Detección automática de lesiones en EM usando técnicas de Machine Learning.

Juan Esteban Cardona Osorio

Trabajo de grado presentado para optar al título de Bioingeniero

Asesores
John Fredy Ochoa Gómez, Doctor (PhD) en Ingeniería Electrónica
Laura Monsalve Cartagena, Bioingeniera

Universidad de Antioquia
Facultad de Ingeniería
Bioingeniería
Medellín, Antioquia, Colombia
2024

Cita	(Cardona Osorio, 2024)
Referencia **Estilo APA 7 (2020)**	Cardona Osorio, JE. (2024). *Detección automática de lesiones en EM usando técnicas de Machine Learning.* Trabajo de grados profesional, Bioingeniería, Universidad de Antioquia, Medellín, Antioquia, Colombia 2024.

Agradecimientos

Antes que nada, quiero expresar mi más sincero agradecimiento a mi familia, que siempre ha estado a mi lado, brindándome su apoyo incondicional en mi camino profesional y personal. A mis asesores, John Fredy Ochoa Gómez y Laura Monsalve Cartagena, les agradezco los conocimientos, la paciencia y los consejos que me brindaron durante la realización de este trabajo. Su apoyo fue fundamental para que pudiera culminarlo con éxito. A la empresa CEDMED S. A. S, les agradezco por brindarme las herramientas necesarias para desarrollar este proyecto y por darme la oportunidad de culminar esta etapa profesional. Por último, quiero agradecer a mis docentes y compañeros de la universidad, que a lo largo de mi camino por la institución me aportaron sus conocimientos, virtudes y vivencias.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

RM Resonancia magnética.

EM Esclerosis múltiple.

LR Learnig rate.

FN Falsos Negativos.

FP Positivo Verdadero.

TP Positivo verdadero.

TN Negativo Verdadero.

IoU índice de similitud de unión.

CNN Red neuronal convolucional.

Resumen

La esclerosis múltiple (EM) es una enfermedad del sistema nervioso central que afecta a millones de personas en todo el mundo. Las lesiones de EM son áreas de daño en el cerebro y la médula espinal que causan una variedad de síntomas. El diagnóstico y la monitorización de la EM se basan en imágenes de resonancia magnética. La segmentación de lesiones de EM es un proceso manual que requiere un alto nivel de experiencia. Los radiólogos deben identificar las lesiones en las imágenes de resonancia magnética y dibujar un contorno alrededor de cada lesión. Este proceso puede ser lento y laborioso, y es susceptible a errores. El sistema propuesto utiliza una red neuronal convolucional (CNN) para identificar las características de las lesiones de EM en las imágenes; se pueden entrenar con un conjunto de datos de imágenes etiquetadas, que contienen información sobre la ubicación y el tamaño de las lesiones. Entrenando el modelo nnUnet con la configuración Fulles hasta 50 épocas y con reconstrucciones del plano axial de secuencias Flair, T2 y T1 preprocesadas y registradas, se obtiene un coeficiente de Sorensen-Dice de 0.14 para la validación, el cual se logró incrementar a un 0.59 al utilizar la configuración 2D de nnUnet, aumentar el número de épocas a 150 y de utilizar 2060 cortes en el plano axial por lo que el modelo nnUnet es una herramienta prometedora para la segmentación de lesiones de EM puesto que el modelo mostró una sensibilidad del 69, 98% en el conjunto de datos de validación.

Palabras clave: esclerosis múltiple, imágenes de resonancia magnética, segmentación, redes neuronales convolucionales, aprendizaje automático, nnUnet.

Abstract

Multiple sclerosis (MS) is a disease of the central nervous system that affects millions of people around the world. MS lesions are areas of damage in the brain and spinal cord that cause a variety of symptoms. Diagnosis and monitoring of MS is based on magnetic resonance imaging. EM lesion segmentation is a manual process that requires a high level of expertise. Radiologists must identify lesions on MRI images and draw an outline around each lesion. This process can be slow and laborious and is susceptible to errors. The proposed system uses a convolutional neural network (CNN) to identify the characteristics of MS lesions in images; They can be trained with a dataset of labeled images, which contain information about the location and size of lesions. By training the nnUnet model with the Fulles configuration up to 50 epochs and with reconstructions of the axial plane of preprocessed and registered Flair, T2 and T1 sequences, a Sorensen-Dice coefficient of 0.14 was obtained for validation, which was increased to 0.59 When using the 2D configuration of nnUnet, increase the number of epochs to 150 and use 2060 slices in the axial plane so the nnUnet model is a promising tool for segmentation of MS lesions since the model showed a sensitivity of 69 , 98% in the validation data set.

Keywords: multiple sclerosis, magnetic resonance imaging, segmentation, convolutional neural networks, machine learning, nnUnet.

I. Introducción

La esclerosis múltiple (EM) es una enfermedad crónica del sistema nervioso central que afecta a millones de personas en todo el mundo. Las lesiones de EM son áreas de daño en el cerebro y la médula espinal que pueden causar una variedad de síntomas, como fatiga, debilidad muscular, problemas de equilibrio, coordinación, trastornos de la visión y problemas cognitivos (García-Lorenzo et al., 2013).

El diagnóstico de la EM se basa en la historia clínica, los exámenes físicos y las pruebas de imagen, como las imágenes de resonancia magnética que son una herramienta importante para el diagnóstico y la monitorización de la EM, ya que pueden mostrar las lesiones cerebrales y su evolución (Buccafusca, 2014; García-Lorenzo et al., 2013).

La segmentación de lesiones de EM es un proceso manual que requiere un alto nivel de experiencia y pericia. Los radiólogos deben identificar las lesiones en las imágenes de resonancia magnética y dibujar un contorno alrededor de cada lesión. Este proceso puede ser lento y laborioso, y es susceptible a errores (García-Lorenzo et al., 2013).

El proyecto de segmentación de lesiones de esclerosis múltiple usando redes convolucionales, pretende desarrollar un sistema automático que identifique y segmente las lesiones de EM en imágenes de resonancia magnética (Boutet et al., 2021).

La segmentación automática de lesiones de EM tiene el potencial de mejorar la precisión y la eficiencia del diagnóstico. Los sistemas automáticos pueden segmentar las lesiones de forma más rápida y precisa que los humanos, lo que puede ayudar a los radiólogos a diagnosticar la EM de forma más precisa y a realizar un seguimiento de la enfermedad de forma más eficiente (Boutet et al., 2021).

II. Objetivos

A. Objetivo general.

Evaluar métodos automatizados para detectar cambios en la carga de lesiones en pacientes con EM utilizando redes neuronales convolucionales (CNNs) en la segmentación de imágenes de resonancia magnética.

B. Objetivos específicos.

- Identificar herramientas de procesamiento de imágenes que permitan segmentar lesiones en EM en repositorios de código abierto.
- Comparar los diferentes modelos de segmentación de lesiones en pacientes con EM partiendo de los resultados de validación.
- Mejorar el modelo de segmentación en el cambio de la carga de las lesiones de EM mediante el sobre entrenamiento y ajuste de hiperparámetros de redes existentes.

III. Marco teórico

A. Esclerosis múltiple.

La esclerosis múltiple (EM) es una enfermedad crónica autoinmune que afecta al sistema nervioso central (SNC). El SNC está formado por el cerebro, la médula espinal y los nervios ópticos. La EM causa inflamación y daño a la mielina, una capa protectora que rodea las neuronas. El daño a la mielina interrumpe la transmisión de señales entre las neuronas, lo que puede causar una variedad de síntomas, que varían en gravedad y frecuencia. Los síntomas más comunes incluyen fatiga, debilidad muscular, problemas de equilibrio y coordinación, trastornos de la visión, problemas cognitivos y depresión (Buccafusca, 2014).

La causa exacta de la EM es desconocida, pero se cree que es una combinación de factores genéticos y ambientales. La EM es más común en mujeres que en hombres, y suele aparecer entre los 20 y los 40 años. No existe cura para la EM, pero hay tratamientos disponibles que pueden ayudar a controlar los síntomas y ralentizar la progresión de la enfermedad. Los tratamientos incluyen medicamentos modificadores de la enfermedad, inyecciones de esteroides y rehabilitación (Buccafusca, 2014).

La EM es una enfermedad compleja que puede tener un impacto significativo en la vida de las personas afectadas. Sin embargo, con el tratamiento adecuado, las personas con EM pueden llevar una vida plena y productiva (García-Lorenzo et al., 2013) .

B. Resonancia magnética.

La resonancia magnética (RM) es una técnica de imagen médica que utiliza campos magnéticos y ondas de radiofrecuencia para crear imágenes detalladas del interior del cuerpo. La RM es una técnica no invasiva que no requiere cirugía ni anestesia (García-Lorenzo et al., 2013).

El funcionamiento de la RM se basa en que los átomos de hidrógeno, los más abundantes en el cuerpo, absorben y emiten ondas de radiofrecuencia en un campo magnético. Un escáner de RM utiliza un potente campo magnético para alinear los átomos de hidrógeno en el cuerpo. Luego, se utilizan ondas de radiofrecuencia para perturbar la alineación de los átomos. Cuando los átomos se alinean de nuevo, emiten ondas de radio detectadas por el escáner (Buccafusca, 2014).

Las imágenes de RM se utilizan para diagnosticar y evaluar una amplia gama de afecciones médicas, incluyendo lesiones cerebrales y de la médula espinal, cáncer, enfermedades cardíacas, enfermedades del sistema digestivo y enfermedades musculoesqueléticas (Buccafusca, 2014; García-Lorenzo et al., 2013).

La RM es una técnica segura y eficaz que proporciona imágenes detalladas del interior del cuerpo. Es una herramienta valiosa para los médicos que necesitan diagnosticar y tratar afecciones médicas (Buccafusca, 2014).

C. Secuencias de adquisición.

Las secuencias de RM son imágenes que utilizan campos magnéticos y ondas de radiofrecuencia para visualizar diferentes tipos de tejidos y estructuras del cuerpo. Las secuencias T1 y T2 son las más comunes, y muestran la estructura y la función del tejido, respectivamente. La secuencia FLAIR son secuencias potenciadas en T2 que usa una técnica de supresión de la grasa para mejorar la visualización de tejidos con alto contenido de líquido. Las imágenes FLAIR son útiles para detectar la inflamación y los tumores, especialmente en el cerebro y la médula espinal (Boutet et al., 2021).

La técnica de supresión de grasa de la secuencia FLAIR se logra mediante la aplicación de una secuencia de inversión a los tejidos con alto contenido de grasa. Esta secuencia de invierte la dirección de precesión de los núcleos de hidrógeno en los tejidos con alto contenido de grasa. Cuando se aplica una secuencia de recuperación, los tejidos con alto contenido de grasa no absorben tanta energía como los tejidos con bajo contenido de grasa. Esto hace que los tejidos con alto contenido de grasa aparezcan negros u oscuros en las imágenes FLAIR (Boutet et al., 2021).

La secuencia T1 es un tipo de imagen de resonancia magnética (RM) que se utiliza para visualizar la estructura del tejido. Las imágenes T1 se basan en el tiempo que tarda un núcleo de hidrógeno en volver a su estado de equilibrio después de ser perturbado por un campo magnético externo. Los tejidos con tiempos de relajación T1 largos aparecen brillantes en las imágenes ponderadas en T1, mientras que los tejidos con tiempos de relajación T1 cortos aparecen oscuros. Por ejemplo, la grasa tiene un tiempo de relajación T1 largo, por lo que aparece brillante en las imágenes ponderadas en T1. El agua tiene un tiempo de relajación T1 corto, por lo que aparece oscura en las imágenes ponderadas en T1 (Boutet et al., 2021).

La secuencia T2 es un tipo de imagen de resonancia magnética (RM) que se utiliza para visualizar la función del tejido. Las imágenes T2 se basan en el tiempo que tarda un núcleo de hidrógeno en perder su orientación en un campo magnético externo. Los tejidos con tiempos de relajación T2 largos aparecen brillantes en las imágenes ponderadas en T2, mientras que los tejidos con tiempos de relajación T2 cortos aparecen oscuros. Por ejemplo, el líquido tiene un tiempo de relajación T2 largo, por lo que aparece brillante en las imágenes ponderadas en T2. La grasa tiene un tiempo de relajación T2 corto, por lo que aparece oscura en las imágenes ponderadas en T2 (Boutet et al., 2021).

D.FSL.

FSL es una colección completa de herramientas de análisis para datos de imágenes cerebrales de resonancia magnética funcional (fMRI), resonancia magnética (RM) e imágenes por difusión. Se ejecuta en macOS (Intel y M1/M2), Linux y Windows a través del Subsistema de Windows para Linux, y es muy fácil de instalar. La mayoría de las herramientas se pueden ejecutar tanto desde la línea de comandos como como interfaces gráficas de usuario (GUI) ("apuntar y hacer clic") (Jenkinson et al., 2012).

FSL es una biblioteca de software de código abierto para el procesamiento de imágenes de resonancia magnética (RM). Es una herramienta poderosa y versátil que puede ser utilizada para mejorar la calidad y la interpretabilidad de las imágenes de RM (Jenkinson et al., 2012).

FSL ofrece una amplia gama de herramientas y algoritmos para el procesamiento de imágenes de RM. Estas herramientas se pueden utilizar para realizar una variedad de tareas, como (Jenkinson et al., 2012):

- Segmentación de tejidos: FSL se puede utilizar para segmentar imágenes de RM en diferentes tipos de tejidos, como la materia blanca, la materia gris y el líquido cefalorraquídeo. Esto puede ser útil para el diagnóstico, el tratamiento y la investigación de una variedad de condiciones médicas (Jenkinson et al., 2012).
- Corrección de artefactos: FSL se puede utilizar para corregir artefactos de imágenes de RM, como artefactos de movimiento y artefactos de gradiente. Esto puede mejorar la calidad de las imágenes y facilitar su interpretación (Jenkinson et al., 2012).
- Registro de imágenes: FSL se puede utilizar para registrar imágenes de RM de diferentes sujetos o de diferentes tiempos. Esto puede ser útil para comparar imágenes de RM de un mismo sujeto a lo largo del tiempo o de diferentes sujetos (Jenkinson et al., 2012).

E. Nipype.

Los módulos de interfaz de Nipype proporcionan una interfaz Python para paquetes externos como FSL y SPM. Dentro del módulo hay una serie de clases de Python que encapsulan funcionalidades específicas del paquete. Por ejemplo, en el módulo FSL, la clase Nipype. interfaces. FSL. Bet encapsula la herramienta de línea de comandos BET. Al usar la herramienta de línea de comandos, se especificarían las opciones usando banderas como -o, -m, -f <f>, etc. Sin embargo, en Nipype, las opciones se asignan a atributos de Python (Gorgolewski et al., 2011a).

F. ANTs.

ANTs es una biblioteca de software de código abierto para el procesamiento de imágenes de resonancia magnética. Es una herramienta poderosa y versátil utilizada en aplicaciones, incluyendo investigación médica, diagnóstico y programación quirúrgica (Avants et al., 2022).

ANTs proporciona una amplia gama de herramientas para el preprocesamiento, la segmentación, el registro y el análisis de imágenes. Las herramientas de preprocesamiento se utilizan para corregir los artefactos de las imágenes de RM, como el movimiento de la cabeza, la falta de homogeneidad del campo magnético y el ruido. Las herramientas de segmentación se utilizan para identificar las estructuras cerebrales en las imágenes de IRM. Las herramientas de registro se utilizan para imágenes alinéales de IRM de diferentes pacientes o de diferentes tiempos. Las herramientas de análisis se utilizan para cuantificar las características de las imágenes de IRM, como el volumen cerebral y la presencia de lesiones (Gorgolewski et al., 2011a).

ANTs está escrito en C++, pero se puede utilizar desde una variedad de lenguajes de programación, incluidos Python, MATLAB y R. Está disponible para una variedad de plataformas, incluidas Windows, macOS y Linux.(Avants et al., 2022).

G. NNUNET.

El machine learning es un campo de la inteligencia artificial que permite a los ordenadores aprender sin ser programados explícitamente. Los ordenadores aprenden a realizar tareas mediante la construcción de modelos a partir de datos de entrenamiento. Estos modelos se utilizan para hacer predicciones o tomar decisiones sobre nuevos datos (Isensee et al., 2020).

El machine learning y las CNN son dos tecnologías poderosas que tienen el potencial de transformar la forma en que vivimos y trabajamos. A medida que estas tecnologías continúan desarrollándose, se espera que tengan un impacto aún mayor en nuestras vidas (Isensee et al., 2020).

En resumen, el aprendizaje automático y las CNN son dos tecnologías que permiten a los ordenadores aprender y adaptarse. Estas tecnologías se utilizan en una amplia gama de aplicaciones, y tienen el potencial de transformar la forma en que vivimos y trabajamos (Isensee et al., 2020). nnUnet es una red convolucional de aprendizaje profundo diseñada para el segmento de imágenes médicas. El equipo de investigación de la Universidad de Oxford lo desarrolló y se publicó por primera vez en 2018 (Isensee et al., 2020). nnUnet es una red convolucional de tipo U-Net, que consta de una serie de capas convolucionales, de agrupación y de conexión completa. Las capas convolucionales extraen características de las imágenes,

mientras que las capas de agrupación reducen la dimensionalidad de las características. Las capas de conexión completa se usan para la segmentación final (Isensee et al., 2020).

IV. Metodología

A. Sujetos del estudio.

Un equipo de investigación de la Universidad de Stanford ha desarrollado un conjunto de datos de imágenes de resonancia magnética (RM) de pacientes con esclerosis múltiple (EM), con segmentaciones de lesiones de sustancia blanca. El conjunto de datos consta de reconstrucciones de imágenes de RM en el plano axial de secuencias tipo FLAIR, T2, y T1 y segmentaciones de lesiones de esclerosis múltiple y datos demográficos de los pacientes (Hata et al., 2023).

Las imágenes de RM se adquirieron en un escáner de RM 3T con una resolución de 1x1 x1 mm3. Las segmentaciones de lesiones esclerosis múltiple se realizaron manualmente por neuro radiólogos experimentados. Los datos demográficos de los pacientes incluyen la edad, el sexo, la duración de la enfermedad, el tipo de EM y el tratamiento actual (Hata et al., 2023).

