INFORME FINAL

SEGMENTACIÓN AUTOMÁTICA DE TUMORES CEREBRALES A PARTIR DE IMÁGENES DE RESONANCIA MAGNÉTICA ESTRUCTURAL

Fundamentos de Deep Learning

Isabella Ariza Cuberos, 1.152.470.641, isabella.ariza@udea.edu.co

Luisa María Zapata Saldarriaga, 1.035.235.580,luisa.zapatas@udea.edu.co

CONTEXTO

Los tumores cerebrales representan un desafío médico, ya que su presencia puede generar complicaciones graves para la salud al ejercer presión sobre áreas cerebrales vitales y desencadenar una amplia gama de problemas, dependiendo de su ubicación [1]. Entre estos desafíos se incluyen la resistencia a la terapia, la alteración del suministro sanguíneo que dificulta la entrega efectiva de medicamentos, complicaciones asociadas al edema cerebral, la hipertensión intracraneal, episodios de convulsiones y efectos neurotóxicos [2].

El diagnóstico de tumores cerebrales es fundamental para iniciar un tratamiento efectivo a tiempo. En este sentido, las técnicas de imagen permiten realizar un apoyo diagnóstico no invasivo, siendo la resonancia magnética (RM) una de las más utilizadas. Sin embargo, para realizar un diagnóstico adecuado y planificar un tratamiento, es necesario realizar un proceso de segmentación de los tumores en las imágenes de RM. La segmentación permite una delimitación de las estructuras afectadas, proporcionando información de tamaño, forma y características específicas del tumor, lo que facilita la evaluación de su comportamiento y la planificación de intervenciones terapéuticas personalizadas. No obstante, estos procesos usualmente se realizan de forma manual, por un radiólogo experto, lo que hace que sea una tarea que toma mucho tiempo y es propensa a la variabilidad entre diferentes observadores [3].

Para abordar esta problemática, se han desarrollado métodos semiautomatizados y automatizados de segmentación de estructuras anatómicas o patológicas, que van desde enfoques basados en regiones hasta algoritmos avanzados de aprendizaje automático y técnicas especializadas de aprendizaje profundo que permitan disminuir los tiempos de segmentación conservando al máximo la precisión de esta [4].

ESTRUCTURA DE LOS NOTEBOOKS

- **Análisis exploratorio de los datos y creación modelo**

Paso 1: Carga de librerías y datos

Las principales librerías de trabajo son: numpy, nilean, sklearn y tensorflow.

Clases:

- ✓ **0:** 'NOT tumor',
- ✓ **1:** 'NECROTIC/CORE',
- ✓ **2:** 'EDEMA',
- ✓ **3:** 'ENHANCING'

Paso 2: Exploración de datos

Se realizó una primera visualización de las imágenes y sus máscaras de segmentación para verificar que correspondan correctamente a la delimitación del tumor. Para ello, se utilizaron las librerías Nilearn, Nibabel y Matplotlib. Esta visualización también permitió detectar imágenes con componentes artefactuales que podrían dificultar el entrenamiento del modelo.Adicionalmente, se filtraron los datos excluyendo a aquellos sujetos que no contaran con al menos la clase 3 (Enhancing). Como resultado, se obtuvieron 341 datos válidos para el entrenamiento del modelo.

