual se puede visualizar en la Ilustración 3-1, en el gráfico del coeficiente F1, en donde se ve que la dispersión de datos para el dataset de Singapore es bajo indicando que probablemente las lesiones en ese conjunto de datos sean mucho menores o con menor volumen que las lesiones observadas en Utrecht.

3.7 Análisis de una imagen a través de la red

En esta sección se va a presentar qué pasa cuando una imagen ingresa al modelo de U-Net 3D para entender mejor los resultados obtenidos en la sección anterior. En la sección 2.3.1, se mostró como restricción computacional el redimensionamiento de la imagen a una resolución de 128x128, siendo el tensor de entrada 128x128x16x1. Debido a que así mismo se reduce el número de cortes axiales a 16.

La resolución de las imágenes es clave al momento de hacer la segmentación debido al nivel de detalle que se tiene esto también aplica al eje z, ya que es aquí donde

va a ser muy buena la red para detectar lesiones. La Ilustración 3-3 nos muestra por capa como se va transformando la imagen a lo largo de la arquitectura, en lo que podemos apreciar que dentro de las convoluciones que hace se resaltan las lesiones con más volumen en el eje axial, de tal manera que es mejor al momento de clasificarlas. Así mismo esto mejora si la lesión tiene volumen en el eje z.

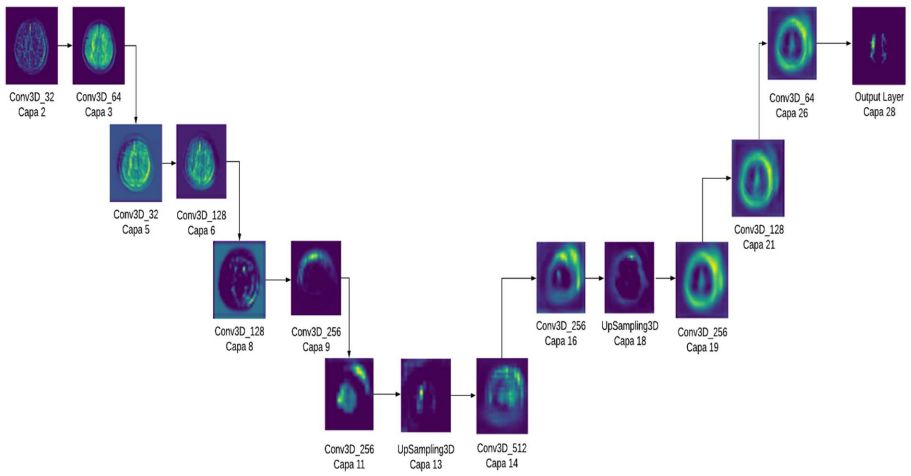

Ilustración 3-3 Transformaciones de una imagen dentro de la red.

3.8 Análisis de viabilidad

El principal componente en el análisis de viabilidad presentado va a ser los costos, del procesamiento computacional, en este trabajo se hará mediante la adquisición del hardware necesario para poder correr la simulación y el proceso de aprendizaje. Para este trabajo se ha tomado como referente una maquina con las características mostradas en la Ilustración 3-4.

Ilustración 3-4 Características de la máquina usada

Debido a esta restricción se limitó el modelo de la red y del dataset de entrada a usar tensores de 128x128x16x1, de un máximo teórico de 240x240x48x1. Sin embargo, para poder hacer un modelo de aprendizaje con dichas imágenes de entradas se necesitaría una instancia de computo mayor.

El tiempo total de aprendizaje fue de 2 días y 10 horas usando los 4 GB de video que provee la maquina usada en la Ilustración 3-4, sin embargo, existen instancias de alquiler en la nube con un valor por minuto, como la instancia p3.8xlarge de AWS que tiene un valor de 12,24 $USD (Amazon, 2014).

El número total de parámetros entrenables fue de 22398465, lo cual es bastante fuerte de entrenar, sin embargo, con las restricciones el proyecto es considerablemente viable debido a que se usa una máquina de gama media para poder hacer los cálculos.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

En este capítulo se va a discutir porque se hicieron las elecciones de arquitectura e hiperparametros para configurar al modelo de aprendizaje profundo, así también como fue el rendimiento, las pruebas realizadas y las complicaciones obtenidas durante el proceso de desarrollo.