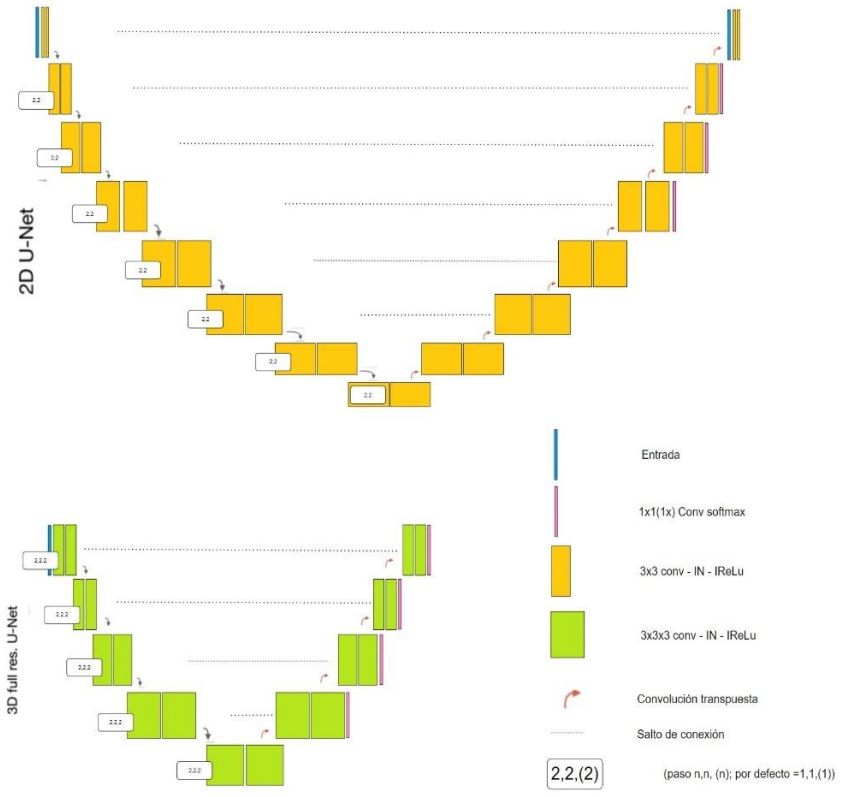
El conjunto de datos se puede utilizar para una amplia gama de aplicaciones de investigación, incluyendo el desarrollo de algoritmos de detección y segmentación de lesiones de EM, el estudio de la progresión de la enfermedad y el desarrollo de nuevos tratamientos para el diagnóstico de esta (Hata et al., 2023).

Los datos tienen 50 pacientes diagnosticados con EM, de los que la mayoría son mujeres coinciden con la tasa de incidencia mayor en mujeres que en hombres. La edad media de los pacientes es de 37 años, que es la edad a la que la EM suele diagnosticarse. El tipo de EM más común es el RR. El intervalo de tiempo más común entre estudios es de 200 a 300 días, que es un intervalo de tiempo común para los estudios de seguimiento de la EM. Los datos están disponibles en https://github.com/muschellij2/open_ms_data (Hata et al., 2023). De los cuales se escogieron los mejores 2060 cortes en el plano axial de 29 pacientes con el objetivo de reducir el costo computacional a la hora de realizar el entrenamiento (McKinley et al., 2020).

B. Redes de entrenamiento.

Dado que una primera instancia se tenían reconstrucciones del plano axial en formato NIFTI se optó por trabajar con la configuración nnUnet denominada como Fullres, esta red convolucional se ejecuta con datos de resolución completa. El tamaño del parche está limitado por disponibilidad de memoria GPU. Sin embargo, para datos grandes, el tamaño del parche puede ser demasiado pequeño para agregar suficiente información contextual, por lo que se inclinó a usar la configuración 2D del modelo convolucional nnUnet que se ejecuta con datos de resolución completa con datos anisotrópicos, en la Figura 1, se puede observar ejemplos de arquitectura para las dos redes, ya que el algoritmo en el preprocesamiento recodifica la red con los parámetros de entrada disponibles en la Tabla I y que están disponibles en el archivo "nnUNetPlans.json" (Isensee et al., 2020).

Figura 1. Configuración de redes neuronales



Nota. Fuente (Isensee et al., 2020).

Tabla I. Parámetros de las redes convolucionales.

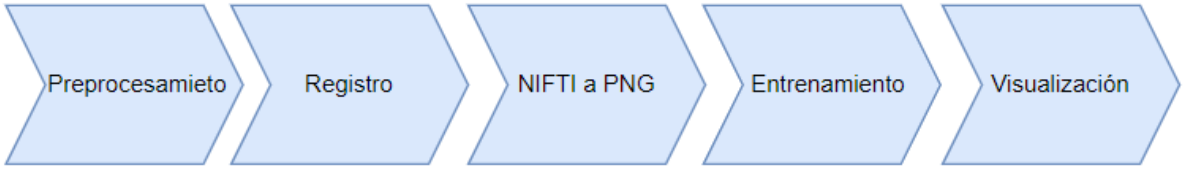
Dataset	
Median Image Size	167 x 217 x 178
Median Image Spacing	1 x 1x 1
Normalization	ZScoreNormalization
2D- nnUnet	
Targen Spacing	Na x 1x 1
Median Shape @ Target Spacing	Na x 217x 178
Patch Size	224x192
BatchSize	76
3D- nnUnet	
Targen Spacing	1 x 1 x 1
Median Shape @ Target Spacing	167 x 217 x 178
Patch Size	112 x 160 x 128
BatchSize	2

Nota. Adaptado de Isensee et al., 2020

C. Flujo de procesamiento.

El flujo de procesamiento consta de 5 fases, la primera fase de preprocesado donde se aplica una extracción de cerebro haciendo uso de FSL(Jenkinson et al., 2012) y una corrección de la intensidad de las imágenes (Sled et al., 1998), en la fase dos se construye un mapa de probabilidad(Bodini et al., 2011a; Dworkin et al., 2018) haciendo uso de FSL para llevar a cabo una serie de registros de secuencias Flair, T2 y T1, en la fase 3 se realiza una conversión de archivos NIFTI a png por medio de las librerías Nibabel y open CV (McAtee, Ia). En la fase 4 se realiza el entrenamiento por parte de la nnUnet y finalmente, En la fase 5 se superponen las imágenes de etiqueta haciendo uso de la librería Pillow y Nibabel (Roscheck, Florián) (Figura 2).

Figura 2. Flujo de procesamiento.



1.Flujo de preprocesamiento.

1.1. Extracción de cerebro con FSL y Nipype.

La extracción de cerebro es un proceso de preprocesamiento de imágenes de resonancia magnética (IRM) que se utiliza para eliminar el tejido no cerebral de las imágenes. FSL proporciona una herramienta llamada BET (Brain Extraction Tool) para realizar la extracción de cerebro. BET funciona mediante la identificación de los bordes del tejido cerebral en las imágenes de IRM y luego eliminando el tejido no cerebral de las imágenes (Jenkinson et al., 2012).

1.2. N4 Bias Correction con ANTs y Nipype.

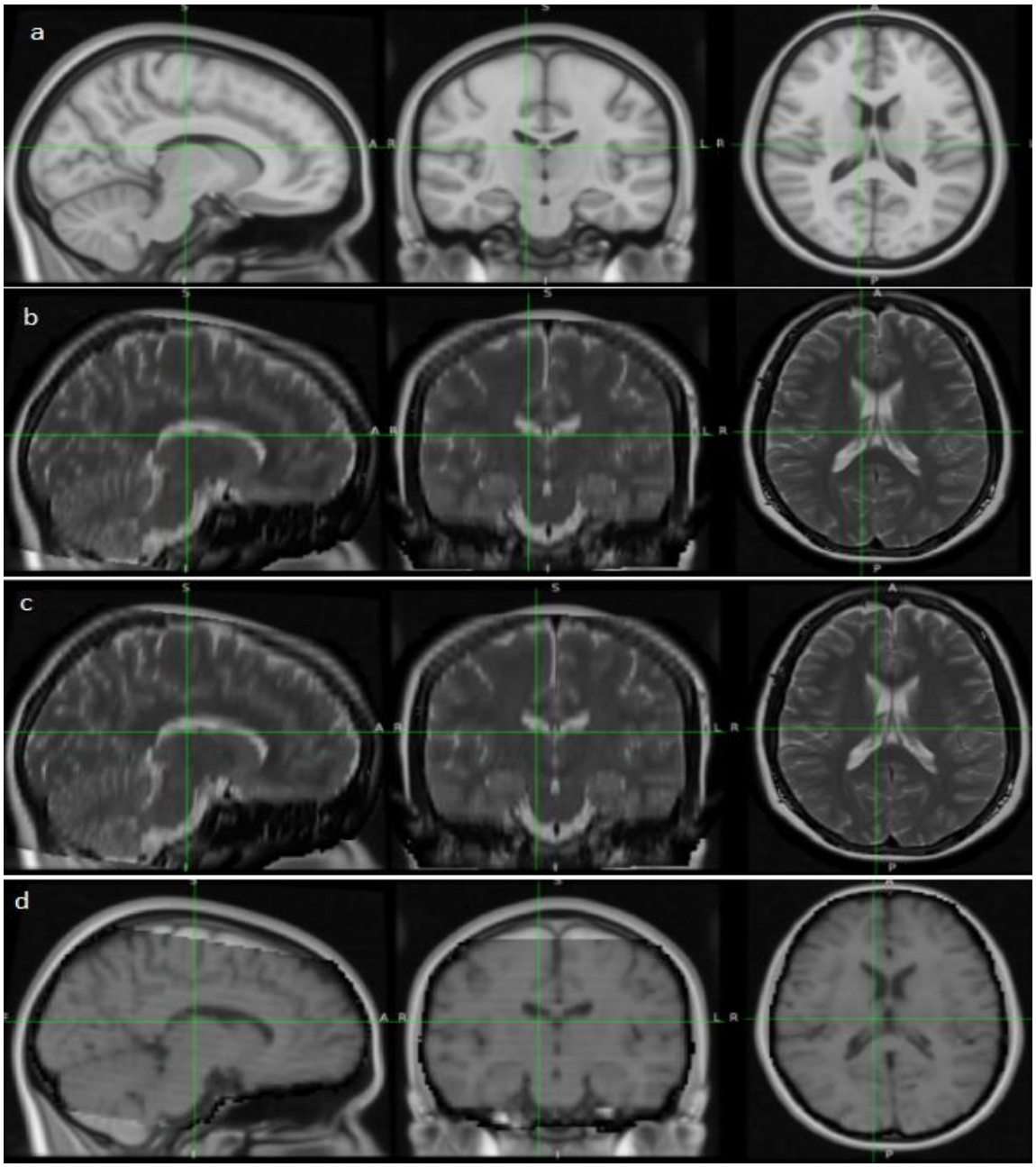
La corrección N4Bias es un método de preprocesamiento de imágenes de RM que se utiliza para corregir los sesgos causados por la heterogeneidad del campo magnético, la intensidad del pulso de radiofrecuencia y la posición del paciente de manera que mejora la uniformidad del contraste y la nitidez de la imagen. La falta de homogeneidad del campo de intensidad es un artefacto común de las imágenes de RM que puede causar variaciones en la intensidad de la señal en toda la imagen. Esto puede dificultar la visualización y la segmentación de las estructuras del cerebro (Gorgolewski et al., 2011b; Sled et al., 1998).

La corrección N4Bias funciona mediante la estimación y eliminación de un campo de sesgo de las imágenes de RM. El campo de sesgo es un mapa que representa la inhomogeneidad del campo de intensidad de la imagen. Cuando se estima el campo de sesgo, se puede eliminar de la imagen con varios métodos (Gorgolewski et al., 2011b).

2. Registro.

Se registraron inicialmente las secuencias T1 de los pacientes en el espacio estándar, utilizando las plantillas MNI152_T1_1mm (Figura 3.a) de FSL y haciendo uso de los comandos FLIRT, FNIRT Y APPLYWARP (Figura 3.c) (Jenkinson et al.). Para obtener el mapa de probabilidad (Bodini et al., 2011a; Murphy & Gaillard, 2015)de los vóxeles se utilizaron las transformaciones del registro FLAIR sobre el registro T1 para registrar las secuencias T2 en las secuencias T1 con la herramienta FLIRT (Figura 3 .b) (Bodini et al., 2011a; Jenkinson et al.), finalmente los registros de T2 sobre T1 usando las transformaciones de las secuencias FLAIR registradas en las secuencias T1 estandarizadas en la plantilla MNI152_T1_1mm haciendo uso del comando APPLYWARP (Figura 3.c) (Bodini et al., 2011a; Jenkinson et al.). En la Figura 5 y Figura 6 se describe el paso a paso del flujo de preprocesamiento.

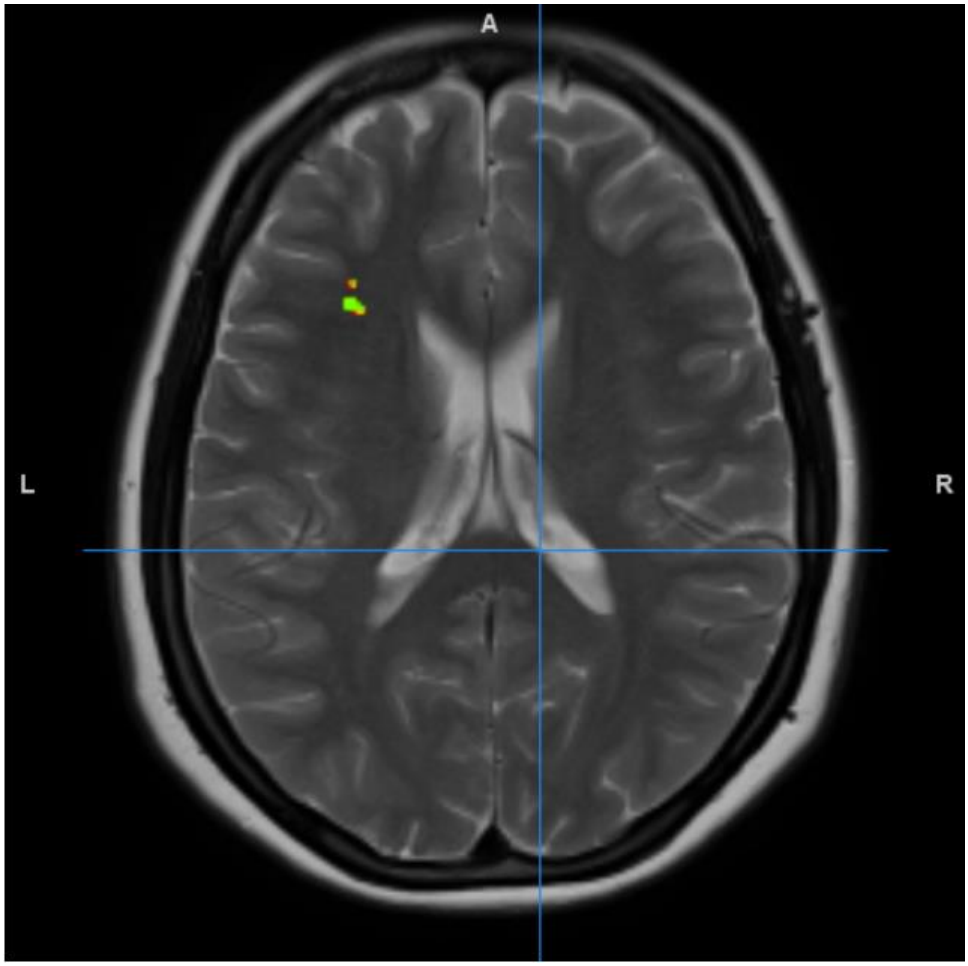
Figura 3. Registro de la secuencia T2 sobre la secuencia T1 registrada en el espacio estándar MNI152. a) Plantilla MNI152_T1_1mm; b) T2toMNI, T1toMNI, superpuestos; c) T1toMNI, Plantilla MNI152_T1_1mm, superpuestos; d) T2toMNI, Plantilla MNI152_T1_1mm, superpuesto..



Nota. Fuente (Jenkinson et al.).

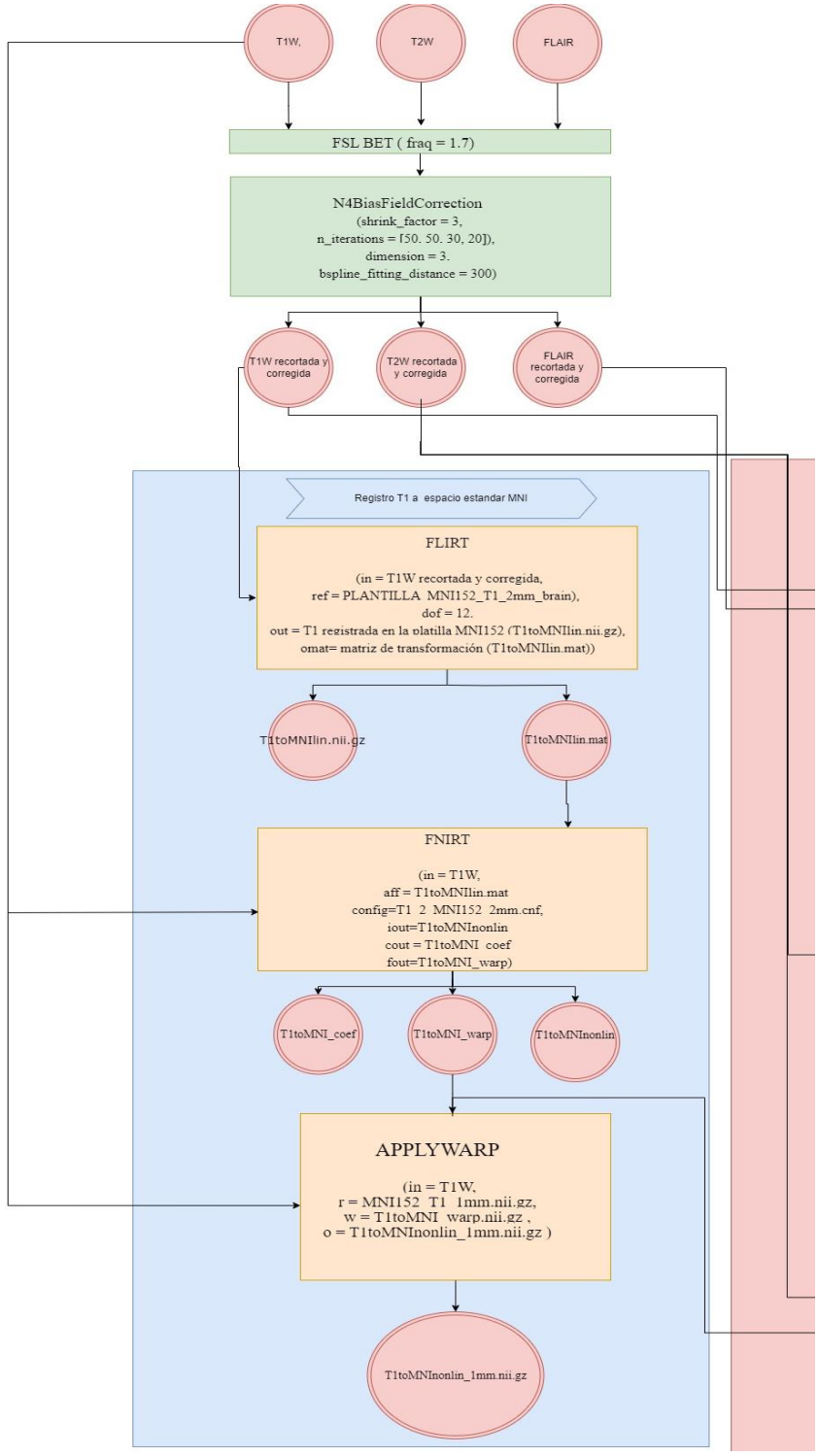
Después, se registró (Figura 7) las máscaras de las lesiones al espacio estándar con FLIRT y se normalizaron usando las herramientas MATH de FSL (Figura 4). (Jenkinson et al.).