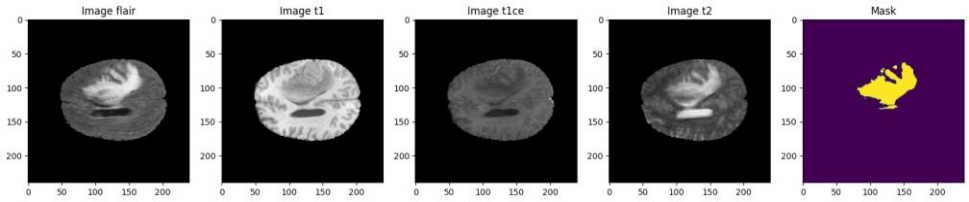


Fig 1. Diferencia de contraste entre las modalidades de imágenes

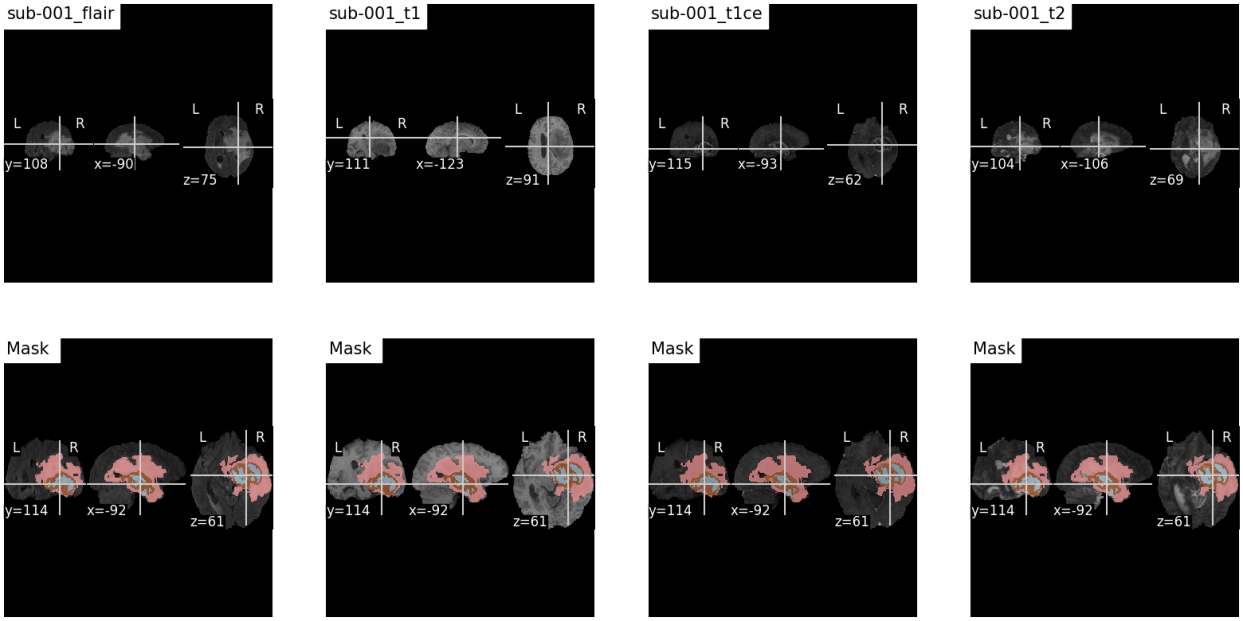


Fig 2. Superposición entre las modalidades de imágenes y la máscara de segmentación

Paso 3: Creación de modelo

En esta sección, vamos a construir y configurar el modelo U-Net para la segmentación de tumores cerebrales en imágenes de resonancia magnética. U-Net es una arquitectura de red neuronal convolucional diseñada específicamente para tareas de segmentación de imágenes médicas. Su estructura única facilita la captura de contextos locales y globales en las imágenes, lo cual es crucial para la identificación precisa de regiones tumorales.

