UNIVERSIDAD PERUANA
CAYETANO HEREDIA

Optimized U-Net for Brain Tumor Segmentation

Aplicaciones Clínicas en Señales e Imágenes

Grupo 1

Semestre 2024-2

Contenido

Introducción

Gliomas: Son un tipo de tumor cerebral que se origina en las células gliales, que son las células de soporte del sistema nervioso central (SNC)

- Representan aproximadamente el 1-2% de todos los cánceres diagnosticados en el país, según el Instituto Nacional de Enfermedades Neoplásicas (INEN)

- Se estima que entre 500 y 700 nuevos casos se detectan cada año en Perú, siendo más comunes en adultos, pero también presentes en niños.

- Las tasas de supervivencia a 5 años para este tipo de tumores en Perú están por debajo del 10%.

Desafios en la interpretacion de las imagenes

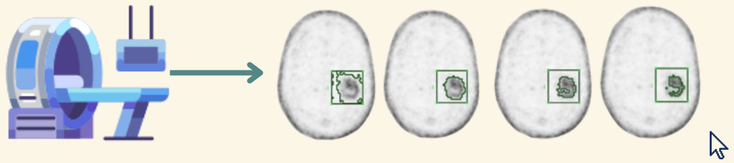
- Difícil Diferenciación entre Tumor y Edema
- Distinción entre Recurrencia Tumoral y Efectos Postratamiento
- Variabilidad entre Modalidades de Imágenes
- Limitaciones de los Algoritmos Automáticos

Referencias

1. Instituto Nacional de Enfermedades Neoplásicas (INEN). "Registro Nacional de Cáncer 2021."
2. GLOBOCAN, International Agency for Research on Cancer (IARC). "Global Cancer Observatory 2020: Brain Cancer Statistics."
3. Organización Panamericana de la Salud (OPS). "Situación del cáncer en América Latina 2022."
4. Central Brain Tumor Registry of the United States (CBTRUS). "Statistical Report on Primary Brain and Central Nervous System Tumors Diagnosed in the United States 2020."

Problemática

La segmentación tumoral...



- Identificar
- Delinear regiones afectadas por un tumor

La segmentación automática

- Proporciona asistencia a los radiólogos
- Enfoque más preciso y fiable
- Permite planificación de tratamiento y seguimiento con más tiempo

Desafío en diseñar una arquitectura de red neuronal óptima y un programa de entrenamiento adecuado

Segmentación manual

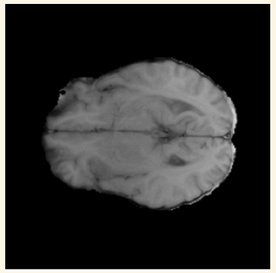
Realizada por radiólogos con experiencia

- Proceso que consume mucho tiempo
- Carencia de consistencia y reproducibilidad
- Sujeto a errores humanos

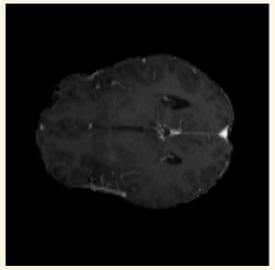
Base de Datos

Se utiliza la base de datos proporcionada por el desafío BraTS21 (Brain Tumor Segmentation Challenge 2021)

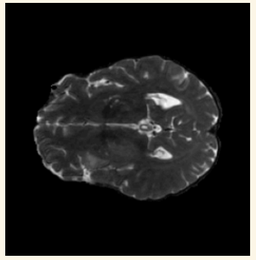
Tipos de imágenes: Imágenes de MRI



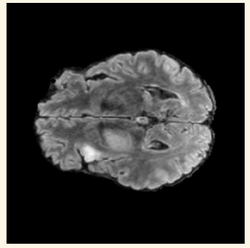
T1: Proporcionan un contraste claro entre la materia gris y blanca



T1Gd: Realzan las áreas con vascularidad de los tumores



T2: Destacan el líquido cerebroespinal y permiten la visualización de edema



FLAIR: Suprime el líquido cerebroespinal para resaltar las lesiones cerebrales

Característica de las imágenes: Almacenadas en *formato NIfTI (.nii)*. Este formato permite manejar volúmenes de datos 3D, preservando la información espacial necesaria para la segmentación precisa.