4.1 Elección de U-Net3D como arquitectura

U-Net es una arquitectura de red neuronal convolucional presentada por Çiçek et al. (2016) que ha permitido a muchos investigadores hacer modelos de aprendizaje en imágenes de resonancia magnética, se han encontrado limitantes para las arquitecturas que analizan imágenes en 3D, debido a que no son muy buenos para detectar lesiones pequeñas como las hiperintensidades de sustancia blanca cerebral (Jin, Xu, Harrison, & Mollura, 2018), sin embargo esto va a depender bastante del modelo de aprendizaje usado. Li, Jiang, et al. (2018) nos muestra, más que la dependencia en la arquitectura, una metodología de aprendizaje que se puede utilizar para potenciar los resultados. Por esto en el trabajo actual, se usó una metodología de aprendizaje adaptativo en el cual se entrena varias veces a la red de tal manera que aprenda a identificar las posibles lesiones.

U-Net3D tiene la principal ventaja de poder detectar las características tridimensionales, aunque se puede usar una arquitectura 2D y realizar cortes axiales de las imágenes, como lo han realizado diferentes autores (Li, Zhang, Muehlau, Kirschke, & Menze, 2018; Orbes-Arteaga et al., 2018). Como se muestra en la Ilustración 4-1, el algoritmo ha detectado de manera correcta lesiones pequeñas en el eje axial pero que tienen volumen en el eje z, esto significa que las lesiones encontradas fueron segmentadas correctamente debido a que el nivel volumétrico permite saber la forma que tienen las lesiones tienen antes de incurrir en la identificación axial.

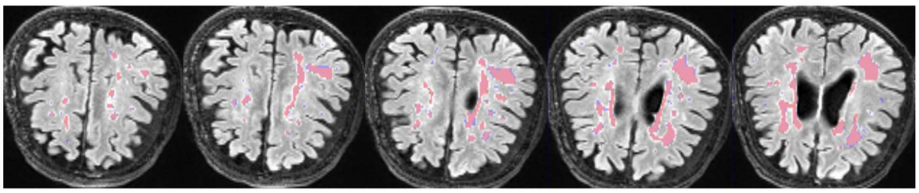

Ilustración 4-1 Detección de lesiones en diferentes cortes axiales. Se muestra en rojo las lesiones detectadas por el algoritmo y en azul el ground truth.

Esto se corrobora en la Ilustración 4-2, aquí se presentan problemas o falsos negativos cuando las lesiones son pequeñas en el eje axial pero no tienen mucha profundidad. Así mismo se presentan falsos positivos cuando las lesiones tienen bastante información en el eje z, lo que hace que el sistema prediga que la forma de la lesión va a tener aparición en las capas superiores y eso puede ser confundido fácilmente con la intensidad normal de una imagen

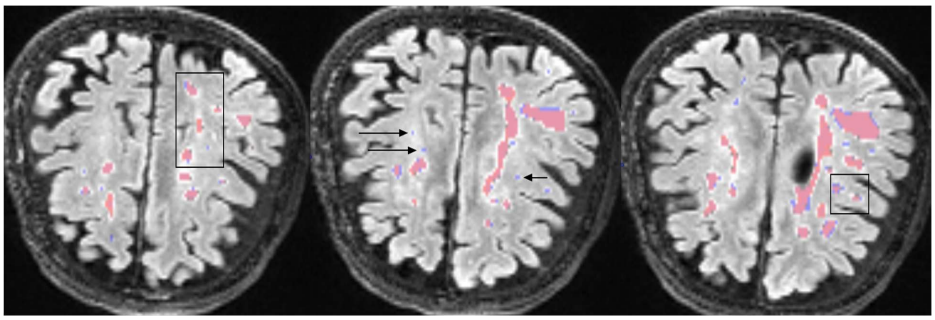

Ilustración 4-2 Detección de falsos positivos. Se puede apreciar en la imagen segmentos sin detectar, falsos positivos y falsos negativos. Rojo indica segmentación automatica y azul segmentación real.

Poder extraer las formas de los volúmenes segmentados, las características 3D de las imágenes nos da un valor de ventaja para poder hacer la extracción correcta de las mismas. Aunque esto también implica en un mayor impacto al momento de hacer los cálculos computacionales, debido al gran tamaño de las imágenes.