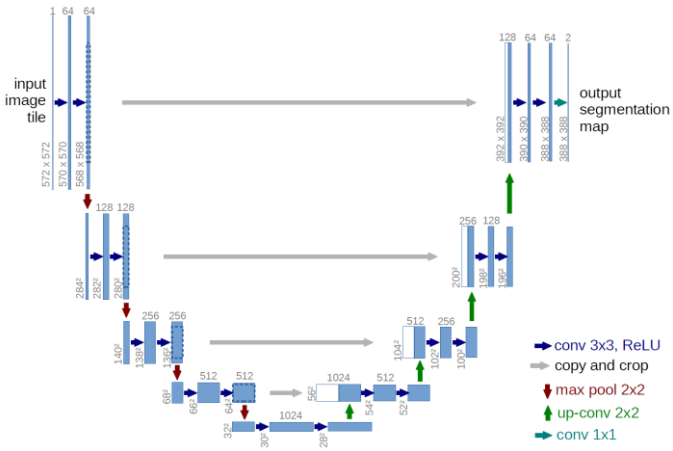


Fig 3. Arquitectura de red neuronal UNET

Paso 4: Preparación de datos para entrenamiento

A - División de datos en 3 sets

Acondicionaremos los datos para que sean adecuados para el entrenamiento de la red neuronal. Dividiremos nuestro conjunto de datos en:

- Training - 75% /80%
- Test - 15% /80%
- Validation - 20%/100%

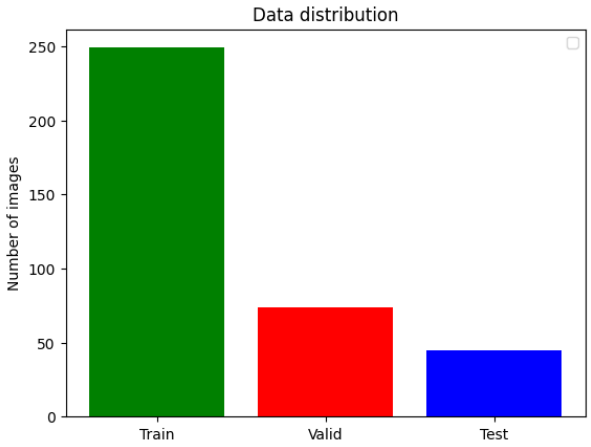


Fig 4. División de los datos

B - Crear un Generador de Datos

Para entrenar una red neuronal en la segmentación de objetos en imágenes, es necesario alimentarla tanto con los datos brutos de la imagen (X) como con las segmentaciones de verdad fundamental (y). Combinando estos dos tipos de datos, la red neuronal puede aprender a reconocer patrones de tumores y hacer predicciones precisas sobre el contenido del escáner de un paciente.

Desafortunadamente, nuestras imágenes de modalidades (X) y nuestras segmentaciones (y) no pueden ser enviadas directamente al modelo de IA. Cargar todas estas imágenes 3D sobrecargaría la memoria

de nuestro entorno y podría causar la caída del sistema. Esto también llevaría a errores de desajuste de forma. Por tanto, hay que realizar un preprocesamiento de imágenes antes de usarlas, lo que se hace mediante un Generador de Datos, más específicamente, para cada muestra:

- Recuperamos las rutas de todas sus modalidades (FLAIR, T1CE, T1 y T2), ya que cada una de estas proporciona información única y complementaria sobre la anatomía y el contraste de tejidos del cerebro.
- Cargamos todas las modalidades y la segmentación e hicimos variaciones realizando selección de ciertas modalidades.
- Asignamos a todos los 4 en el arreglo de máscara el valor 3 (para corregir el caso faltante explicado anteriormente).

Step 5: Entrenamiento del modelo

A - Callbacks

Los callbacks son funciones que se pueden ejecutar durante el proceso de entrenamiento de un modelo de aprendizaje automático y son esenciales para optimizar este proceso y guardar información relevante.

En nuestro caso, utilizaremos tres callbacks específicos:

- **ReduceLROnPlateau:** Este callback reduce la tasa de aprendizaje cuando una métrica ha dejado de mejorar, en nuestro caso, la pérdida de validación. La tasa de aprendizaje se reduce por un factor de 0.2 después de 2 épocas sin mejoras, y la tasa mínima de aprendizaje se establece en 0.000001. Esta técnica ayuda a refinar el entrenamiento cuando nos acercamos a un óptimo local.
- **ModelCheckpoint**: Guarda los mejores pesos del modelo (el modelo que ha obtenido la menor pérdida de validación durante las diferentes épocas). Guardar un modelo nos permite reutilizarlo más tarde o compartirlo, sin tener que entrenarlo desde cero. Esto permite ahorras tiempo y recursos.
- **CSVLogger**: Añade métricas a un archivo CSV, que en nuestro caso se llama 'training.log'. El parámetro append está establecido en False, lo que significa que el archivo se sobrescribirá si ya existe. Esto nos permite tener un registro detallado del proceso de entrenamiento para su análisis posterior.