Figura 4. T2toMNI, Lesión sin registrar, Lesión registrada al espacio estándar superpuestos.



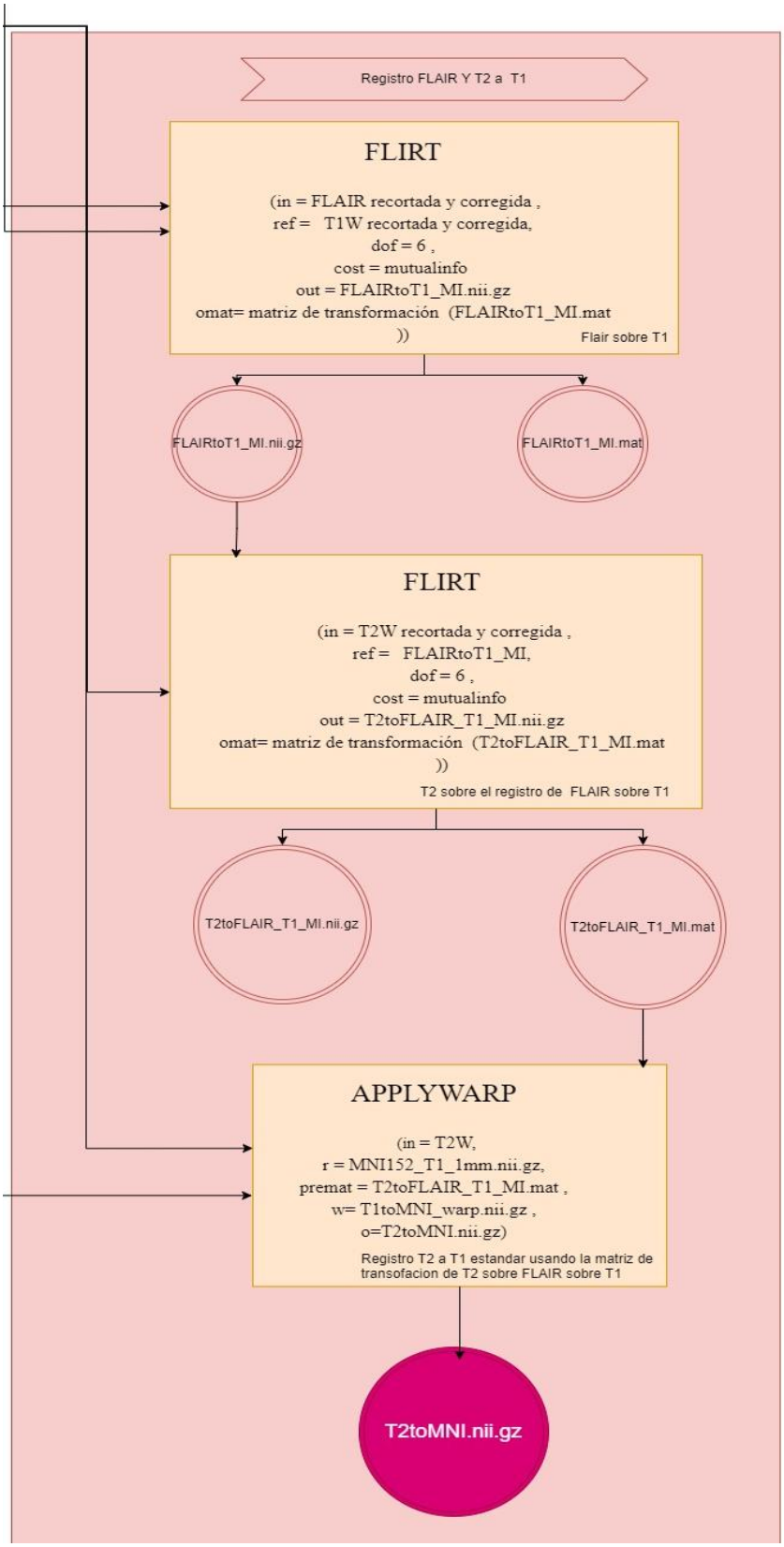
Nota. Fuente (Jenkinson et al).

Figura 5. Flujo. de preprocesamiento (BET y N4Bias) y Flujo de registro T1 a plantilla estándar MNI152.



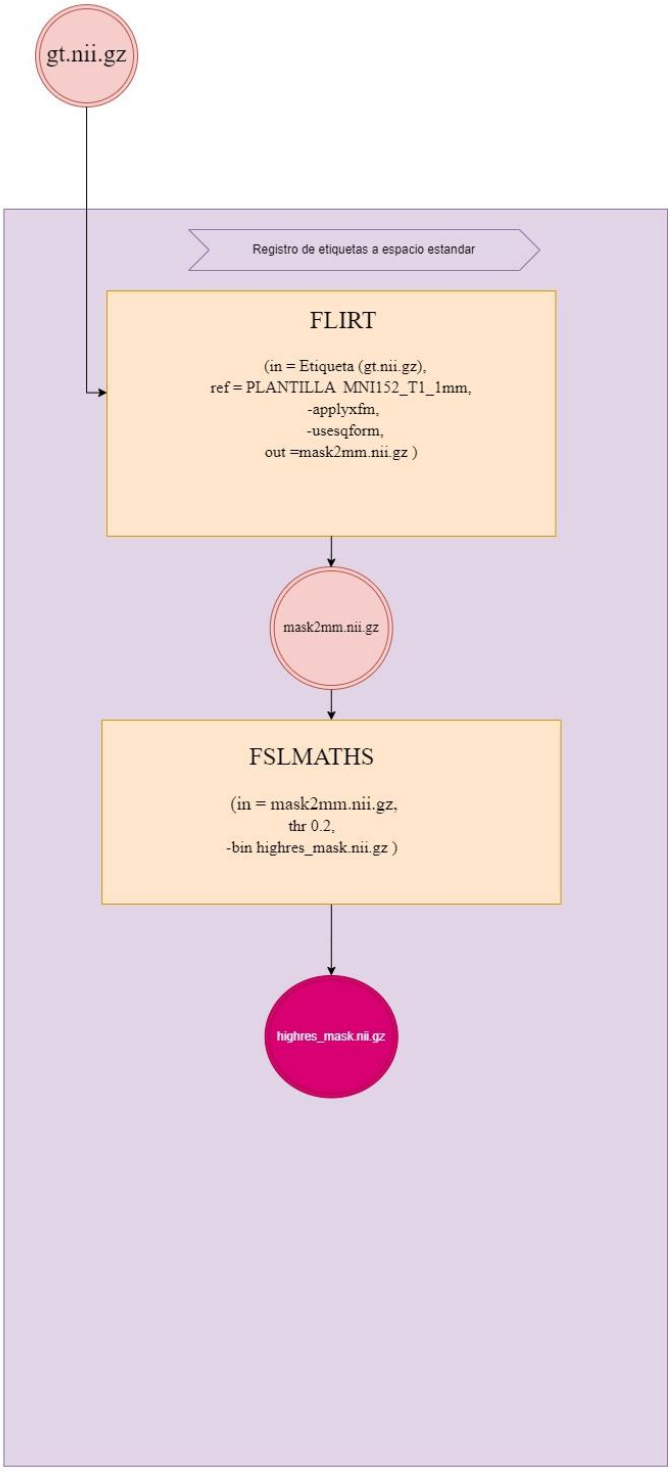
Nota. Fuente (Bodini et al., 2011b; Jenkinson et al., 2012).

Figura 6. Flujo de registro Flair y T2 a T1 registrado en la plantilla estándar MNI152.



Nota. Fuente (Bodini et al., 2011b; Jenkinson et al., 2012).

Figura 7. Flujo de registro Flair y T2 a T1 registrado en la plantilla estándar MNI152.



Nota. Fuente (Jenkinson et al., 2012).

3. Conversión NIFTI a Cortes en PNG.

La conversión de datos de NIFTI a PNG se realizó por medio de Python, haciendo uso de las librerías Nibabel para realizar la lectura de los datos NIFTI, una vez se obtiene la matriz de los archivos NIFTI por medio del comando get_fdata se obtienen 218 cortes para el plano coronal y 180 cortes para los planos sagital y axial. Para trabajar de una forma anatómicamente más precisa se usaron los cortes con los cuales se realizó la reconstrucción del volumétrico ya que son las de mayor calidad (cortes axiales), se ajustar la relación de aspecto de las imágenes por medio de la librería open CV (cv2). Con el comando cv2.resize, utilizando otra vez de la librería open CV, se rota el corte con el comando cv2.rotate, se ajusta la luminosidad (alfa) y el contraste (beta) por medio del comando cv2.convertScaleAbs para finalmente organizar las dimensiones a escala de grises con la función cv2.cvtColor y usando el parámetro cv2.COLOR_GRAY2BGR guardando con el comando cv2.imwrite que recibe la matriz y la ruta de salida (McAtee Ian,).

4. Entrenamiento.

Tanto para la etapa de entrenamiento como de validación y predicción de la nnUnet se utilizó la versión gratuita de la herramienta T4 GPU de Google Colab, por lo que primero se realizó la instalación de la CNN nnUnet por medio del comando “pip” de Python. Una vez se instala, se modifica el archivo nnUNetTrainer.py para que guarde todas las épocas, ya que inicialmente se programa para guardar cada 50 épocas, y se modifica el archivo nnUNetTrainer_Xepochs.py para determinar el número de épocas hasta el cual se desea entrenar el modelo (Isensee et al., 2020).

Posteriormente, se crean las carpetas que serán definidas como las variables de ambiente (“nnUnet_raw”, “nnUnet_preprocessed”, “nnUNet_results”). Los registros T2 sobre T1 estandarizado entrenarán el algoritmo de aprendizaje, por lo que deben estar localizados en la carpeta “nnUnet_Raw”. Esta carpeta, contiene los ficheros “imagesTs” que contiene los registros T2toMNI para la validación, el fichero “imagesTr” que contiene las imágenes para el entrenamiento y las etiquetas de las lesiones estarán contenidas en el fichero “labelsTr”. Además, las imágenes de entrenamiento y validación fueron renombradas con el formato “DATASET_numeroImage_numeroMascara”, por ejemplo, para el primer dato se renombraría de la siguiente manera, “EM_0000_0000”. Mientras que para las etiquetas se tiene

"DATASET_numeroImagen", por ejemplo, la etiqueta de lesión para el dato anterior sería "EM_0000" (Anexo A) (McAtee Ian).

Una vez se tienen las imágenes renombradas y localizadas en los respectivos ficheros, se ejecuta el comando, se construye el dataset.json que es el archivo de configuraciones iniciales para el algoritmo de machine learning. Posteriormente, se ejecuta el comando "! nnUNetv2_plan_and_preprocess -d 2 -c 2d --verify_dataset_integrity" para verificar y preprocesar los datos de forma que se especifica el dataset con el cual se desea entrenar (-d 2) y la configuración a usar (-c 2d). Con este comando se generan los archivos de preprocesamiento que serán guardados en la carpeta previamente definida como nnUnet_preprocessed, que contiene las configuraciones iniciales que utilizará el algoritmo para realizar el entrenamiento. Finalmente, se ejecuta el " !nnUNetv2_train 2 2d 0 -tr nnUNetTrainer_300epochs" de manera que se especifica el dataset "2", la configuración "2d", la carpeta donde se guardarán los resultados "0" (nnUnet_results/Dataset002_EM/config_epochs/fold0) y las épocas "nnUNetTrainer_300epochs" - (Isensee et al., 2020).

Para la configuración inicial se escribe un archivo json el cual será nombrado como "dataset.json" en este archivo se escribe el nombre del dataset, la modalidad de las imágenes, los canales, el tipo de archivo, las etiquetas que se esperan para las máscaras en el entrenamiento y las rutas donde se encuentran los datos de entrenamiento. Además, se umbralizan las imágenes de manera que se tienen valores de intensidad igual a 1 para píxeles con intensidad mayores a 180, y una intensidad de 0 para píxeles con una intensidad menor a 180, esto se realiza, puesto que en el archivo dataset.json se definen valores de 1 para lesión y 0 para no lesión(Isensee et al., 2020).

5. Visualización de lesiones.

Para la visualización de las lesiones, se hace uso de la librería Pillow de Python, en primer lugar, se abren los cortes en los cuales se detectan las lesiones y las lesiones predichas y las etiquetas de las lesiones en formato PNG. Una vez se cargan se utiliza el comando convert("RGBA") utilizada para convertir el modo de una imagen a RGBA, este modo hace referencia a los cuatro canales: rojo, verde, azul y alfa. El canal alfa se utiliza para almacenar la transparencia de cada píxel. Posteriormente se extrae el vector del corte ya leído, en el caso de las lesiones, se modifica el color tanto para la lesión predicha como para la etiqueta, con la finalidad de distinguirlas en el plano. Para combinar el fondo (el corte de cerebro en el plano axial) se utiliza el comando Image.blend que recibe como argumentos el fondo (corte en el plano axial), el overlay (lesiones predichas y etiquetas de lesiones) y la transparencia del overlay para finalmente guardar las imágenes con el comando Image.save que recibe como argumentos la ruta de salida de las imágenes y el tipo de archivo que en este caso será nuevamente PNG (Roscheck, Florian).

D. Cambio de hiperparámetros.

Algunos hiperparámetros para el modelo convolucional nnUnet están definidos como variables en el archivo nnUNetTrainer.py, los hiperparámetros que fueron modificados en el proyecto están definidos en este archivo como variables. En primer lugar, se tiene el initial_lr que hace referencia al learning rate que controla la velocidad con la que el modelo aprende a predecir, otro parámetro modificado fue weight_decay que hace referencia al decaimiento de pesos, una técnica de regularización que ayuda a evitar el sobreajuste. El oversample_foreground_percent es un parámetro que se utiliza para aumentar la proporción de píxeles de primer plano en el conjunto de datos de entrenamiento con el objetivo de mejorar la precisión del modelo en la detención de objetos de primer plano, finalmente se modificó el parámetro de momentum, el cual también es usado para regular la velocidad de aprendizaje del modelo y se define como la proporción de la actualización de los pesos del modelo que se basa en la actualización anterior. Como ya se mencionó anteriormente, el hiperparámetro que hace referencia al número de épocas se define en el archivo nnUNetTrainer_Xepochs, por último, se tiene el batchsize, este parámetro define el número de muestras que se utilizan para actualizar los pesos del modelo en cada entrenamiento que se modifica en el archivo nnUNetPlans.json,

este archivo se genera en la carpeta nnUnet_preprocessed una vez se realiza la verificación de los datos (Arai & Lyubchich, 2022; Isensee et al., 2020).

1. Configuración 3D Fullres.

Inicialmente, se entrenó el algoritmo con 29 volúmenes de los 50 pacientes, utilizando la configuración 3Dfullres con 50 épocas, un learning rate de 0.01, un momento de 0.99, utilizando la función de perdida BCE, un batchsize de 2, un decaimiento de pesos de 0.00001 y un oversample_foreground_percent de 0.33 (Arai & Lyubchich, 2022).

2. Configuración 2D.

Se clasificaron los mejores 2060 cortes en el plano axial y con lesión de los 29 pacientes y se utilizó la configuración 2d de la nnUnet. Con un número de épocas de 300. Además, para agilizar el proceso de entramiento y evaluar diferentes hiperparámetros(Tabla II), se trabajaron solo con 50 cortes de manera que se pueda determinar el mejor modelo.

Tabla II. Parámetros de entrenamiento para el modelo nnUnet con configuración 2D.

Número de Entrenamiento	**Learnig rate**	**weight_decay**	**oversample_foreground_percent**	**momentum**	**batchsize**
1	1e-2	3e-5	0.33	0.99	2
2	1e-3	3e-5	0.33	0.99	2
3	1e-3	3e-6	0.43	0.98	2
4	1e-2	3e-5	0.43	0.97	2
5	1e-2	3e-6	0.43	0.98	2

V. Resultados

A. Comparación entre diferentes CNNs.

Las arquitecturas de CNN son una herramienta prometedora para la segmentación automática de lesiones de esclerosis múltiple. Las arquitecturas más recientes, como TransBTS, TransUNet y nnU-Net, parecen ofrecer los mejores resultados en términos de precisión, sensibilidad y especificidad (Basaran et al., 2022).

En general, las diferencias entre estas arquitecturas se encuentran en la forma en que aprenden las relaciones entre los píxeles de la imagen. UNETR utiliza capas de transformación, Tiramisu 2. 5D utiliza capas de expansión, TransBTS utiliza capas de transformadores, TransUNet utiliza una combinación de capas convolucionales y de transformadores, y nnU-Net utiliza una combinación de capas convolucionales y de DNN (Basaran et al., 2022) .

En cuanto a la mejor arquitectura para segmentar lesiones de esclerosis múltiple, no hay una respuesta definitiva. Los resultados de los estudios publicados varían en función del conjunto de datos utilizados y de los parámetros de la red. Sin embargo, en general, las arquitecturas más recientes, como TransBTS, TransUNet y nnU-Net, parecen ofrecer mejores resultados que las arquitecturas más antiguas, como U-Net y Tiramisu 2.5D (Basaran et al., 2022).

En la Tabla III se muestra la comparación de distintas métricas para los diferentes modelos, utilizando el conjunto de datos de desafío de segmentación de lesiones nuevas de EM MICCAI 2021. De la cual se puede observar que el modelo nnUnet es la que presenta el coeficiente de Dice (DSC) más grande (0.490) por lo que se decide trabajar con este modelo (Basaran et al., 2022).