4.2 Dificultades alcanzadas

Durante el proceso de escoger la arquitectura correcta se probó con diferentes configuraciones siendo las más relevantes las presentadas por Li, Jiang, et al. (2018) y Çiçek et al. (2016), sin embargo la estructura 2D mostro inconvenientes de memoria, lo que logro que se hiciera un mal proceso de aprendizaje porque hubo que restringir los hiperparametros para hacer funcionar la arquitectura. La Ilustración 4-3 muestra como la estructura 2D también introducía mucho ruido a la imagen reconstruida y también confundía las lesiones con el líquido cefalorraquídeo.

Ilustración 4-3 Imágenes generadas usando U-Net2D como método de segmentación.

Luego manteniendo la estructura 2D se redujo el *batch size* a 1 y el número de *epochs* a 50, logrando una efectividad del 70%, por lo cual se probó, usando el mismo número de hiperparametros la arquitectura 3D. La efectividad del modelo casi no se vio alterada, pero se vio mejor identificación de las lesiones volumétricas. Luego de esto se decidio usar una metodología más iterativa, en la cual el método seria con un *batch size* de 1 y un numero de *epochs* de 50, por cada uno de los sujetos de prueba usando la metodología *"leave one paciente out"* presentada por (Li, Jiang, et al., 2018). Ese método se repitió 3 veces debido a que se entrenó primero con las imágenes FLAIR, luego con las imágenes T1 y luego con las imágenes sin cráneo para tener una mejor precisión en el modelo final.

Se logro establecer un método efectivo para la segmentación automática de lesiones en la sustancia blanca cerebral, por medio del uso de una red neuronal convolucional de 28 capas.

4.3 Conclusiones

- El método y la metodología aquí planteadas pueden ser usadas en una maquina con recursos limitados obteniendo una eficacia del 81%
- El sistema propuesto mostro un buen desempeño en diferentes escáneres y protocolos usados.
- No se puede decir que se hizo un análisis exhaustivo de diferencias entre las arquitecturas 3D y 2D, lo que implica que se necesitaría verificar con mayor detalle porque un sistema actuó mejor que el otro y porque los resultados lo corroboran.
- El método empleado muestra la alta eficacia de las redes neuronales convolucionales al momento de realizar la tarea de segmentación automática, de tal manera que muestra potencial para pruebas reales.

4.4 Recomendaciones

- Este trabajo queda abierto a trabajos futuros, debido a que se tiene que poder resolver la identificación de lesiones pequeñas de poca profundidad, así mismo, aunque los resultados son buenos pueden ser mejores, si se optimiza la asignación del uso del GPU el rendimiento puede ser mucho mayor.
- Debido existe más de una marca de máquinas de imágenes por resonancia magnética en el mercado, realizar un algoritmo valido para todas es problemático, hay que recordar que una restricción del problema fue perder información de capas axiales y tomar solo las 16 capas principales.
- Se debería probar la eficacia del algoritmo en un recurso computacional mayor que permita tomar las imágenes 3D y procesarlas sin tener que tomar solo un fragmento.

BIBLIOGRAFÍA

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., . . . Isard, M. (2016). *Tensorflow: a system for large-scale machine learning.* Paper presented at the OSDI.

Amazon, A. (2014). Amazon.

Bradski, G., & Kaehler, A. (2000). OpenCV. *Dr. Dobb's journal of software tools*.

Brett, M., Hanke, M., Cipollini, B., Côté, M.-A., Markiewicz, C., Gerhard, S., . . . Kastman, E. (2016). nibabel: 2.1. 0. *Zenodo*.

Caligiuri, M. E., Perrotta, P., Augimeri, A., Rocca, F., Quattrone, A., & Cherubini, A. (2015). Automatic Detection of White Matter Hyperintensities in Healthy Aging and Pathology Using Magnetic Resonance Imaging: A Review. *Neuroinformatics, 13*(3), 261-276. doi: 10.1007/s12021-015-9260-y

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). *3D U-Net: learning dense volumetric segmentation from sparse annotation.* Paper presented at the International Conference on Medical Image Computing and Computer-Assisted Intervention.