El uso de estos callbacks no solo mejora la eficiencia del proceso de entrenamiento, sino que también proporciona herramientas valiosas para monitorear y ajustar el rendimiento del modelo.

B - Entranamiento

Ahora estamos listos para entrenar nuestra red neuronal profunda utilizando el método .fit() en Keras. Pasaremos nuestros 3 callbacks al método para que se ejecuten durante el proceso de entrenamiento, el cual durará 35 épocas.

Step 6: Predicción de segmentación de tumores

Utilizaremos el modelo entrenado para realizar predicciones de segmentación en nuevas imágenes.En este caso usaremos las imagenes del conjunto de test y compararemos con la máscara de segmentación manual.

La función utilizada es predictByPath()

Step 7: Evaluación del modelo

Para la evaluación del modelo, se calculan métricas para verificar la eficiencia de los modelos implementados. El coeficiente de Dice se usa para evaluar la segmentación de imágenes médicas. En este experimento, se ha calculado el coeficiente de Dice para cada clase (necrótica, edema, realzado) y la segmentación predicha. Además, se calculan la precisión, la sensibilidad y la especificidad para evaluar el rendimiento general y la generalidad del modelo. Para cada criterio de evaluación, un puntaje más alto significa un mejor rendimiento de segmentación.

$$DiceCoefficient = \frac{2Intersection(TP)}{Intersection + Union(TP + FP + FN)}$$

$$DiceCoefficientNecrotic = \frac{2IntersectionofNecrotic(TP)}{UnionofNecrotic(TP + FP + FN)}$$

$$DiceCoefficientEdema = \frac{2IntersectionofEdema(TP)}{UnionofEdema(TP + FP + FN)}$$

$$DiceCoefficientEnhancing = \frac{2IntersectionofEnhancing(TP)}{UnionofEnhancing(TP + FP + FN)}$$

Fig 5. Métricas de evaluación

DESCRIPCIÓN DE LA SOLUCIÓN

Datos

El conjunto de datos que se utilizaron fue BraTS2020, proporcionado por los organizadores del desafío BraTS 2020 y contienen exploraciones de resonancia magnética multimodal preoperatorias clínicamente adquiridas de glioblastoma (GBM/HGG) y glioma de bajo grado (LGG), que incluyen (a) volúmenes nativos (T1) y (b) post-contraste ponderados en T1 (T1Gd), (c) ponderados en T2 (T2), y (d) recuperación de inversión atenuada por líquido (FLAIR).

Para descargar el conjunto de datos de entrenamiento:

!wget 'https://www.cbica.upenn.edu/MICCAI_BraTS2020_TrainingData'

Para descargar el conjunto de datos de validación:

!wget 'https://www.cbica.upenn.edu/MICCAI_BraTS2020_ValidationData'

ITERACIONES

En la siguiente tabla, se encuentra descripto los hiperparámetros que fueron variados a lo largo de cinco iteraciones. Los parámetros principales que se variaron fue número de épocas, y batch size.

Hiperparámetros				
Iteración	**1**	**2**	**3**	**4**
Número de épocas	50	35	35	35
Optimizador	Adam	Adam	Adam	Adam
Batch size	1	32	16	1
Función de pérdida	Categorical Cross-Entropy	Categorical Cross-Entropy	Categorical Cross-Entropy	Categorical Cross-Entropy
Callbacks	Early stopping, Model Checkpoint	Early stopping	Early stopping	Early stopping
Tipo de datos	T1ce y FLAIR	T1	T1ce y FLAIR	T1

Por otro lado, se almacena por cada iteración las variaciones entre los modelos, con la siguiente nomenclatura, en la carpeta alojada en el repositorio (*ResultadosModelos*).