Tabla III. Comparación de diferentes CNNs

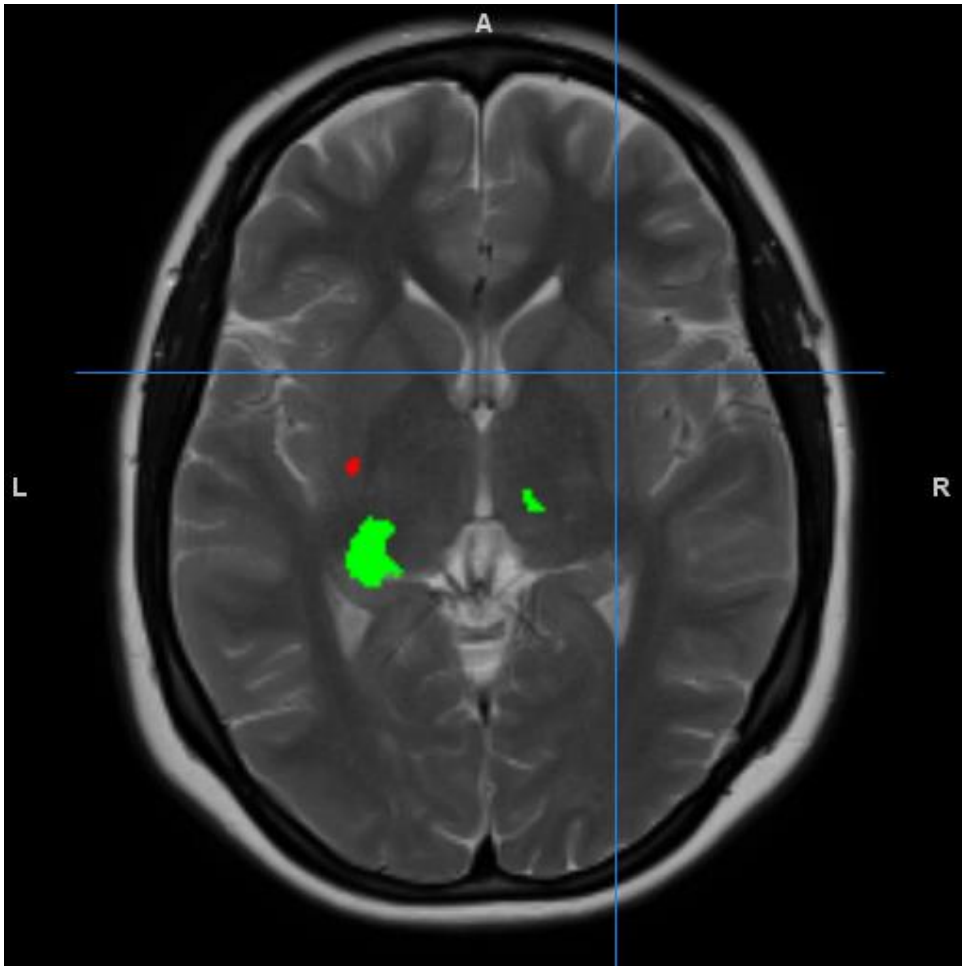
	Nuevos casos de lesiones (*n* = 32)				**No hay nuevos casos de lesiones (n = 28)**		
Método	**DSC**	**F_1**	**n_{TP}**	**n_{FP}**	**n_{FN}**	**n_{FP}**	**V_{FP}**
nnU-Net	0.490	0.548	4.562	1.281	2.688	0.036	0.138
TransUNet	0.428	0.434	4.491	4.043	2.102	1.081	9.620
TransBTS	0.477	0.470	5.492	5.718	1.848	0.939	12.238
Tiramisu 2.5D	0.363	0.365	4.313	4.625	2.938	1.384	15.120
UNETR	0.462	0.468	5.343	9.031	1.906	4.214	23.705

B. Comparación entre las configuraciones Fullres y 2D de nnUnet.

1. Configuración 3D Fullres.

Inicialmente se puede observar que el rendimiento del modelo con la configuración 3DFullres (Figura 8) no fue el mejor, puesto que las lesiones se encuentran en lugares anatómicamente diferentes, aunque se probaron diferentes valores de learning rate, momento y decaimiento de pesos, los mejores resultados se obtuvieron con los hiperparámetros mencionados en la Tabla II.

Figura 8. Predicción de la lesión (Rojo) vs etiqueta de la lesión (verde) en el entrenamiento 3D en el sujeto 3.



En la Tabla IV se pueden visualizar los parámetros de evaluación para el modelo 3D Fullres, de la cual se puede observar que el coeficiente Dice del entrenamiento (Figura 9) fue mayor que para el de la validación en el cual se puede observar un valor de 0.023 y un IoU de 0.01. Aunque se tiene una gran cantidad de TP, se puede observar que se tienen un valor muy bajo de TP (367) en comparación con el valor de los FP (5864). Ya que no fue posible mejorar

el rendimiento de la red, por lo que se cambiaron los formatos de los registros T2toMNI de nii.gz a .png.

Figura 9. Entrenamiento para volúmenes con 3DFullres.

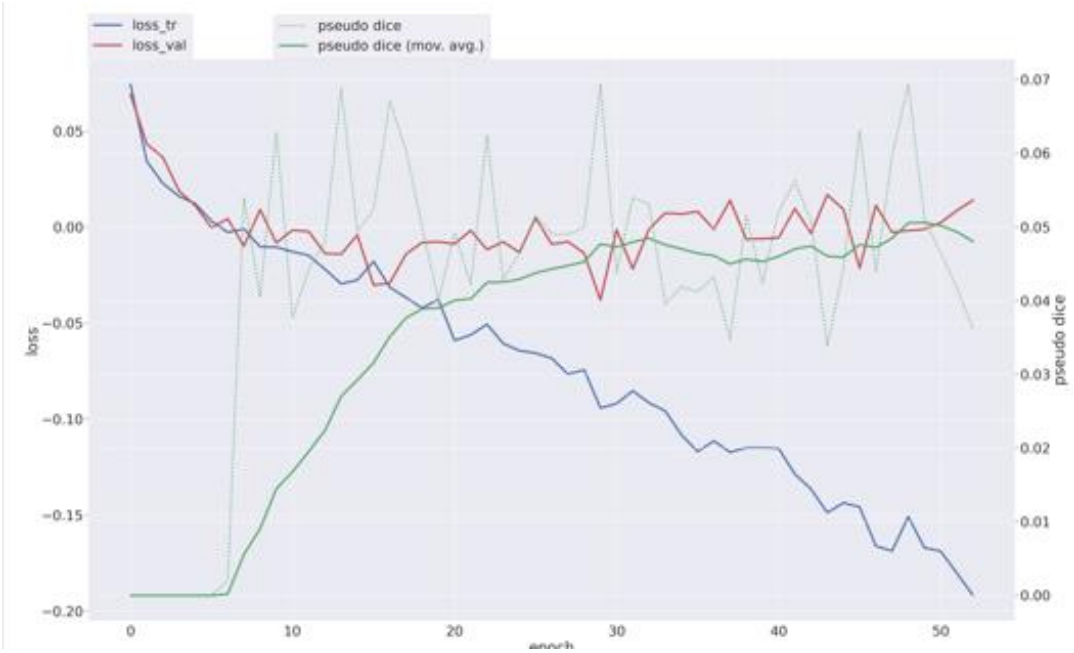


Tabla IV. Resultados del entrenamiento 1 para el modelo nnUnet con configuración 3D Fullres.

Dice	0.023
FN	12934
FP	5864
IoU	0.01
TN	7201866
TP	367

2. Configuración 2D.

Con una selección visual favorable a los mejores cortes, fue posible mejorar el rendimiento de la red convolucional usando la configuración 2D y usando los cortes en el plano axial (Figura 10). En cuanto el rendimiento en la etapa de entrenamiento, se alcanzó un valor para el coeficiente de Dice alrededor de 0.84 (Figura 11) utilizando los mismo hiperparámetros que en el entramiento para la configuración Fullres. Sin embargo, para la gráfica de perdida entrenamiento “loss_tr” y perdida de validación “loss_val” se tienen valores muy inestables por lo en entrenamientos posteriores se realizaron cambios en los hiperparámetros que permitieran disminuir la inestabilidad en estas dos gráficas.

Figura 10. Predicción de la lesión (Rojo) vs etiqueta de la lesión (verde) en el entrenamiento 2D. 1 en el corte 020.

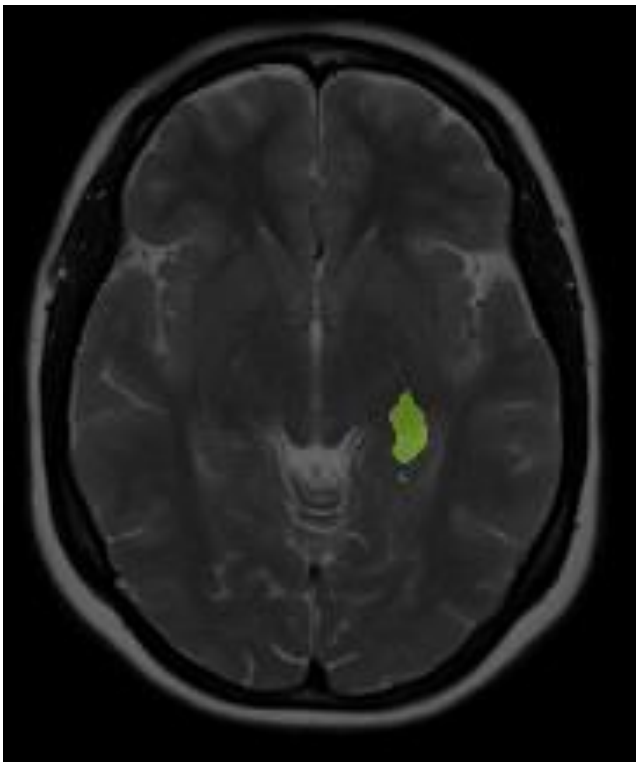
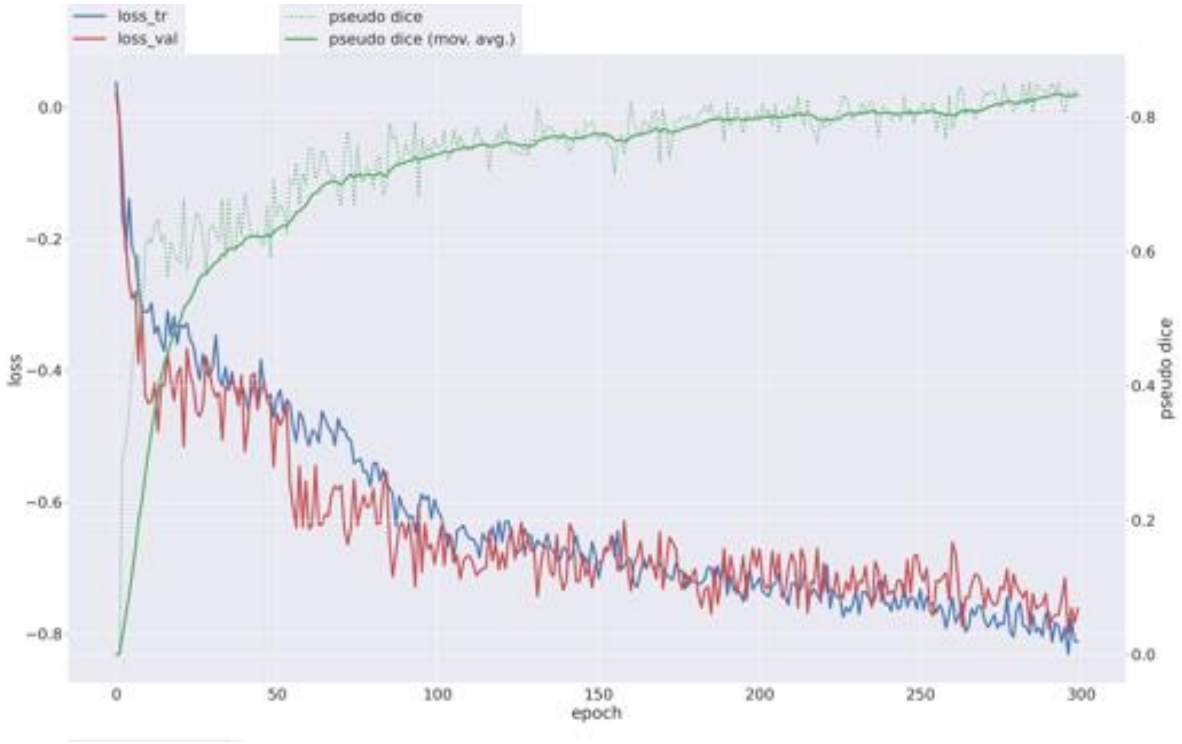


Figura 11. Entrenamiento 1 para cortes con nnUnet 2D.



El coeficiente de Dice para la validación fue igual de alto que para el entrenamiento con un valor de 0.84 y un valor de IoU de 0.74 (Tabla V), a diferencia de los volúmenes con los que se entrenó la configuración Fullres, por lo que se tienen menos píxeles a los cuales aplicar las métricas de evaluación que se presentan en la Tabla IV.

Tabla V. Resultados del entrenamiento 1 para el modelo nnUnet con configuración 2D.

Dice	0.84
FN	63.3
FP	39
IoU	0.74
TN	118605
TP	319

C. Comparación entre entrenamientos con diferentes hiperparámetro.

1. Entrenamiento 2D 2.

En primer lugar, se realizó el cambio del learning rate, de 0.01 a 0.001, esto con la finalidad de mejorar el coeficiente de Dice, sin embargo, se obtuvo un valor de 0.65 (Figura 13) lo cual no fue muy útil para predecir corte con lesiones muy grandes (Figura 12). Además, se puede observar una gran diferencia entre los valores de la gráfica loss_tr y loss_val, lo que puede dar la idea de que el modelo puede estar sobre ajustado y aprende las características específicas del conjunto de datos.

Figura 12. Predicción de la lesión (Rojo) vs etiqueta de la lesión (verde) en el entrenamiento 2D 2 en el corte 035.

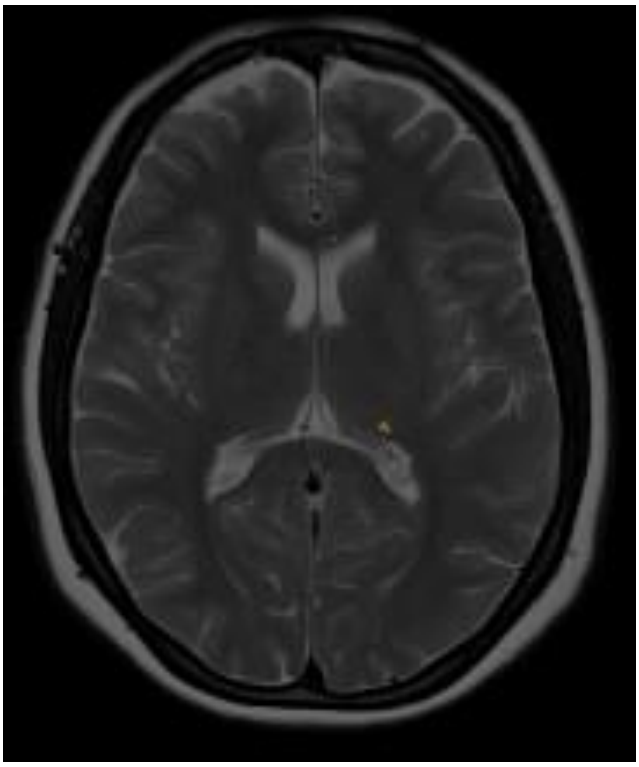
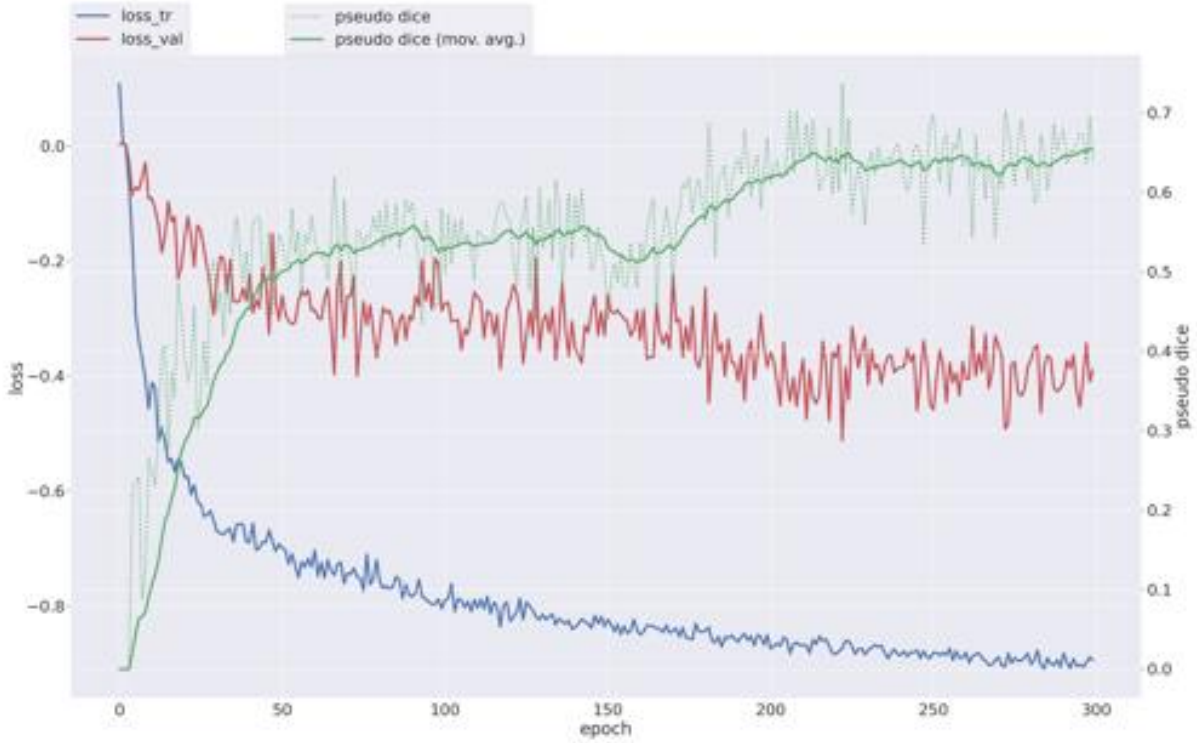


Figura 13. Entrenamiento 2 para cortes con nnUnet 2D.