Características del Conjunto de Datos Resolución:

- Preprocesadas con resolución isotrópica de 1mm3, con dimensiones de 240x240x155 voxeles.
- Etiquetas de Segmentación: Las etiquetas incluyen cuatro clases: tumor realzado (ET), tejido edematoso peritumoral (ED), núcleo necrótico del tumor (NCR), y fondo (voxeles que no son parte del tumor).

Anotaciones

Las anotaciones de los tumores han sido realizadas manualmente **por entre uno y cuatro expertos**, lo que asegura una alta calidad en las etiquetas de segmentación.

Conjunto de datos

Entrenamiento: 1,251 casos
Validación: 219 casos
Prueba 570 casos

Adaptaciones en la base de Datos

Conjunto de datos

Para en el entrenamiento utilizamos **32 casos**

En el código

```
label_slice = F.one_hot(label_slice, num_classes=4).permute(2, 0, 1).float()[:3]
```

Los **targets** se preprocesan y expanden a múltiples canales, donde cada canal corresponde a una clase. Esto es importante porque el modelo debe aprender a predecir múltiples clases de tejidos en lugar de solo una.

Targets

- 0 - Fondo
- 1 - Región del edema peritumoral
- 2 - Tejido necrótico del núcleo tumoral
- 3 - Componente realzado del tumor (Enhancing tumor)

En la clase personalizada de pérdida

```
def forward(self, p, y):
    y_wt, y_tc, y_et = y > 0, ((y == 1) + (y == 3)) > 0, y == 3  # Aquí se definen los
    p_wt, p_tc, p_et = p[:, 0].unsqueeze(1), p[:, 1].unsqueeze(1), p[:, 2].unsqueeze(1)
    return self._loss(p_wt, y_wt) + self._loss(p_tc, y_tc) + self._loss(p_et, y_et)
```