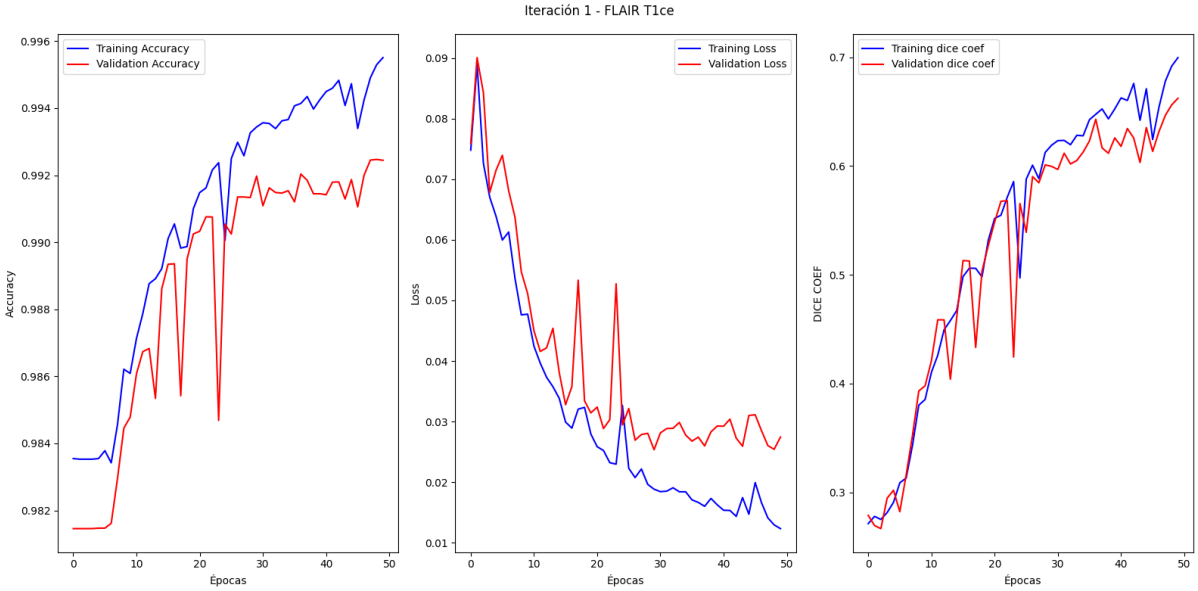
Iteración	Modelo H5
1	model_x1_2_50_1_p(1).h5
2	model_t1_35_32.h5
3	model_x1_2_35_16_p(1).h5
4	model_t1_35_1.h5