En la Tabla VI es evidente que a diferencia del entrenamiento 1, para el entrenamiento 2 se tiene un coeficiente de Dice para validación de 0.28 y IoU del 0.22, y aunque se tenga un valor menor de FN (17.1) Y FP (9.3) que el entrenamiento 1, se tiene un valor menos para los TP (24.9) en el entramiento 2 que en el entramiento 1.

Tabla VI. Resultados del entrenamiento 2 para el modelo nnUnet con configuración 2D.

Dice	0.28
FN	17.1
FP	9.3
IoU	0.22
TN	118976
TP	24.9

2.Entrenamiento 2D 3.

Con el propósito de mejorar el sobreajuste, No se modifica el learning rate de 0.001 pero se disminuye el momento (0.98) y se disminuye el decaimiento de pesos (0.0003) además de que se aumenta el oversample_foreground_percent a 0.43, se puede observar en la Figura 15 las gráficas loss_tr y loss_val toman valores cercanos además de que sus valores oscilan menos, mejorando así el sobreajuste. No obstante, se obtuvo un coeficiente de Dice de alrededor 0.84 para el entrenamiento observando buenos resultados para imágenes con lesiones grandes (Figura 14) por lo que se sigue optando por los resultados del entrenamiento 1.

Figura 14. Predicción de la lesión (Rojo) vs etiqueta de la lesión (verde) en el entrenamiento 2D 3 en el corte 028.

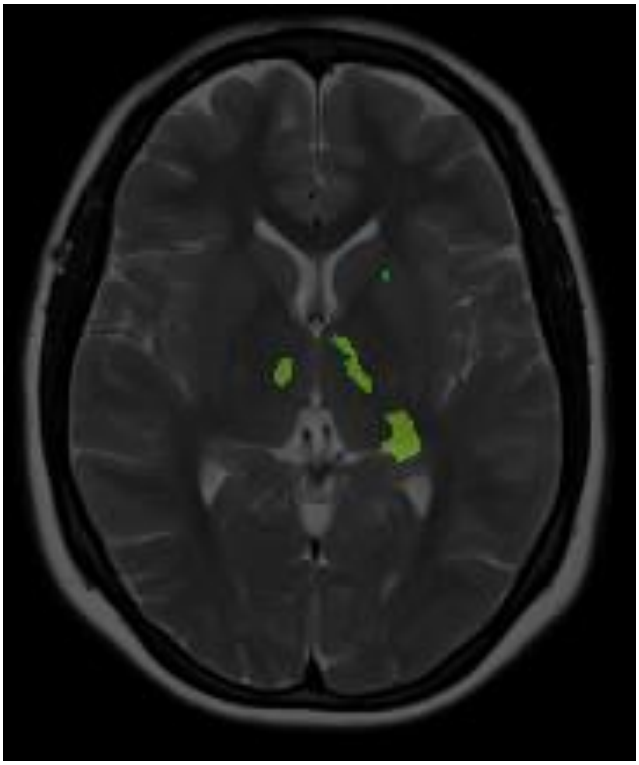
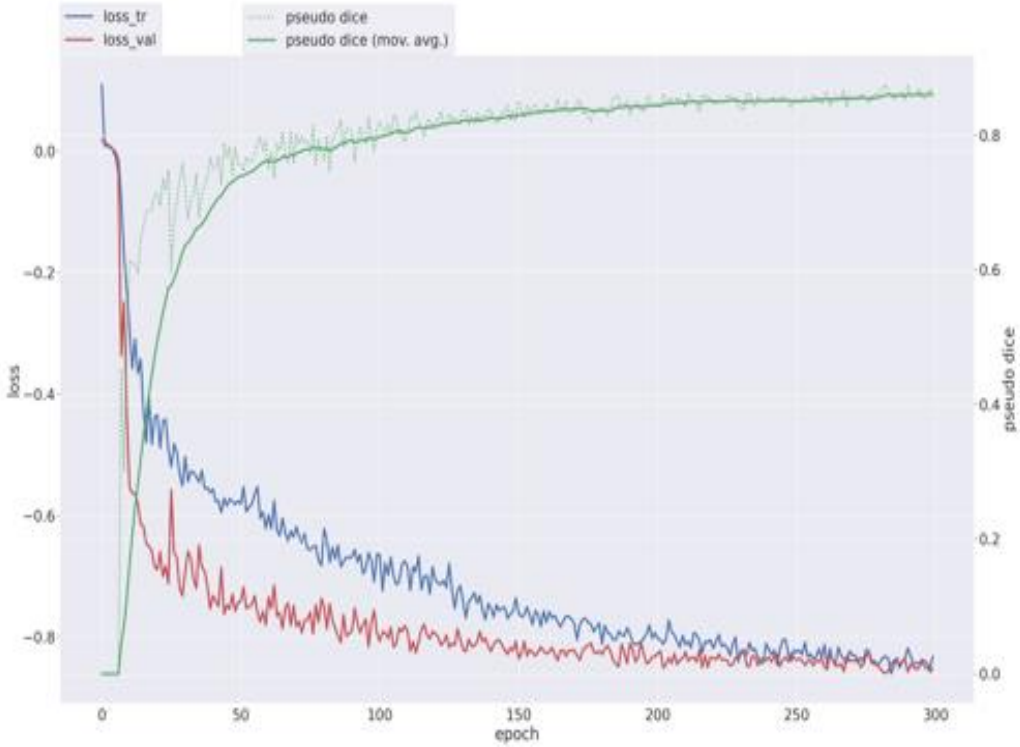


Figura 15. Entrenamiento 3 para cortes con nnUnet 2D.



Aunque de la Tabla VII se puede observar se obtuvieron valores mayores de FN (55.5) Y FP (24.9). Se obtuvo un coeficiente de Dice de (0.67) a diferencia del entrenamiento 2, a pesar de ello, las mejores métricas de validación siguen siendo para el entrenamiento 1.

Tabla VII. Resultados del entrenamiento 3 para el modelo nnUnet con configuración 2D.

Dice	0.67
FN	55.5
FP	24.9
IoU	0.56
TN	118747
TP	200

3. Entrenamiento 2D 4.

Dado que los mejores resultados fueron con un learning rate de 0.01, se aumenta el oversample_foreground_percent y se disminuye el momento a 0.97 con la finalidad de mejorar el sobreajuste y la inestabilidad entre las gráficas de loss_tr y loss_val, por otro lado, se deja el decaimiento de pesos en el valor estándar 3e-5. Obteniendo como resultado un Coeficiente de Dice de alrededor 0.8 para el entrenamiento (Figura 17), Aún asi, en esta misma figura se pudo observar un mayor sobreajuste en los datos dado que los valores loss_tr y loss_val toman valores diferentes y presentan mayor inestabilidad de que el entrenamiento 3 (Figura 16).

Figura 16. Predicción de la lesión (Rojo) vs etiqueta de la lesión (verde) en el entrenamiento 2D 4 en el corte 023.

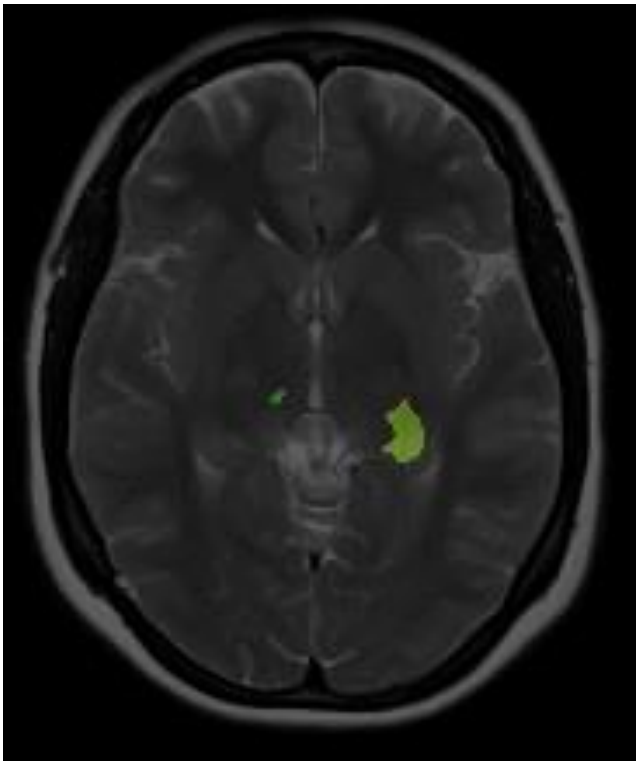
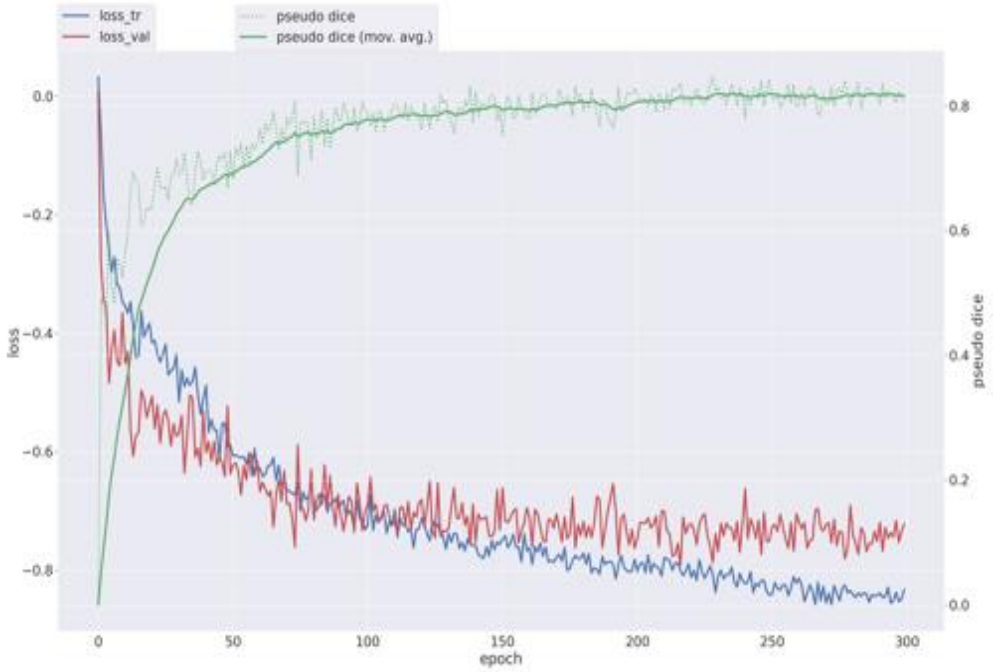


Figura 17.Entrenamiento 4 para cortes con nnUnet 2D.



Aunque se obtuvo un mayor sobreajuste de los datos, se puede observar que el coeficiente de Dice (0.69) y el IoU (0.57) (Tabla VIII) aumentaron en relación con el entrenamiento 3, sin embargo, se siguen siendo mejores los resultados del entrenamiento 1

Tabla VIII. Resultados del entrenamiento 4 para el modelo nnUnet con configuración 2D.

Dice	0.69
FN	60.6
FP	30.9
IoU	0.57
TN	118719
TP	216

4. Entrenamiento 2D 5.

Finalmente, ya que los mejores resultados fueron obtenidos con un learning rate inicial de 0.01 para el entrenamiento 3, se toma hiperparámetro de dicho entrenamiento (momento igual a 0.98, oversample_foreground_percent igual a 0.43 y un decaimiento de pesos de 3e-6), del cual se puede observar predicciones muy parecida al del entrenamiento 4 (Figura 18), Sin embargo, se obtiene un Coeficiente de Dice de alrededor 0.65 (Figura 19), Además de un sobreajuste ya que se puede observar que las gráficas de loss_val y loss_tr convergen en valores diferentes.

.

Figura 18. Predicción de la lesión (Rojo) vs etiqueta de la lesión (verde) en el entrenamiento 2D 5 en el corte 024

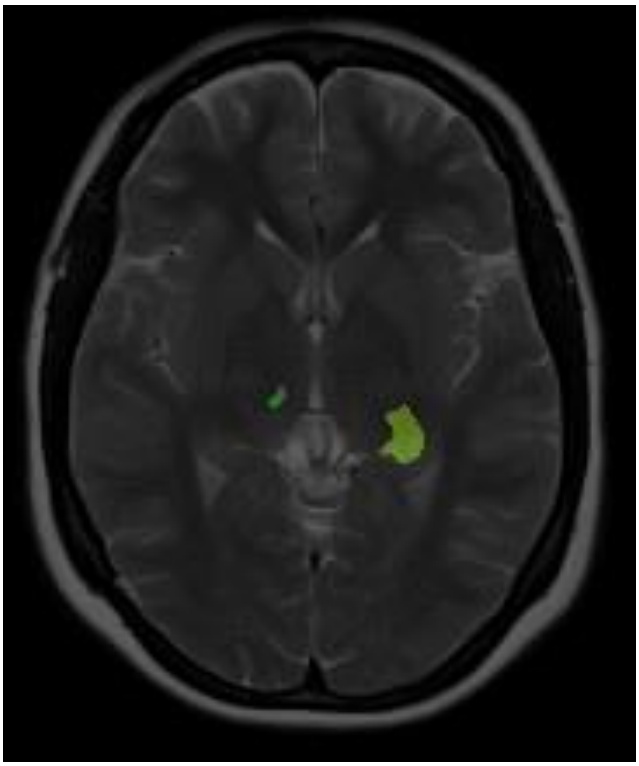
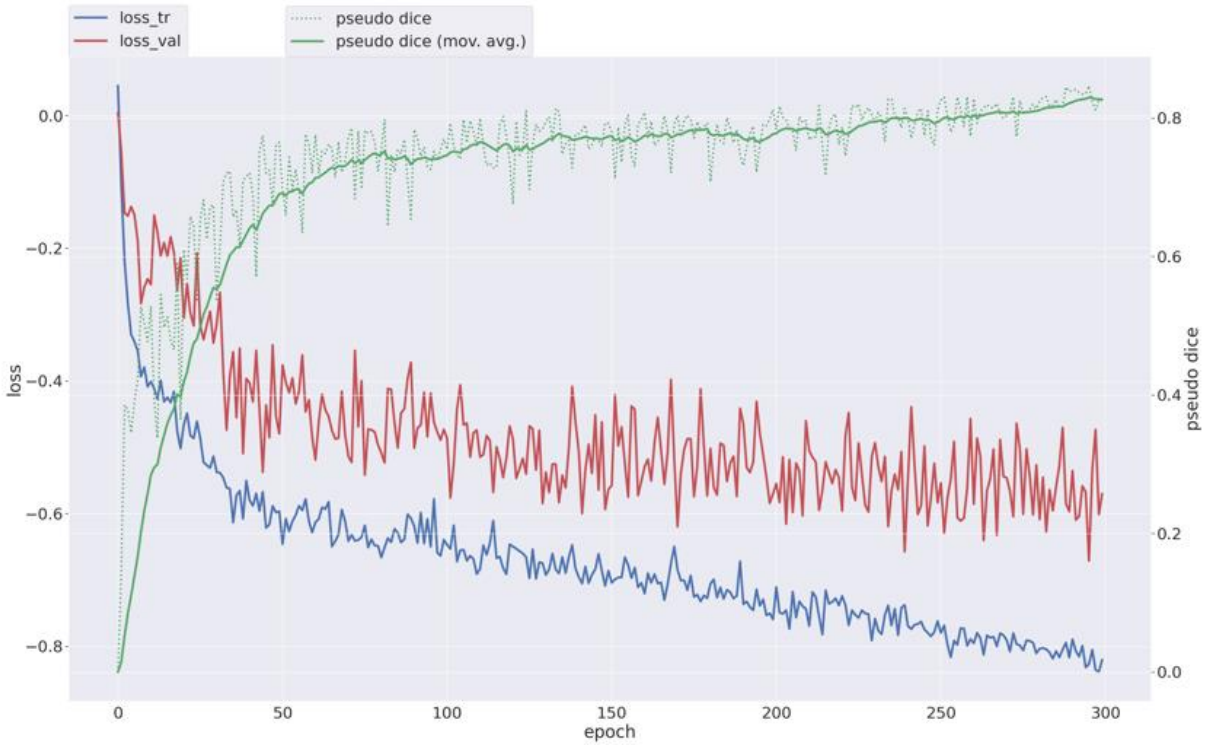


Figura 19. Entrenamiento 5 para cortes con nnUnet 2D.



Así mismo, En la Tabla IX se ilustran valores de 0.42 para el coeficiente de Dice en validación y de 0.37 de IoU, por lo que finalmente se opta por utilizar los hiperparámetro del modelo 1 para entrenar los 2060 cortes.

Tabla IX. Resultados del entrenamiento 5 para el modelo nnUnet con configuración 2D.

Dice	0.42
FN	24.9
FP	10.8
IoU	0.37
TN	118905
TP	87

D. Modelo con mejores hiperparámetros.

Dado que los mejores resultados obtenidos fueron para el entrenamiento 1(Learning rate inicial de 0.01, momento de 0.99, oversample_foreground_percent de 0.33, y un decaimiento de pesos igual a 3e-5), se realiza un último entrenamiento con 150 épocas y con los 2060 cortes, además, dado que de las gráficas se puede observar cierta inestabilidad en la curva de loss_tr y loss_val, se incrementa el bachtsize a un valor de 124.

En la Figura 20 se puede visualizar una de las predicciones arrojadas por el entrenamiento, en el entrenamiento se puede observar que se alcanza un coeficiente de Dice alrededor del 0.78 (Figura 21), además de que se puede observar que al aumentar el batchsize se pudo obtener un loss_val y un loss_tr más estables, aunque con un poco de desajuste, puesto que la gráfica de loss_tr converge en 0.0 y loss_val converge alrededor de 0.05.

Figura 20. Predicción de la lesión (Rojo) vs etiqueta de la lesión (verde) en el entrenamiento 2D con 2060 cortes en el corte 023.