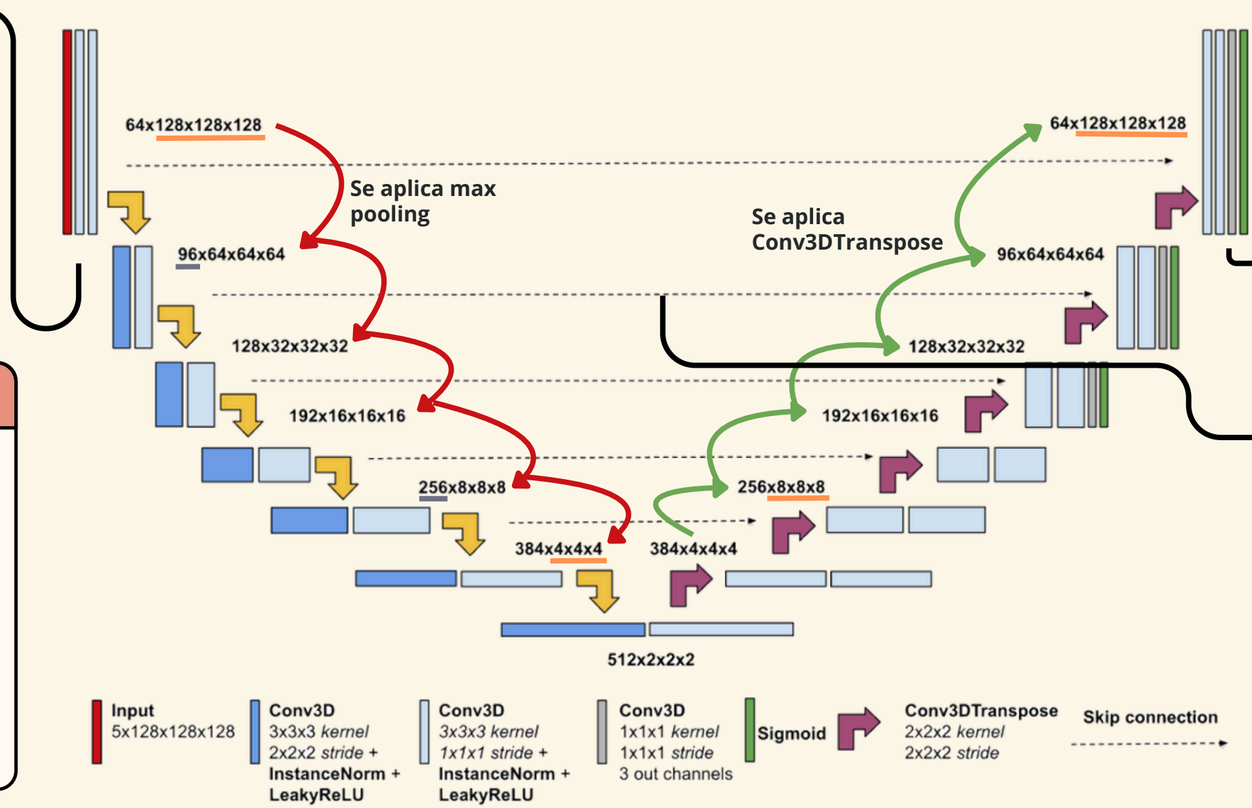
- y_wt (whole tumor): Todos los valores mayores a 0 (clases 1, 2, 3).
- y_tc (tumor core): Núcleo del tumor (clases 1 y 3).
- y_et (enhancing tumor): Tumor realzado (clase 3)

Algoritmos utilizados

Se utilizó la red de U-NET Red convolucional diseñada específicamente para segmentación de imágenes médicas. Se caracteriza por tener una forma de “U”. Es muy eficiente al momento de capturar información global como detalles locales en las imágenes.

Encoder

Reduce las dimensiones espaciales de la imagen mientras **aumenta la profundidad de las características**, utilizando convoluciones y capas de pooling.



Decoder

Aumenta las dimensiones espaciales para recuperar la resolución original y realiza la segmentación. Utilizando convoluciones 3D transpuestas.

Skip Conections

Estas conexiones directas entre capas del encoder y del decoder permiten la recuperación de detalles finos y ayudan a mejorar la precisión de la segmentación.

Pequeñas definiciones

- **Convoluciones:** Es un filtro que aumenta ciertas características de la imagen original.
- **Pooling:** Es una técnica para reducir la imagen sin perder las características de estas.
- **Convoluciones 3D transpuestas:** Aumenta la resolución espacial de la entrada al aplicar filtros de forma "invertida"

Algoritmos utilizados

En nuestro proyecto

Librerias utilizadas

- Pytorch
  - torch.nn
  - torch.optim
  - torch.utils.data
- MONAI
  - DiceLoss
- Numpy
- Matplotlib
- Nibabel
- glob
- Joblib

Se utilizó la red de U-NET

```
class SimpleSegmentationModel(nn.Module):
    def __init__(self):
        super(SimpleSegmentationModel, self).__init__()
        self.encoder = nn.Conv3d(4, 8, kernel_size=3, padding=1)
        self.decoder = nn.ConvTranspose3d(8, 3, kernel_size=2, stride=2)
```

La estructura básica de encoder-decoder de U-Net está presente, aunque es una versión mucho más simplificada en términos de profundidad de capas y detalles arquitectónicos.