RESULTADOS

ITERACIÓN 1

Modelo para 2 tipo de imágenes **T1** y **FLAIR**

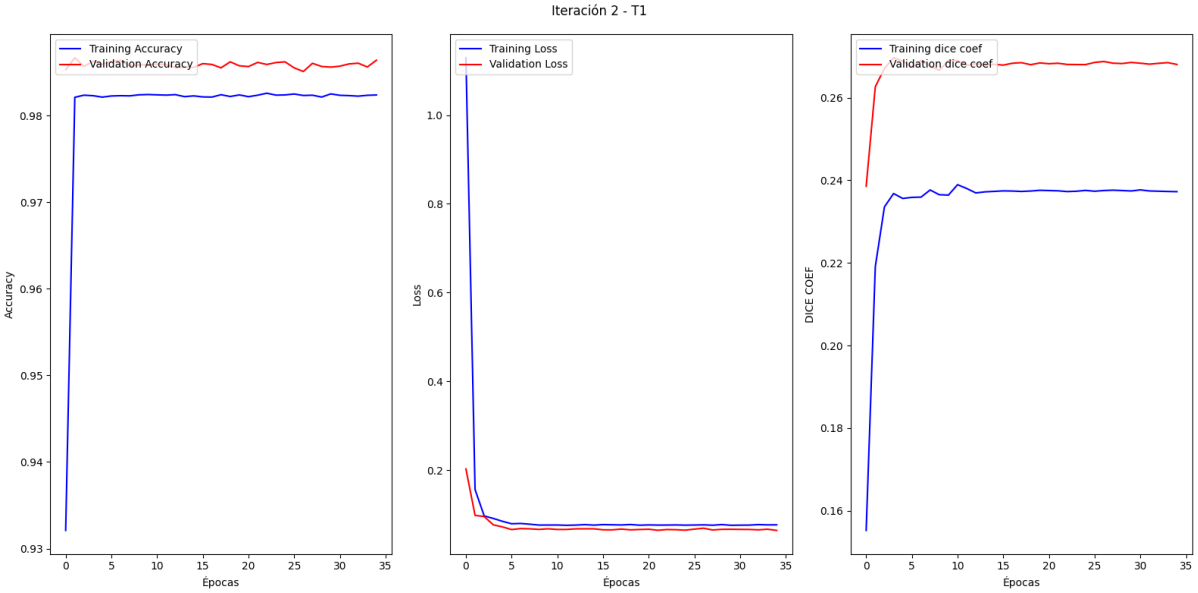
Métrica	Entrenamiento	Validación
Loss	0.0123	0.0274
Accuracy	0.9955	0.9924
Mean IoU	0.8437	0.8416
Precisión	0.9955	0.9925
Sensibilidad	0.9943	0.9914
Especificidad	0.9985	0.9975
Dice coef	**0.6998**	**0.6624**
Dice coef Necrótico	0.7024	0.6066
Dice coef Edema	0.8244	0.7641
Dice coef Realzado	0.7572	0.6935



ITERACIÓN 2

Modelo para 1 tipos de imágenes – **T1**

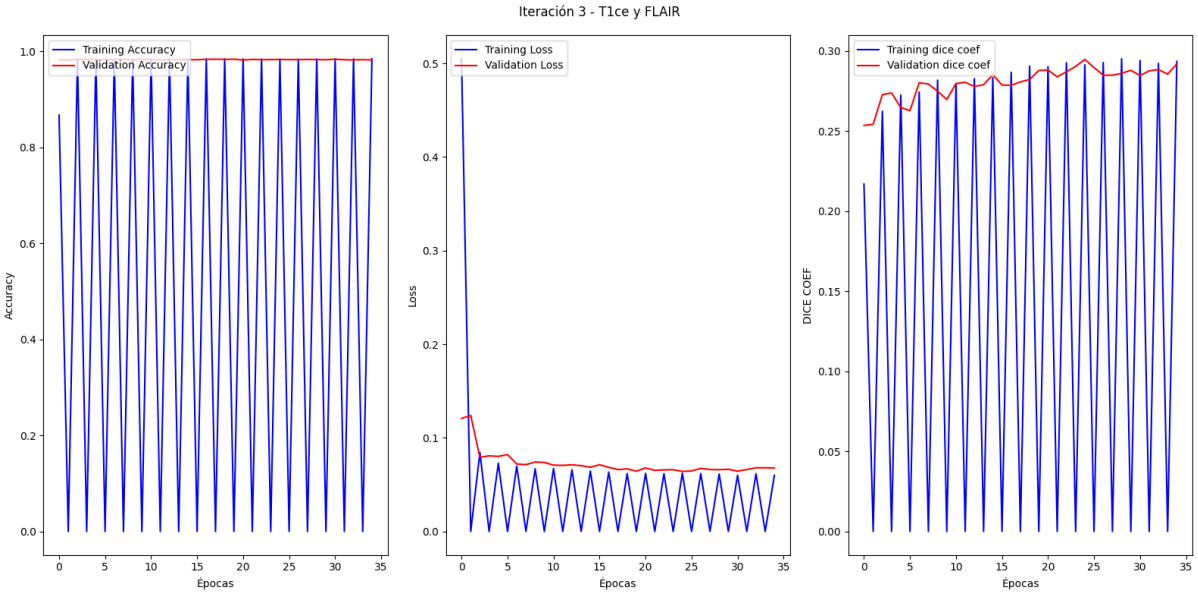
Métrica	Entrenamiento	Validación
Loss	0.0764	0.0638
Accuracy	0.9824	0.9863
Precisión	0.8594	0.9862
Sensibilidad	0.8595	0.9863
Especificidad	0.8698	0.9954
Dice coef	0.2372	0.2680
Dice coef Necrótico	0.0386	0.0428
Dice coef Edema	0.0984	0.1136
Dice coef Realzado	0.0492	0.0468