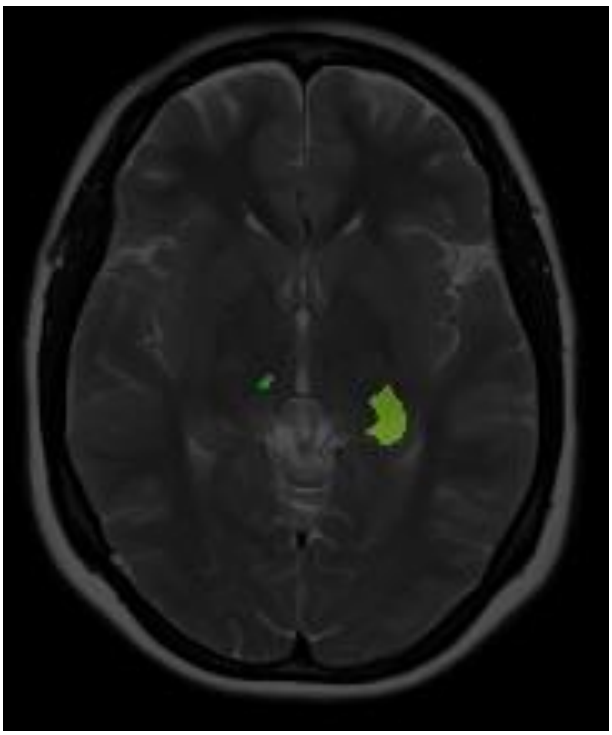
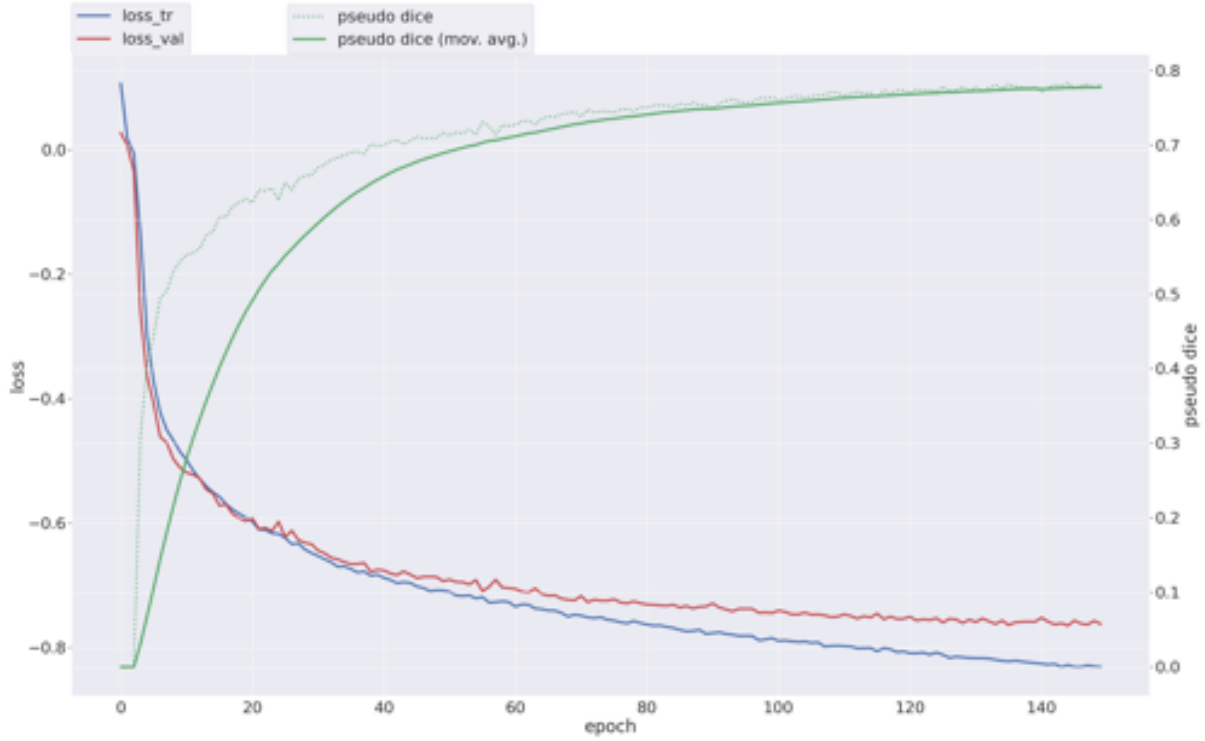


Figura 21. Entrenamiento para cortes 1340 con nnUnet 2D.



La Tabla X muestra los resultados para la validación del modelo final, en el cual se obtuvo un coeficiente de Dice de 0.59, y a diferencia del entrenamiento 1 en el cual se tenían 50 cortes, ara los 1340 cortes se pudo obtener un TP de 394 a diferencia del entrenamiento 1 (TP=319).

Tabla X. Resultados del entrenamiento 1 con 1340 cortes para el modelo nnUnet con configuración 2D.

Dice	0.59
FN	169
FP	53
IoU	0.49
TN	118410
TP	394

VI. Análisis.

Los resultados de la configuración 3D Fullres el modelo convolucional nnUnet no fueron los mejores puesto que no se alcanzó coeficiente de Dice altos para el entrenamiento, esto puede deberse a diversos factores, el primero se relaciona con la calidad de las imágenes, puesto que los volumétricos obtenidos del dataset fueron reconstrucciones del plano axial y la calidad para los planos sagital y coronal no eran los mejores por lo que es necesario de volumétricos en los que se pueda observar con buena calidad en todos los planos, por otro lado, no se realizó algún cambió del batchsize en esta configuración ni se aumentó el número de épocas o el número de sujetos dado que para entrenar el algoritmo hasta las 50 épocas, ya que el tiempo en promedio que se tomó para entrenar cada época oscilaba entre 11 Y 13 minutos, por lo que quizás se hubiese podido aumentar un poco el coeficiente de Dice con un número de épocas igual a 1000 o superior lo que consumiría mucho costo computacional por lo que se aconseja tener un GPU en la cual no se tengan limitación de tiempo impuestas por Google Colab, ya que este último presenta restricciones de tiempo y memoria lo cual dificulta el entrenamiento del modelo convolucional.

Como ya se mencionó anteriormente, se escogen los mejores 2060 cortes de los 29 volúmenes de pacientes con esclerosis múltiple de los cuales se utilizan inicialmente 50 para entrenar y validar el mejor modelo. El entrenamiento 2d 1 fue el que presento mejor resultados en la validación con un coeficiente de Dice igual a 0.84 (Tabla V), sin embargo, los valores que toman las curvas de loss_tr y loss_val presentaban inestabilidad obteniendo como resultado un sobreajuste del modelo que fue mucho más evidente en el entrenamiento 2, el cual se definió un valor más bajo de learning rate (0.001) con la intención de incrementar el coeficiente de Dice de forma que no generalice nuevos datos y el modelo no siga fluctuaciones aleatorias en el conjunto de datos de entrenamiento, Aun así, se obtuvo el coeficientes de Dice más bajo valor (0.28 (Tabla VI) en comparación con los otros entrenamientos, además de presentar bastante inestabilidad en las curvas loss_tr y loss_val, pues de la Figura 13 se puede observar como la perdida de entrenamiento converge a 0.0 mientras que la perdida de validación tiende a 0.4. Con el propósito de mejorar el sobreajuste del entrenamiento 2, se decrementa el valor que hace referencia al decaimiento de pesos a 3e-6, se aumenta el valor oversample_foreground_percent a 0.43 y se decrementa el momento a 0.98, esto con la finalidad de disminuir el sobreajuste y las discrepancias entre la función de perdida de entrenamiento y la función de perdida de validación, dicha mejora queda en evidencia en la Figura 15 ya que loss_tr y loss_val tiende a converger en 0.0, sin embargo, el coeficiente de

Dice (0.67) en la validación (Tabla VII) no supero el del entrenamiento 1, y esto puede deberse a que el modelo es menos sensible a las fluctuaciones aleatorias de los datos y esto a causa de la disminución del learning rate, por lo que en el entrenamiento 4, se deja el mismo learning rate inicial (1e-2) de manera que sea más sensible a los cambios aleatorio y el mismo decaimiento de pesos (3e-5), se decrementa el momento a 0.97 y se aumenta el oversample_foreground_percent a 0.43 con la intención de estabilizar la curva de perdida de entrenamiento y perdida de validación, a pesar de que en la Tabla VIII se observó un coeficiente de Dice mayor que en el entrenamiento 3 (0.68), las curvas loss_tr y loss_val tienden a tener valores diferentes (0.0 y 0.1 respectivamente) al final del entrenamiento. Por último, con la finalidad de corregir el sobreajuste en el entrenamiento 4 y al mismo tiempo obtener el coeficiente de Dice del entrenamiento 1 con la estabilidad en la funciones de perdida de entrenamiento y perdida de validación del entrenamiento 3, se entrena el algoritmo con un learning rate de 0.01, un decaimiento de pesos de 3e-6, un oversample_foreground_percent de 0.43 y un momento de 0.98 y sin modificar el batchsize, aún así, los resultados obtenidos no fueron mejores que para el entrenamiento 1 ya que el coeficiente de Dice en este entrenamiento tomo valores de alrededor 0.42 para la validación (Tabla IX) y esto puede deberse a la calidad y variabilidad de las imágenes, ya que por cuestiones de tiempo, se redujo la cantidad de cortes a 50 con la finalidad de reducir costo computacional y por ende el tiempo.

Así pues, como ya se mencionó previamente el entrenamiento 1 fue el que mejores resultados arrojo en la etapa de validación, con un coeficiente de Dice de 0.84, en general, los resultados de validación de la CNN nnUnet son bastante buenos (Tabla XI). El modelo tiene una alta tasa de detección de lesiones (sensibilidad), una baja tasa de falsos positivos (especificidad) y una buena precisión. La sensibilidad es una medida de la capacidad del modelo para detectar lesiones correctamente. En este caso, la sensibilidad es de 0.6998, lo que significa que el modelo tiene un 69.98% de probabilidades de detectar correctamente una lesión. Es de sensibilidad relativamente alta, lo que indica que el modelo detecta la mayoría de las lesiones. La especificidad es una medida de la capacidad del modelo para evitar falsos positivos. En este caso, la especificidad es de 0.9996, lo que significa que el modelo tiene un 99.96% de probabilidades de evitar falsos positivos. Esta es una especificidad muy alta, lo que indica que el modelo es muy poco probable que genere falsos positivos. La precisión es una medida de la capacidad del modelo para generar predicciones correctas. En este caso, la precisión es de 0.8814, lo que significa que el modelo tiene un 88.14% de probabilidades de generar una predicción correcta. Esta es una precisión relativamente alta, lo que indica que el

modelo es capaz de generar predicciones precisas. Aunque los resultados del entrenamiento de la CNN nnUnet indican que el modelo es capaz de detectar lesiones de esclerosis múltiple con un alto grado de precisión. Hay algunos aspectos en las que el modelo podría mejorar debido a que las curvas de perdida predicción y perdida de validación convergen en valores diferentes. Por otro lado, un área en la que el modelo podría mejorar es en la sensibilidad. La sensibilidad actual del modelo es de 0.6998, lo que significa que el modelo tiene un 30.02% de probabilidades de no detectar una lesión. Para mejorar la sensibilidad, el modelo podría entrenarse con un conjunto de datos más grande que incluya una mayor variedad de lesiones. Otra área en la que el modelo podría mejorar es en la tasa de falsos negativos. La tasa de falsos negativos actual del modelo es de 0.3002, lo que significa que el modelo tiene un 30.02% de probabilidades de generar una predicción negativa cuando hay una lesión presente. Para mejorar la tasa de falsos negativos, el modelo podría utilizar un algoritmo de clasificación más complejo, como un modelo de aprendizaje profundo que utiliza un enfoque de aprendizaje no supervisado.

Tabla XI. Métricas y matriz de confusión para el entrenamiento final.

Medida	Valor	Derivación
Sensibilidad	0.6998	TPR = TP / (TP + FN)
Especificidad	0.9996	SPC = TN / (FP + TN)
precisión	0.8814	PPV = TP / (TP + FP)
Valor predictivo negativo	0.9986	NPV = TN / (TN + FN)
Tasa de falsos positivos	0.0004	FPR = FP / (FP + TN)
Tasa de descubrimiento falso	0.1186	FDR = FP / (FP + TP)
Tasa de falsos negativos	0.3002	FNR = FN / (FN + TP)
Exactitud	0.9981	ACC = (TP + TN) / (P + N)
F1 Score	0.7802	F1 = 2TP / (2TP + FP + FN)
Coeficiente de correlación de Matthews	0.7845	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Hay varios hiperparámetros que se pueden modificar para intentar mejorar los resultados del entrenamiento de la CNN nnUnet. Algunos de estos hiperparámetros incluyen:

Un conjunto de datos de entrenamiento más grande puede ayudar a mejorar la precisión y la sensibilidad del modelo, además, de que sean volumétricos con una buena calidad en el plano sagital, coronal y transversal, que permita evaluar las lesiones con diferentes configuraciones de nnUnet como la es la configuración Fullres, Lowres y 3d_cascade_fullres.

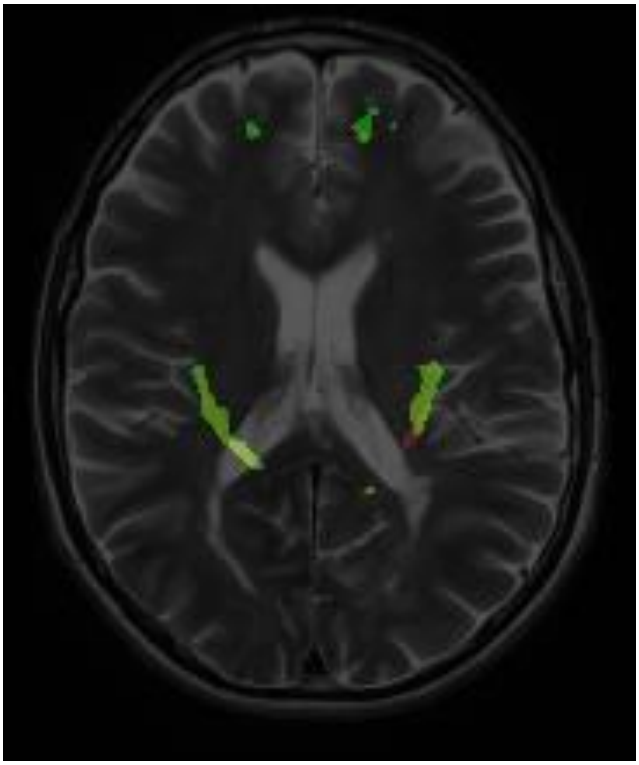
Por otro lado, se podría implementar la opción de Data Augmentation definida en el archivo nnUNetTrainer.py, no obstante, esto también puede aumentar el tiempo de entrenamiento del algoritmo.

Pese a que el cambio de los hiperparámetros de entrenamiento, como el learning rate inicial, el momento, el oversample_foreground_percent y el decaimiento de pesos no afecto el rendimiento del modelo de manera adecuada, existen otros hiperparámetros que pueden ser modificados en el archivo nnUNetTrainer.py como lo son num_iterations_per_epoch que determina el número de veces que el modelo se itera sobre el conjunto de datos de entrenamiento por lo que un número más alto significa que el modelo tendrá más oportunidades de aprender de los datos y num_val_iterations_per_epoch que determina el número de veces que el modelo se itera sobre el conjunto de datos de validación por lo que un valor más alto para este hiperparámetro significa que el modelo tendrá más oportunidades de evaluar su rendimiento en datos nuevos. Aun así, valores demasiados altos para estos hiperparámetros pueden inferior en aumento de tiempo del entrenamiento.

La arquitectura de la CNN es otro factor para considerar, ya que en la literatura se pueden encontrar resultados, una arquitectura de CNN más compleja puede ayudar a mejorar la precisión y la sensibilidad del modelo, por ejemplo, en la literatura se pueden encontrar modelos con mejores resultados (DeepScan), la arquitectura de DeepScan es similar a la de nnUnet, pero tiene algunas diferencias clave. En particular, DeepScan utiliza un diseño de red más profunda con más capas convoluciones. Esto le permite a DeepScan aprender características más complejas de las imágenes de MRI. Por lo que se podría pensar en capas más profundas en la arquitectura de la nnUnet esto podría hacerse agregando más capas de convoluciones a la red o aumentando la profundidad de las capas existentes.

Por último, se pueden observar la ilustración en el momento de predicción lesiones, de los cuales no se obtuvieron los mejores resultados, pues solo se puede observar que predice la lesión sobre un paciente (Figura 22). Esto puede deberse a un sobre ajuste en el entrenamiento y una falta de predicción en fluctuaciones, es decir, el tamaño de las lesiones.

Figura 22. Predicción del modelo 2D con hiperparámetros modificados.



VII. Conclusiones

Si bien el batchsize es un hiperparámetro importante que puede ayudar a estabilizar la función de perdida de entrenamiento y la función de perdida de validación, en consecuencia, un mayor número de batchsize puede ayudar a evitar el sobreajuste, pero esto implica un mayor consumo computacional. Por otro lado, disminuir el learning rate no ayudo a mejorar el coeficiente de Dice. Pero, se demostró que disminuir el momento y decaimiento de pesos, reduce el sobreajuste de los datos y estabiliza las curvas de perdida de entrenamiento y perdida de validación por lo que la variación de dichos parámetros puede ser fundamentales para obtener mejores resultados a la hora de probar diferentes configuraciones de la red.

Por lo que además del tamaño del conjunto de datos de entrenamiento puede afectar el rendimiento del modelo. Un conjunto de datos de entrenamiento más grande incluye una mayor variedad de lesiones, así mismo, el tipo de imágenes con la cual se desea entrenar el algoritmo puede influir en los resultados obtenidos, puesto que inicialmente se trabajó con reconstrucciones en el plano axial y esto limito el entrenamiento a un solo tipo de configuración de la nnUnet (2D) por lo que trabajar con volumétricos facilita el uso de otro tipo de configuraciones (Fullres o lowres,3d cascade fullres) que puedan arrojar mejores resultados y ayudar al modelo a aprender a detectar lesiones de diferentes tamaños y formas. Pese a esto, si hay pocos recursos computacionales, hay que limitar la información con la que se entrenará el modelo, ya que los entrenamientos pueden tomar mucho tiempo.

La precisión del modelo en la clasificación de imágenes con EM es más baja que la precisión en la clasificación de imágenes sanas. Esto podría deberse a que el modelo no esté expuesto a suficientes imágenes de EM durante el entrenamiento. Para mejorar la precisión del modelo en la clasificación de imágenes con EM, se podría aumentar el tamaño del conjunto de datos de entrenamiento para incluir más imágenes de EM.

El modelo nnUnet es una herramienta prometedora para la segmentación de lesiones de esclerosis múltiple. El modelo mostró una sensibilidad del 69, 98%, una especificidad del 99, 96% y una precisión del 88, 14% en el conjunto de datos de validación. Estos resultados son comparables a los de otros modelos de aprendizaje profundo utilizados para esta tarea. Aún así, La predicción indica que el algoritmo puede estar ajustado por lo que aumentar la variedad de lesiones y trabajar con mejores recursos computacionales es fundamental para futuros entrenamientos.