Se utilizó Pérdida combinada (Dice Loss + BCE)

```
class Loss(nn.Module):
    def __init__(self):
        super(Loss, self).__init__()
        self.dice_loss = DiceLoss(sigmoid=True, batch=True)
        self.ce_loss = nn.BCEWithLogitsLoss()

    def forward(self, p, y):
        dice_loss = self.dice_loss(p, y)
        ce_loss = self.ce_loss(p, y.float())
        return dice_loss + ce_loss
```

Dice Loss

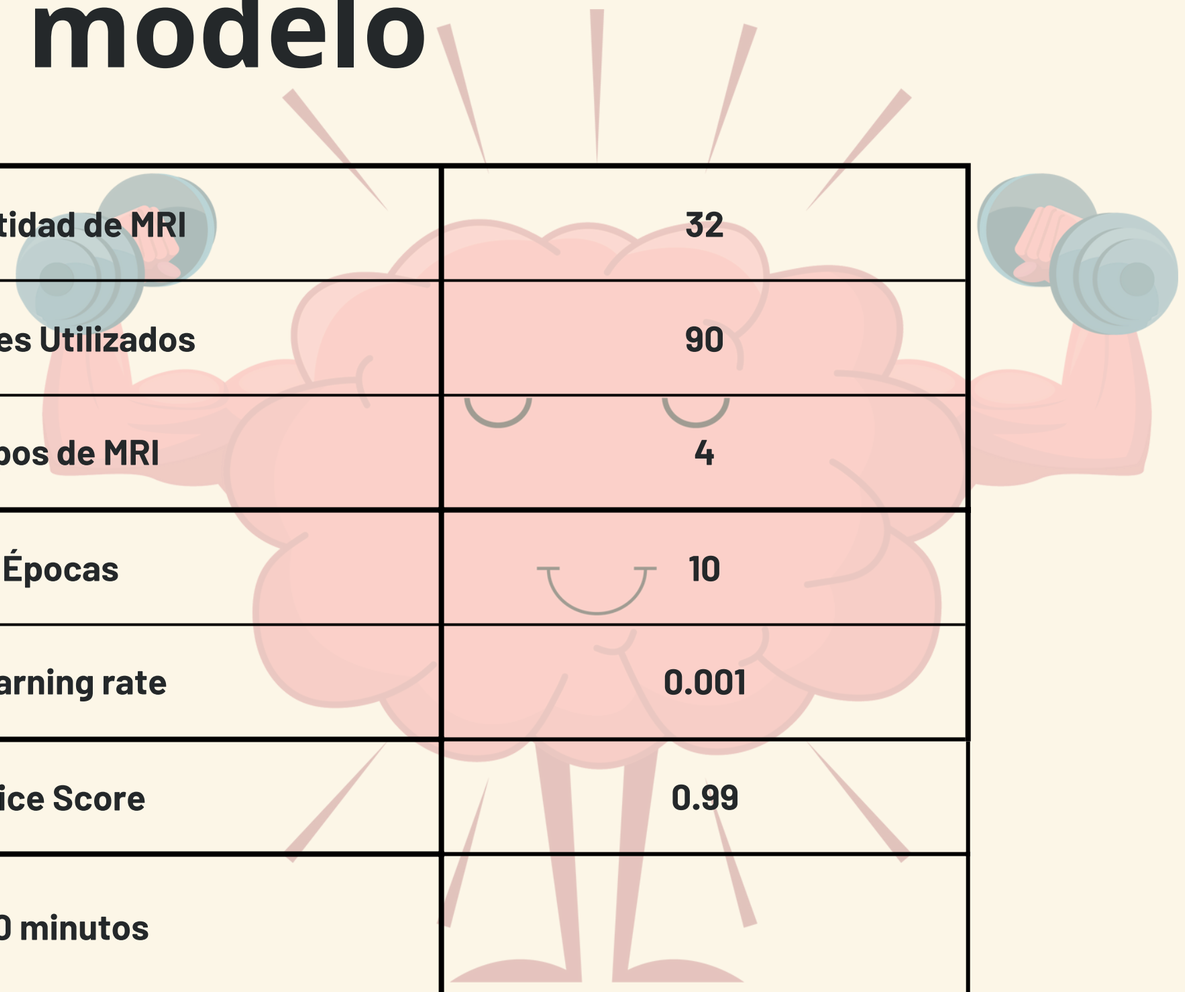
Para medir la superposición entre las predicciones y las etiquetas. Es especialmente útil en casos donde las clases están desbalanceadas.