ITERACIÓN 3

Modelo para 2 tipos de imágenes – **T1ce y FLAIR**

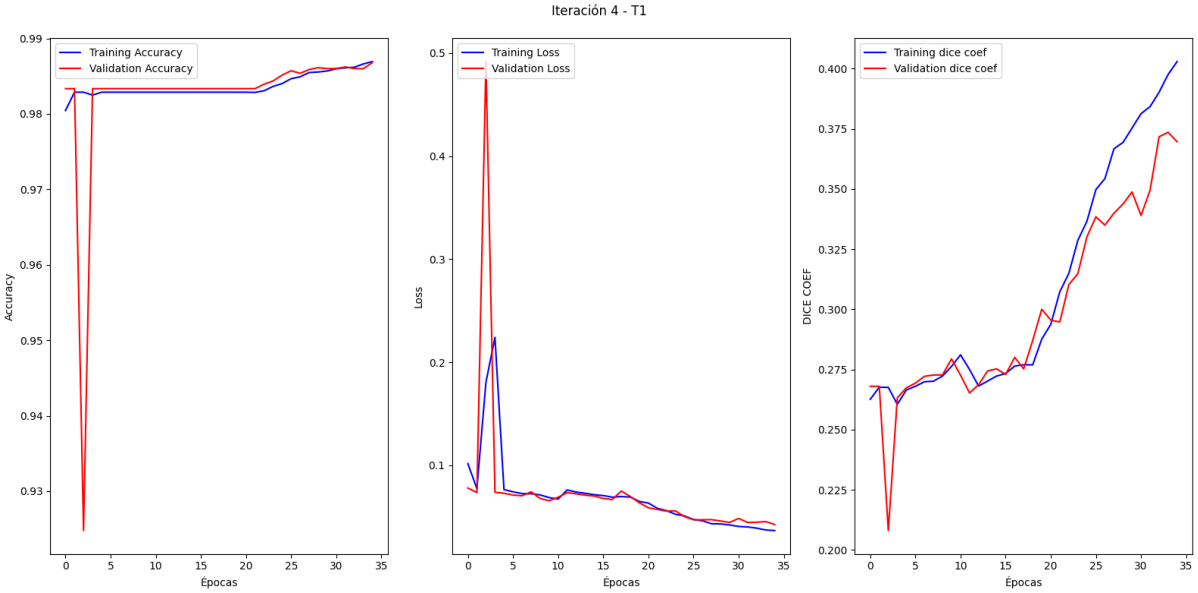
Métrica	Entrenamiento	Validación
Loss	0.036517	0.042421
Accuracy	0.986931	0.986807
Precisión	0.992651	0.991225
Sensibilidad	0.981758	0.982162
Especificidad	0.997402	0.996987
Dice coef	0.402946	0.369624
Dice coef Necrótico	0.367140	0.243598
Dice coef Edema	0.445417	0.399772
Dice coef Realzado	0.240097	0.179526