Además de eso, el modelo aún tiene algunas áreas en las que podría mejorar. En particular, la sensibilidad del modelo podría ser mayor. Esto significa que el modelo tiene un

30,02 % de probabilidades de no detectar una lesión. Por tanto, para mejorar la tasa de falsos negativos, el modelo podría utilizar un algoritmo de clasificación más complejo, como un modelo de aprendizaje profundo que utiliza un enfoque de aprendizaje no supervisado. Por lo que se podría considerar en utilizar una arquitectura de CNN más compleja. Una arquitectura más compleja podría aprender características más complejas de las imágenes de resonancia magnética, lo que podría ayudar al modelo a mejorar su rendimiento y a su vez mejorar su sensibilidad.

En conclusión, el modelo nnUnet es una herramienta prometedora para la segmentación de lesiones de esclerosis múltiple. Hay aspectos que pueden ayudar a mejorar los resultados en la predicción, como la cantidad y el tipo de imágenes, la configuración, la profundidad del modelo convolucional y los hiperparámetros a utilizar obteniendo mejoras en la sensibilidad, el sobreajuste y la tasa de falsos negativos.

VIII. Referencias.

Arai, K., & Lyubchich, V. (2022). Modern Data Science with R (2nd ed.). *Technometrics*, *64*(3), 429–429. https://doi.org/10.1080/00401706.2022.2087421

Avants, B., Tustison, N. J., & Song, G. (2022). Advanced Normalization Tools: V1.0. *The Insight Journal*. https://doi.org/10.54294/UVNHIN

Basaran, B. D., Matthews, P. M., & Bai, W. (2022). New lesion segmentation for multiple sclerosis brain images with imaging and lesion-aware augmentation. *Frontiers in Neuroscience*, *16*. https://doi.org/10.3389/FNINS.2022.1007453

Bodini, B., Battaglini, M., De Stefano, N., Khaleeli, Z., Barkhof, F., Chard, D., Filippi, M., Montalban, X., Polman, C., Rovaris, M., Rovira, A., Samson, R., Miller, D., Thompson, A., & Ciccarelli, O. (2011a). T2 lesion location really matters: A 10 year follow-up study in primary progressive multiple sclerosis. *Journal of Neurology, Neurosurgery and Psychiatry*, *82*(1), 72–77. https://doi.org/10.1136/JNNP.2009.201574

Bodini, B., Battaglini, M., De Stefano, N., Khaleeli, Z., Barkhof, F., Chard, D., Filippi, M., Montalban, X., Polman, C., Rovaris, M., Rovira, A., Samson, R., Miller, D., Thompson, A., & Ciccarelli, O. (2011b). T2 lesion location really matters: A 10 year follow-up study in primary progressive multiple sclerosis. *Journal of Neurology, Neurosurgery and Psychiatry*, *82*(1), 72–77. https://doi.org/10.1136/JNNP.2009.201574

Boutet, A., Loh, A., Chow, C. T., Taha, A., Elias, G. J. B., Neudorfer, C., Germann, J., Paff, M., Zrinzo, L., Fasano, A., Kalia, S. K., Steele, C. J., Mikulis, D., Kucharczyk, W., & Lozano, A. M. (2021). A literature review of magnetic resonance imaging sequence advancements in visualizing functional neurosurgery targets. En *Journal of Neurosurgery* (Vol. 135, Número 5, pp. 1445–1458). American Association of Neurological Surgeons. https://doi.org/10.3171/2020.8.JNS201125

Buccafusca, M. (2014). Clinical and magnetic resonance imaging predictors of disease progression in multiple sclerosis: a nine-year follow-up study. *Multiple Sclerosis Journal*. https://www.academia.edu/13810983/Clinical_and_magnetic_resonance_imaging_predictors_of_disease_progression_in_multiple_sclerosis_a_nine_year_follow_up_study

Dworkin, J. D., Linn, K. A., Oguz, I., Fleishman, G. M., Bakshi, R., Nair, G., Calabresi, P. A., Henry, R. G., Oh, J., Papinutto, N., Pelletier, D., Rooney, W., Stern, W., Sicotte, N. L., Reich, D. S., & Shinohara, R. T. (2018). An Automated Statistical Technique for Counting Distinct Multiple Sclerosis Lesions. *AJNR. American journal of neuroradiology*, *39*(4), 626–633. https://doi.org/10.3174/AJNR.A5556

García-Lorenzo, D., Francis, S., Narayanan, S., Arnold, D. L., & Collins, D. L. (2013). Review of automatic segmentation methods of multiple sclerosis white matter lesions on conventional magnetic resonance imaging. *Medical Image Analysis*, *17*(1), 1–18. https://doi.org/10.1016/J.MEDIA.2012.09.004

Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., & Ghosh, S. S. (2011a). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Frontiers in neuroinformatics*, *5*. https://doi.org/10.3389/FNINF.2011.00013

Gorgolewski, K., Burns, C. D., Madison, C., Clark, D., Halchenko, Y. O., Waskom, M. L., & Ghosh, S. S. (2011b). Nipype: A flexible, lightweight and extensible neuroimaging data processing framework in Python. *Frontiers in Neuroinformatics*, *5*. https://doi.org/10.3389/FNINF.2011.00013

Hata, J., Nakae, K., Tsukada, H., Woodward, A., Haga, Y., Iida, M., Uematsu, A., Seki, F., Ichinohe, N., Gong, R., Kaneko, T., Yoshimaru, D., Watakabe, A., Abe, H., Tani, T., Hamda, H. T., Gutierrez, C. E., Skibbe, H., Maeda, M., … Okano, H. (2023). Multi-modal brain magnetic resonance imaging database covering marmosets with a wide age range. *Scientific Data*, *10*(1). https://doi.org/10.1038/S41597-023-02121-2

Isensee, F., Jaeger, P. F., Kohl, S. A. A., Petersen, J., & Maier-Hein, K. H. (2020). nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods 2020 18:2*, *18*(2), 203–211. https://doi.org/10.1038/s41592-020-01008-z

Jenkinson, M., Beckmann, C. F., Behrens, T. E. J., Woolrich, M. W., & Smith, S. M. (s/f-a). *FLIRT/FAQ"/l "How_do_I_transform_a_mask_with_FLIRT_from_one_space_to_another.3F - FslWiki*. Recuperado el 17 de enero de 2024, de

https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FLIRT/FAQ%22%20/l%20%22How_do_I_transform_a_mask_with_FLIRT_from_one_space_to_another.3F

Jenkinson, M., Beckmann, C. F., Behrens, T. E. J., Woolrich, M. W., & Smith, S. M. (s/f-b). *Section 5.2: Example Box*. Recuperado el 17 de enero de 2024, de https://www.fmrib.ox.ac.uk/primers/intro_primer/ExBox17/IntroBox17.html

Jenkinson, M., Beckmann, C. F., Behrens, T. E. J., Woolrich, M. W., & Smith, S. M. (2012). FSL. *NeuroImage*, *62*(2), 782–790. https://doi.org/10.1016/J.NEUROIMAGE.2011.09.015

McAtee Ian. (s/f). *Converting a NIfTI File to an Image Sequence Using Python – Ian McAtee*. Recuperado el 17 de enero de 2024, de https://ianmcatee.com/converting-a-nifti-file-to-an-image-sequence-using-python/

McKinley, R., Wepfer, R., Grunder, L., Aschwanden, F., Fischer, T., Friedli, C., Muri, R., Rummel, C., Verma, R., Weisstanner, C., Wiestler, B., Berger, C., Eichinger, P., Muhlau, M., Reyes, M., Salmen, A., Chan, A., Wiest, R., & Wagner, F. (2020). Automatic detection of lesion load change in Multiple Sclerosis using convolutional neural networks with segmentation confidence. *NeuroImage: Clinical*, *25*, 102104. https://doi.org/10.1016/J.NICL.2019.102104

Murphy, A., & Gaillard, F. (2015). MRI sequences (overview). *Radiopaedia.org*. https://doi.org/10.53347/RID-37346

Roscheck Florian. (s/f). *blend-modes · PyPI*. Recuperado el 17 de enero de 2024, de https://pypi.org/project/blend-modes/

Sled, J. G., Zijdenbos, A. P., & Evans, A. C. (1998). A nonparametric method for automatic correction of intensity nonuniformity in mri data. *IEEE Transactions on Medical Imaging*, *17*(1), 87–97. https://doi.org/10.1109/42.668698

IX. **Anexos.**

Anexo A. Parámetros de entrenamiento para el modelo nnUnet con configuración 2D.

	Paciente	**Cantidad de Cortes**	**CortesEnPacientes/CortesRenombrados**
0	1	35	['034_1', '035_2', '036_3', '037_4', '038_5', '039_6', '040_7', '041_8', '042_9', '043_10', '058_11', '059_12', '060_13', '061_14', '062_15', '063_16', '064_17', '065_18', '066_19', '067_20', '068_21', '069_22', '070_23', '071_24', '072_25', '073_26', '074_27', '075_28', '076_29', '077_30', '078_31', '079_32', '080_33', '081_34', '082_35']
1	10	48	['084_36', '085_37', '086_38', '087_39', '088_40', '089_41', '090_42', '091_43', '092_44', '093_45', '094_46', '095_47', '096_48', '097_49', '098_50', '102_51', '103_52', '104_53', '105_54', '106_55', '107_56', '108_57', '109_58', '110_59', '111_60', '112_61', '113_62', '120_63', '121_64', '122_65', '123_66', '124_67', '125_68', '126_69', '127_70', '128_71', '129_72', '130_73', '131_74', '132_75', '133_76', '134_77', '135_78', '136_79', '137_80', '138_81', '139_82', '140_83']
2	11	82	['044_84', '045_85', '046_86', '047_87', '048_88', '049_89', '050_90', '051_91', '052_92', '053_93', '054_94', '055_95', '056_96', '057_97', '058_98', '059_99', '060_100', '061_101', '062_102', '063_103', '064_104', '065_105', '066_106', '067_107', '068_108', '069_109', '070_110', '071_111', '072_112', '073_113', '074_114', '075_115', '076_116', '077_117', '078_118', '079_119', '080_120', '081_121', '082_122', '083_123', '084_124', '085_125', '086_126', '087_127', '088_128', '089_129', '090_130', '091_131', '092_132', '093_133', '094_134', '095_135', '096_136', '097_137', '098_138', '099_139', '100_140', '101_141', '102_142', '103_143', '104_144', '105_145', '106_146', '107_147', '108_148', '109_149', '110_150', '111_151', '112_152', '113_153', '114_154', '115_155', '116_156', '117_157', '118_158', '119_159', '120_160', '121_161', '122_162', '123_163', '124_164', '125_165']
3	13	37	['059_166', '060_167', '061_168', '062_169', '063_170', '064_171', '065_172', '066_173', '067_174', '068_175', '080_176', '081_177', '082_178', '083_179', '084_180', '085_181', '086_182', '092_183', '093_184', '094_185', '095_186', '096_187', '097_188', '098_189', '101_190', '102_191', '103_192', '104_193', '105_194', '106_195', '107_196', '108_197', '109_198', '110_199', '111_200', '112_201', '113_202']
4	18	57	['089_203', '090_204', '091_205', '092_206', '093_207', '094_208', '095_209', '096_210', '097_211', '098_212', '099_213', '100_214', '101_215', '102_216', '103_217', '104_218', '105_219', '106_220', '107_221', '108_222', '109_223', '110_224', '111_225', '112_226', '113_227', '114_228', '115_229', '116_230', '117_231', '118_232', '119_233', '120_234', '121_235', '122_236', '123_237', '124_238', '125_239', '126_240', '127_241', '128_242', '129_243', '130_244', '131_245', '132_246',

			'133_247', '134_248', '135_249', '136_250', '137_251', '138_252', '139_253', '140_254', '141_255', '142_256', '143_257', '144_258', '145_259']
5	19	97	['065_260', '066_261', '067_262', '068_263', '069_264', '070_265', '071_266', '072_267', '073_268', '074_269', '075_270', '076_271', '077_272', '078_273', '079_274', '080_275', '081_276', '082_277', '083_278', '084_279', '085_280', '086_281', '087_282', '088_283', '089_284', '090_285', '091_286', '092_287', '093_288', '094_289', '095_290', '096_291', '097_292', '098_293', '099_294', '100_295', '101_296', '102_297', '103_298', '104_299', '105_300', '106_301', '107_302', '108_303', '109_304', '110_305', '111_306', '112_307', '113_308', '114_309', '115_310', '116_311', '117_312', '118_313', '119_314', '120_315', '121_316', '122_317', '123_318', '124_319', '125_320', '126_321', '127_322', '128_323', '129_324', '130_325', '131_326', '132_327', '133_328', '134_329', '135_330', '136_331', '137_332', '138_333', '139_334', '140_335', '141_336', '142_337', '143_338', '144_339', '145_340', '146_341', '147_342', '148_343', '149_344', '150_345', '151_346', '152_347', '153_348', '154_349', '155_350', '156_351', '157_352', '158_353', '159_354', '160_355', '161_356']
6	20	22	['083_357', '084_358', '085_359', '086_360', '087_361', '088_362', '089_363', '090_364', '091_365', '092_366', '093_367', '094_368', '095_369', '096_370', '097_371', '098_372', '099_373', '100_374', '101_375', '102_376', '103_377', '104_378']
7	22	49	['062_379', '063_380', '064_381', '078_382', '079_383', '080_384', '081_385', '082_386', '083_387', '084_388', '085_389', '086_390', '087_391', '088_392', '089_393', '090_394', '091_395', '092_396', '099_397', '100_398', '101_399', '102_400', '103_401', '104_402', '105_403', '106_404', '107_405', '108_406', '109_407', '110_408', '111_409', '112_410', '113_411', '114_412', '115_413', '116_414', '117_415', '118_416', '119_417', '120_418', '121_419', '122_420', '123_421', '124_422', '125_423', '126_424', '127_425', '146_426', '147_427']
8	23	46	['096_428', '097_429', '098_430', '099_431', '100_432', '101_433', '102_434', '103_435', '104_436', '105_437', '106_438', '107_439', '108_440', '109_441', '110_442', '111_443', '112_444', '113_445', '114_446', '115_447', '116_448', '117_449', '118_450', '119_451', '120_452', '121_453', '122_454', '123_455', '124_456', '125_457', '126_458', '127_459', '128_460', '129_461', '130_462', '131_463', '132_464', '133_465', '134_466', '135_467', '136_468', '137_469', '138_470', '139_471', '140_472', '141_473']

	9	24	108	['065_474', '066_475', '067_476', '068_477', '069_478', '070_479', '071_480', '072_481', '073_482', '074_483', '075_484', '076_485', '077_486', '078_487', '079_488', '080_489', '081_490', '082_491', '083_492', '084_493', '085_494', '086_495', '087_496', '088_497', '089_498', '090_499', '091_500', '092_501', '093_502', '094_503', '095_504', '096_505', '097_506', '098_507', '099_508', '100_509', '101_510', '102_511', '103_512', '104_513', '105_514', '106_515', '107_516', '108_517', '109_518', '110_519', '111_520', '112_521', '113_522', '114_523', '115_524', '116_525', '117_526', '118_527', '119_528', '120_529', '121_530', '122_531', '123_532', '124_533', '125_534', '126_535', '127_536', '128_537', '129_538', '130_539', '131_540', '132_541', '133_542', '134_543', '135_544', '136_545', '137_546', '138_547', '139_548', '140_549', '141_550', '142_551', '143_552', '144_553', '145_554', '146_555', '147_556', '148_557', '149_558', '150_559', '151_560', '152_561', '153_562', '154_563', '155_564', '156_565', '157_566', '158_567', '159_568', '160_569', '161_570', '162_571', '163_572', '164_573', '165_574', '166_575', '167_576', '168_577', '169_578', '170_579', '171_580', '172_581']
0	**1**	25	94	['089_582', '090_583', '091_584', '092_585', '093_586', '094_587', '095_588', '096_589', '097_590', '098_591', '099_592', '100_593', '101_594', '102_595', '103_596', '104_597', '105_598', '106_599', '107_600', '108_601', '109_602', '110_603', '111_604', '112_605', '113_606', '114_607', '115_608', '116_609', '117_610', '118_611', '119_612', '120_613', '121_614', '122_615', '123_616', '124_617', '125_618', '126_619', '127_620', '128_621', '129_622', '130_623', '131_624', '132_625', '133_626', '134_627', '135_628', '136_629', '137_630', '138_631', '139_632', '140_633', '141_634', '142_635', '143_636', '144_637', '145_638', '146_639', '147_640', '148_641', '149_642', '150_643', '151_644', '152_645', '153_646', '154_647', '155_648', '156_649', '157_650', '158_651', '159_652', '160_653', '161_654', '162_655', '163_656', '164_657', '165_658', '166_659', '167_660', '168_661', '169_662', '170_663', '171_664', '172_665', '173_666', '174_667', '175_668', '176_669', '177_670', '178_671', '179_672', '180_673', '181_674', '182_675']
1	**1**	27	74	['046_676', '047_677', '048_678', '049_679', '050_680', '051_681', '052_682', '053_683', '054_684', '055_685', '056_686', '057_687', '058_688', '059_689', '060_690', '061_691', '062_692', '063_693', '064_694', '065_695', '066_696', '067_697', '068_698', '069_699', '070_700', '071_701', '072_702', '073_703', '074_704', '075_705', '076_706', '077_707', '078_708', '079_709', '080_710', '081_711', '082_712', '083_713', '084_714', '085_715', '086_716', '087_717', '088_718', '089_719', '090_720', '091_721', '092_722', '093_723', '094_724', '095_725', '096_726', '097_727', '098_728', '099_729', '100_730', '101_731', '102_732', '103_733', '104_734', '105_735', '106_736', '107_737', '108_738', '109_739', '110_740', '111_741', '112_742', '113_743', '114_744', '115_745', '116_746', '117_747', '118_748', '119_749']