Chancay, O., Haro, T., Yapur, M., Alvarado, R., Pastor, M., & Loayza, F. (2015). Nuevo Biomarcador en la Enfermedad de Parkinson Mediante el Análisis y Cuantificación de Lesiones Cerebrales en Secuencias Flair Obtenidas por Resonancia Magnética (ACL-Tool). *Revista Tecnológica-ESPOL, 28*(5).

Chollet, F. (2015). Keras: Deep learning library for theano and tensorflow. *URL: https://keras. io/k, 7*(8).

Dalca, A. V., Balakrishnan, G., Guttag, J., & Sabuncu, M. R. (2018). Unsupervised Learning for Fast Probabilistic Diffeomorphic Registration. *arXiv preprint arXiv:1805.04605*.

Estève, L. (2015). *Big data in practice: the example of nilearn for mining brain imaging data.* Paper presented at the Scipy 2015.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90-95.

Jin, D., Xu, Z., Harrison, A. P., & Mollura, D. J. (2018). *White matter hyperintensity segmentation from T1 and FLAIR images using fully convolutional neural networks enhanced with residual connections.* Paper presented at the Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on.

Jones, E., Oliphant, T., & Peterson, P. (2014). {SciPy}: open source scientific tools for {Python}.

Lampe, L., Kharabian-Masouleh, S., Kynast, J., Arelin, K., Steele, C. J., Löffler, M., . . . Bazin, P.-L. (2017). Lesion location matters: The relationships between white matter hyperintensities on cognition in the healthy elderly. *Journal of Cerebral Blood Flow & Metabolism*, 0271678X17740501.

Li, H., Jiang, G., Wang, R., Zhang, J., Wang, Z., Zheng, W.-S., & Menze, B. (2018). Fully Convolutional Network Ensembles for White Matter Hyperintensities Segmentation in MR Images. *arXiv preprint arXiv:1802.05203*.

Li, H., Zhang, J., Muehlau, M., Kirschke, J., & Menze, B. (2018). Multi-Scale Convolutional-Stack Aggregation for Robust White Matter Hyperintensities Segmentation. *arXiv preprint arXiv:1807.05153*.

Lowekamp, B. C., Chen, D. T., Ibáñez, L., & Blezek, D. (2013). The design of SimpleITK. *Frontiers in neuroinformatics, 7*, 45.

Maniega, S. M., Hernández, M. C. V., Clayden, J. D., Royle, N. A., Murray, C., Morris, Z., . . . Bastin, M. E. (2015). White matter hyperintensities and normal-appearing white matter integrity in the aging brain. *Neurobiology of aging, 36*(2), 909-918.

Moreso, V. G. (2015). *Adaptación de algoritmos de aprendizaje automático para su ejecución sobre GPUs.*

Orbes-Arteaga, M., Sørensen, L., Modat, M., Cardoso, M. J., Ourselin, S., Nielsen, M., & Pai, A. (2018). Simultaneous synthesis of FLAIR and segmentation of white matter hypointensities from T1 MRIs.

Pandey, R. K., Vasan, A., & Ramakrishnan, A. (2018). Segmentation of Liver Lesions with Reduced Complexity Deep Models. *arXiv preprint arXiv:1805.09233*.

Pantoni, L. (2010). Cerebral small vessel disease: from pathogenesis and clinical characteristics to therapeutic challenges. *The Lancet Neurology, 9*(7), 689-701. doi: 10.1016/S1474-4422(10)70104-6

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research, 12*(Oct), 2825-2830.

Respino, M., Kuceyeski, A., Hoptman, M., Victoria, L., Scult, M., Chowdhury, N., . . . Gunning, F. (2018). T111. Changes in Connectivity Induced by White Matter Hyperintensities Predict Dysexecutive Behaviors in Late-Life Depression. *Biological Psychiatry, 83*(9), S171.

Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E., Johansen-Berg, H., . . . Flitney, D. E. (2004). Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage, 23*, S208-S219.

Walt, S. v. d., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering, 13*(2), 22-30.

Wardlaw, J. M., Smith, E. E., Biessels, G. J., Cordonnier, C., Fazekas, F., Frayne, R., . . . Dichgans, M. (2013). Neuroimaging standards for research into small vessel disease and its contribution to ageing and neurodegeneration. *The Lancet Neurology, 12*(8), 822-838. doi: 10.1016/S1474-4422(13)70124-8