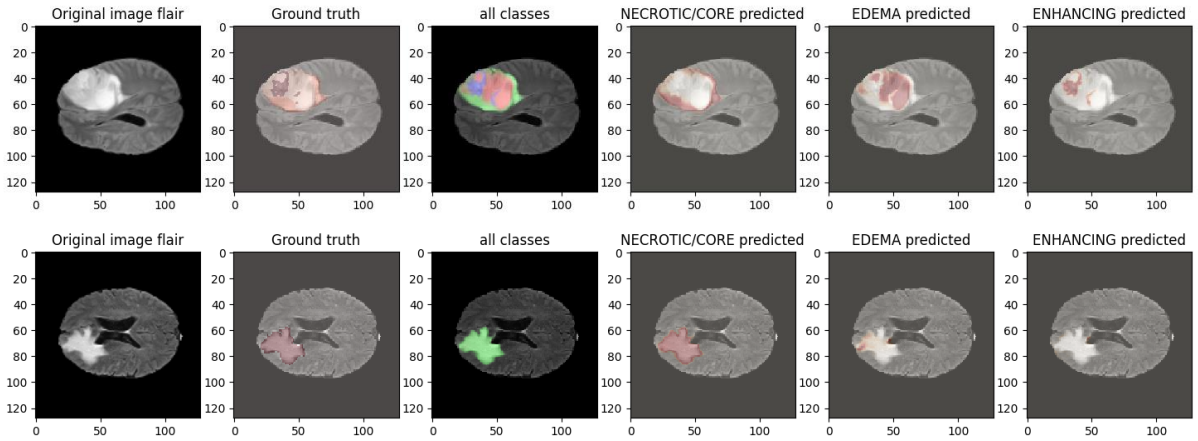
ITERACIÓN 4

Modelo para 1 tipo de imágenes – **T1**

Métrica	Entrenamiento	Validación
Loss	0.036517	0.042421
Accuracy	0.986931	0.986807
Precisión	0.992651	0.991225
Sensibilidad	0.981758	0.982162
Especificidad	0.997402	0.996987
Dice coef	0.402946	0.369624
Dice coef Necrótico	0.367140	0.243598
Dice coef Edema	0.445417	0.399772
Dice coef Realzado	0.240097	0.179526



A continuación, se presentan los resultados de la evaluación del modelo de la iteración 1, el cual presentó los mejores resultados para diferentes sujetos:



Los resultados a lo largo de las diferentes iteraciones presentan métricas de desempeño muy altas. Sin embargo, para la segmentación de tumores cerebrales, es especialmente relevante el coeficiente de DICE, que mide la similitud entre las máscaras predichas y las máscaras reales. Aunque la precisión (accuracy) de los modelos superaba el 90%, no se obtuvieron buenos resultados en el coeficiente de DICE, lo cual es crucial para una segmentación precisa en este contexto.

Para la tarea de segmentación, utilizamos el modelo U-Net, una arquitectura de red neuronal convolucional diseñada específicamente para la segmentación de imágenes médicas. U-Net es conocida por su capacidad para capturar contextos locales y globales en las imágenes, esencial para identificar con precisión las regiones tumorales.

Una de las principales limitaciones durante el entrenamiento de estos datos fue el consumo significativo de recursos computacionales. Inicialmente, utilizamos Google Colab con un tamaño de lote (batch size) de 1, debido a las restricciones de memoria. No obstante, para

experimentar con tamaños de lote mayores, como 16 o 32, se recurrió a un servidor con más recursos computacionales. Esto permitió obtener resultados derivados de estos hiperparámetros, pero también incrementó considerablemente el tiempo de entrenamiento.

El tiempo de entrenamiento de los modelos con tamaños de lote mayores oscilaba entre 4 y 10 horas. Este aumento en el tiempo de procesamiento se debió a la mayor cantidad de datos que el modelo debía procesar en cada iteración y a la necesidad de ajustar los parámetros del modelo de manera más precisa.