2	**1**	28	93	['043_750', '044_751', '045_752', '060_753', '061_754', '062_755', '063_756', '064_757', '065_758', '066_759', '067_760', '068_761', '069_762', '070_763', '071_764', '072_765', '073_766', '074_767', '075_768', '076_769', '077_770', '078_771', '079_772', '080_773', '081_774', '082_775', '083_776', '084_777', '085_778', '086_779', '087_780', '088_781', '089_782', '090_783', '091_784', '092_785', '093_786', '094_787', '095_788', '096_789', '097_790', '098_791', '099_792', '100_793', '101_794', '102_795', '103_796', '104_797', '105_798', '106_799', '107_800', '108_801', '109_802', '110_803', '111_804', '112_805', '113_806', '114_807', '115_808', '116_809', '117_810', '118_811', '119_812', '120_813', '121_814', '122_815', '123_816', '124_817', '125_818', '126_819', '127_820', '128_821', '129_822', '130_823', '131_824', '132_825', '133_826', '134_827', '135_828', '136_829', '137_830', '138_831', '139_832', '140_833', '141_834', '142_835', '143_836', '144_837', '145_838', '146_839', '147_840', '148_841', '149_842']
3	**1**	29	98	['036_843', '037_844', '038_845', '039_846', '040_847', '041_848', '042_849', '043_850', '044_851', '045_852', '046_853', '047_854', '048_855', '049_856', '050_857', '051_858', '052_859', '053_860', '054_861', '055_862', '056_863', '057_864', '058_865', '059_866', '060_867', '061_868', '062_869', '063_870', '064_871', '065_872', '066_873', '067_874', '068_875', '069_876', '070_877', '071_878', '072_879', '073_880', '074_881', '075_882', '076_883', '077_884', '078_885', '079_886', '080_887', '081_888', '082_889', '083_890', '084_891', '085_892', '086_893', '087_894', '088_895', '089_896', '090_897', '091_898', '092_899', '093_900', '094_901', '095_902', '096_903', '097_904', '098_905', '099_906', '100_907', '101_908', '102_909', '103_910', '104_911', '105_912', '106_913', '107_914', '108_915', '109_916', '110_917', '111_918', '112_919', '113_920', '114_921', '115_922', '116_923', '117_924', '118_925', '119_926', '120_927', '121_928', '122_929', '123_930', '124_931', '125_932', '126_933', '127_934', '128_935', '129_936', '130_937', '131_938', '132_939', '133_940']
4	**1**	3	59	['044_941', '045_942', '046_943', '057_944', '058_945', '059_946', '060_947', '061_948', '067_949', '068_950', '069_951', '070_952', '071_953', '072_954', '073_955', '074_956', '075_957', '076_958', '077_959', '078_960', '079_961', '080_962', '081_963', '082_964', '083_965', '084_966', '085_967', '086_968', '087_969', '088_970', '089_971', '090_972', '091_973', '092_974', '093_975', '094_976', '095_977', '096_978', '097_979', '098_980', '099_981', '100_982', '101_983', '102_984', '103_985', '104_986', '105_987', '106_988', '107_989', '108_990', '109_991', '115_992', '118_993', '128_994', '129_995', '130_996', '131_997', '132_998', '133_999']

15	30	98	['023_1000', '024_1001', '025_1002', '026_1003', '027_1004', '028_1005', '029_1006', '030_1007', '031_1008', '032_1009', '033_1010', '034_1011', '035_1012', '036_1013', '037_1014', '038_1015', '039_1016', '040_1017', '041_1018', '042_1019', '043_1020', '044_1021', '045_1022', '046_1023', '047_1024', '048_1025', '049_1026', '050_1027', '051_1028', '052_1029', '053_1030', '054_1031', '055_1032', '056_1033', '057_1034', '058_1035', '059_1036', '060_1037', '061_1038', '062_1039', '063_1040', '064_1041', '065_1042', '066_1043', '067_1044', '068_1045', '069_1046', '070_1047', '071_1048', '072_1049', '073_1050', '074_1051', '075_1052', '076_1053', '077_1054', '078_1055', '079_1056', '080_1057', '081_1058', '082_1059', '083_1060', '084_1061', '085_1062', '086_1063', '087_1064', '088_1065', '089_1066', '090_1067', '091_1068', '092_1069', '093_1070', '094_1071', '095_1072', '096_1073', '097_1074', '098_1075', '099_1076', '100_1077', '101_1078', '102_1079', '103_1080', '104_1081', '105_1082', '106_1083', '107_1084', '108_1085', '109_1086', '110_1087', '111_1088', '112_1089', '113_1090', '114_1091', '115_1092', '116_1093', '117_1094', '118_1095', '119_1096', '120_1097']
16	31	103	['041_1098', '042_1099', '043_1100', '044_1101', '045_1102', '046_1103', '047_1104', '048_1105', '049_1106', '050_1107', '051_1108', '052_1109', '053_1110', '054_1111', '055_1112', '056_1113', '063_1114', '064_1115', '065_1116', '066_1117', '067_1118', '068_1119', '069_1120', '070_1121', '071_1122', '072_1123', '073_1124', '074_1125', '075_1126', '076_1127', '077_1128', '078_1129', '079_1130', '080_1131', '081_1132', '082_1133', '083_1134', '084_1135', '085_1136', '086_1137', '087_1138', '088_1139', '089_1140', '090_1141', '091_1142', '092_1143', '093_1144', '094_1145', '095_1146', '096_1147', '097_1148', '098_1149', '099_1150', '100_1151', '101_1152', '102_1153', '103_1154', '104_1155', '105_1156', '106_1157', '107_1158', '108_1159', '109_1160', '110_1161', '111_1162', '112_1163', '113_1164', '114_1165', '115_1166', '116_1167', '117_1168', '118_1169', '119_1170', '120_1171', '121_1172', '122_1173', '123_1174', '124_1175', '125_1176', '126_1177', '127_1178', '128_1179', '129_1180', '130_1181', '131_1182', '132_1183', '133_1184', '134_1185', '135_1186', '136_1187', '137_1188', '138_1189', '139_1190', '140_1191', '141_1192', '142_1193', '143_1194', '144_1195', '145_1196', '146_1197', '147_1198', '148_1199', '149_1200']

7	**1**	33	91	['074_1201', '075_1202', '076_1203', '077_1204', '078_1205', '079_1206', '080_1207', '081_1208', '082_1209', '083_1210', '084_1211', '085_1212', '086_1213', '087_1214', '088_1215', '089_1216', '090_1217', '091_1218', '092_1219', '093_1220', '094_1221', '095_1222', '096_1223', '097_1224', '098_1225', '099_1226', '100_1227', '101_1228', '102_1229', '103_1230', '104_1231', '105_1232', '106_1233', '107_1234', '108_1235', '109_1236', '110_1237', '111_1238', '112_1239', '113_1240', '114_1241', '115_1242', '116_1243', '117_1244', '118_1245', '119_1246', '120_1247', '121_1248', '122_1249', '123_1250', '124_1251', '125_1252', '126_1253', '127_1254', '128_1255', '129_1256', '130_1257', '131_1258', '132_1259', '133_1260', '134_1261', '135_1262', '136_1263', '137_1264', '138_1265', '139_1266', '140_1267', '141_1268', '142_1269', '143_1270', '144_1271', '145_1272', '146_1273', '147_1274', '148_1275', '149_1276', '150_1277', '151_1278', '152_1279', '153_1280', '154_1281', '155_1282', '156_1283', '157_1284', '158_1285', '159_1286', '160_1287', '161_1288', '162_1289', '163_1290', '164_1291']
8	**1**	34	93	['041_1292', '042_1293', '043_1294', '044_1295', '045_1296', '046_1297', '047_1298', '048_1299', '049_1300', '050_1301', '051_1302', '052_1303', '053_1304', '054_1305', '055_1306', '056_1307', '057_1308', '058_1309', '059_1310', '060_1311', '061_1312', '062_1313', '063_1314', '064_1315', '065_1316', '066_1317', '067_1318', '068_1319', '069_1320', '070_1321', '071_1322', '072_1323', '073_1324', '074_1325', '075_1326', '076_1327', '077_1328', '078_1329', '079_1330', '080_1331', '081_1332', '082_1333', '083_1334', '084_1335', '085_1336', '086_1337', '087_1338', '088_1339', '089_1340', '090_1341', '091_1342', '092_1343', '093_1344', '094_1345', '095_1346', '096_1347', '097_1348', '098_1349', '099_1350', '100_1351', '101_1352', '102_1353', '103_1354', '104_1355', '105_1356', '106_1357', '107_1358', '108_1359', '109_1360', '110_1361', '111_1362', '112_1363', '113_1364', '114_1365', '115_1366', '116_1367', '117_1368', '118_1369', '119_1370', '120_1371', '121_1372', '122_1373', '123_1374', '124_1375', '125_1376', '126_1377', '127_1378', '128_1379', '129_1380', '130_1381', '131_1382', '132_1383', '133_1384']
9	**1**	35	83	['050_1385', '051_1386', '052_1387', '053_1388', '054_1389', '055_1390', '056_1391', '057_1392', '058_1393', '059_1394', '060_1395', '061_1396', '062_1397', '063_1398', '064_1399', '065_1400', '066_1401', '067_1402', '068_1403', '069_1404', '070_1405', '071_1406', '077_1407', '078_1408', '079_1409', '080_1410', '081_1411', '082_1412', '083_1413', '084_1414', '085_1415', '086_1416', '087_1417', '088_1418', '089_1419', '090_1420', '091_1421', '092_1422', '093_1423', '094_1424', '095_1425', '096_1426', '097_1427', '098_1428', '099_1429', '100_1430', '101_1431',

				'102_1432', '103_1433', '104_1434', '105_1435', '106_1436', '107_1437', '108_1438', '109_1439', '110_1440', '111_1441', '112_1442', '113_1443', '114_1444', '115_1445', '116_1446', '117_1447', '118_1448', '119_1449', '120_1450', '121_1451', '122_1452', '123_1453', '124_1454', '125_1455', '126_1456', '127_1457', '128_1458', '129_1459', '130_1460', '131_1461', '132_1462', '133_1463', '134_1464', '135_1465', '136_1466', '137_1467']
0	**2**	38	86	['033_1468', '034_1469', '035_1470', '036_1471', '037_1472', '038_1473', '039_1474', '040_1475', '041_1476', '042_1477', '043_1478', '044_1479', '045_1480', '046_1481', '047_1482', '048_1483', '049_1484', '050_1485', '051_1486', '052_1487', '053_1488', '054_1489', '055_1490', '056_1491', '057_1492', '058_1493', '059_1494', '060_1495', '061_1496', '062_1497', '063_1498', '064_1499', '065_1500', '066_1501', '067_1502', '068_1503', '069_1504', '070_1505', '071_1506', '072_1507', '073_1508', '074_1509', '075_1510', '076_1511', '077_1512', '078_1513', '079_1514', '080_1515', '081_1516', '082_1517', '083_1518', '084_1519', '085_1520', '086_1521', '087_1522', '088_1523', '089_1524', '090_1525', '091_1526', '092_1527', '093_1528', '094_1529', '095_1530', '096_1531', '097_1532', '098_1533', '099_1534', '100_1535', '101_1536', '102_1537', '103_1538', '104_1539', '105_1540', '106_1541', '107_1542', '108_1543', '109_1544', '110_1545', '111_1546', '112_1547', '113_1548', '114_1549', '115_1550', '116_1551', '117_1552', '118_1553']
1	**2**	4	53	['069_1554', '070_1555', '071_1556', '086_1557', '087_1558', '088_1559', '089_1560', '090_1561', '091_1562', '092_1563', '093_1564', '094_1565', '095_1566', '096_1567', '097_1568', '098_1569', '101_1570', '103_1571', '104_1572', '105_1573', '106_1574', '107_1575', '108_1576', '109_1577', '110_1578', '111_1579', '112_1580', '113_1581', '114_1582', '115_1583', '116_1584', '117_1585', '118_1586', '119_1587', '120_1588', '121_1589', '122_1590', '123_1591', '124_1592', '125_1593', '126_1594', '127_1595', '128_1596', '129_1597', '130_1598', '131_1599', '132_1600', '133_1601', '134_1602', '137_1603', '139_1604', '154_1605', '155_1606']
2	**2**	43	99	['058_1607', '059_1608', '060_1609', '061_1610', '062_1611', '063_1612', '064_1613', '065_1614', '066_1615', '067_1616', '068_1617', '069_1618', '070_1619', '071_1620', '072_1621', '073_1622', '074_1623', '075_1624', '076_1625', '077_1626', '078_1627', '079_1628', '080_1629', '081_1630', '082_1631', '083_1632', '084_1633', '085_1634', '086_1635', '087_1636', '088_1637', '089_1638', '090_1639', '091_1640', '092_1641', '093_1642', '094_1643', '095_1644', '096_1645', '097_1646', '098_1647', '099_1648', '100_1649', '101_1650', '102_1651', '103_1652', '104_1653', '105_1654', '106_1655', '107_1656', '108_1657',

				'109_1658', '110_1659', '111_1660', '112_1661', '113_1662', '114_1663', '115_1664', '116_1665', '117_1666', '118_1667', '119_1668', '120_1669', '121_1670', '122_1671', '123_1672', '124_1673', '125_1674', '126_1675', '127_1676', '128_1677', '129_1678', '130_1679', '131_1680', '132_1681', '133_1682', '134_1683', '135_1684', '136_1685', '137_1686', '138_1687', '139_1688', '140_1689', '141_1690', '142_1691', '143_1692', '144_1693', '145_1694', '146_1695', '147_1696', '148_1697', '149_1698', '150_1699', '151_1700', '152_1701', '153_1702', '154_1703', '155_1704', '156_1705']
3	**2**	44	71	['077_1706', '078_1707', '079_1708', '080_1709', '081_1710', '082_1711', '083_1712', '084_1713', '085_1714', '086_1715', '087_1716', '088_1717', '089_1718', '090_1719', '091_1720', '092_1721', '093_1722', '094_1723', '095_1724', '096_1725', '097_1726', '098_1727', '099_1728', '100_1729', '101_1730', '102_1731', '103_1732', '104_1733', '105_1734', '106_1735', '107_1736', '108_1737', '109_1738', '110_1739', '111_1740', '112_1741', '113_1742', '114_1743', '115_1744', '116_1745', '117_1746', '118_1747', '119_1748', '120_1749', '121_1750', '122_1751', '123_1752', '124_1753', '125_1754', '126_1755', '127_1756', '128_1757', '129_1758', '130_1759', '131_1760', '132_1761', '133_1762', '134_1763', '135_1764', '136_1765', '137_1766', '138_1767', '139_1768', '140_1769', '141_1770', '142_1771', '143_1772', '144_1773', '145_1774', '146_1775', '147_1776']
4	**2**	46	56	['074_1777', '075_1778', '076_1779', '077_1780', '078_1781', '079_1782', '080_1783', '081_1784', '082_1785', '083_1786', '084_1787', '085_1788', '086_1789', '087_1790', '088_1791', '089_1792', '090_1793', '091_1794', '092_1795', '093_1796', '094_1797', '095_1798', '096_1799', '097_1800', '098_1801', '099_1802', '100_1803', '101_1804', '102_1805', '103_1806', '104_1807', '105_1808', '106_1809', '107_1810', '108_1811', '109_1812', '110_1813', '111_1814', '112_1815', '113_1816', '114_1817', '115_1818', '116_1819', '117_1820', '118_1821', '119_1822', '120_1823', '121_1824', '122_1825', '123_1826', '124_1827', '125_1828', '126_1829', '127_1830', '128_1831', '129_1832']
5	**2**	5	47	['072_1833', '073_1834', '074_1835', '075_1836', '076_1837', '077_1838', '078_1839', '079_1840', '080_1841', '081_1842', '082_1843', '083_1844', '084_1845', '085_1846', '086_1847', '087_1848', '088_1849', '089_1850', '090_1851', '091_1852', '092_1853', '110_1854', '111_1855', '112_1856', '113_1857', '114_1858', '115_1859', '116_1860', '117_1861', '118_1862', '119_1863', '120_1864', '121_1865', '122_1866', '123_1867', '124_1868', '125_1869', '126_1870', '127_1871', '128_1872', '129_1873', '130_1874', '131_1875', '132_1876', '133_1877', '134_1878', '135_1879']

6	**2**	50	41	['109_1880', '110_1881', '111_1882', '112_1883', '113_1884', '114_1885', '115_1886', '116_1887', '117_1888', '118_1889', '119_1890', '120_1891', '121_1892', '122_1893', '123_1894', '124_1895', '125_1896', '126_1897', '127_1898', '128_1899', '129_1900', '130_1901', '131_1902', '132_1903', '133_1904', '134_1905', '135_1906', '136_1907', '137_1908', '138_1909', '139_1910', '140_1911', '141_1912', '142_1913', '143_1914', '144_1915', '145_1916', '146_1917', '147_1918', '148_1919', '149_1920']
7	**2**	6	57	['011_1921', '012_1922', '013_1923', '014_1924', '015_1925', '016_1926', '017_1927', '018_1928', '019_1929', '020_1930', '021_1931', '022_1932', '023_1933', '024_1934', '025_1935', '026_1936', '027_1937', '028_1938', '029_1939', '030_1940', '031_1941', '032_1942', '033_1943', '034_1944', '035_1945', '036_1946', '037_1947', '038_1948', '056_1949', '057_1950', '058_1951', '059_1952', '060_1953', '061_1954', '062_1955', '063_1956', '064_1957', '065_1958', '066_1959', '067_1960', '068_1961', '069_1962', '070_1963', '071_1964', '072_1965', '073_1966', '074_1967', '075_1968', '076_1969', '077_1970', '078_1971', '079_1972', '080_1973', '081_1974', '082_1975', '083_1976', '084_1977']
8	**2**	7	83	['024_1978', '025_1979', '026_1980', '027_1981', '028_1982', '029_1983', '030_1984', '031_1985', '032_1986', '033_1987', '042_1988', '043_1989', '044_1990', '045_1991', '046_1992', '047_1993', '048_1994', '049_1995', '050_1996', '051_1997', '052_1998', '053_1999', '054_2000', '055_2001', '056_2002', '057_2003', '058_2004', '059_2005', '060_2006', '061_2007', '062_2008', '063_2009', '064_2010', '065_2011', '066_2012', '067_2013', '068_2014', '069_2015', '070_2016', '071_2017', '072_2018', '073_2019', '074_2020', '075_2021', '076_2022', '077_2023', '078_2024', '079_2025', '080_2026', '081_2027', '082_2028', '083_2029', '084_2030', '085_2031', '086_2032', '087_2033', '088_2034', '089_2035', '090_2036', '091_2037', '092_2038', '093_2039', '094_2040', '095_2041', '096_2042', '097_2043', '098_2044', '099_2045', '100_2046', '101_2047', '102_2048', '103_2049', '104_2050', '105_2051', '106_2052', '107_2053', '108_2054', '109_2055', '110_2056', '111_2057', '112_2058', '113_2059', '114_2060']