Binary Cross Entropy

Se utiliza para problemas de clasificación binaria (en este caso, presencia o ausencia de una determinada clase). Es útil para estabilizar el aprendizaje del modelo y mejorar la convergencia.

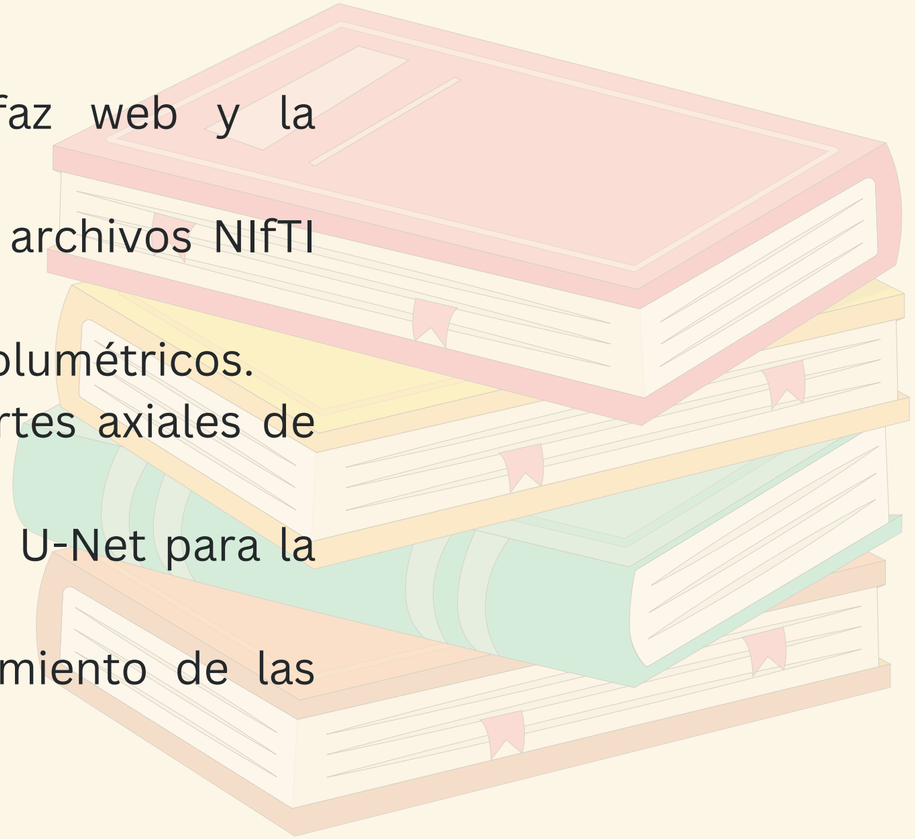
Entrenamiento del modelo

Datos	Cantidad de MRI	32
	Cortes Utilizados	90
	Tipos de MRI	4
Parámetros	Épocas	10
	Learning rate	0.001
Métrica	Dice Score	0.99
Tiempo Aproximado	20 minutos	



Librerías utilizadas en la código Streamlit

- **Streamlit:** Para la creación de la interfaz web y la visualización interactiva de imágenes.
- **Nibabel:** Para la carga y procesamiento de archivos NIfTI (.nii/.nii.gz).
- **NumPy:** Manejo y procesamiento de datos volumétricos.
- **Matplotlib:** Visualización de imágenes y cortes axiales de las modalidades MRI.
- **PyTorch:** Implementación y uso del modelo U-Net para la segmentación.
- **Scipy:** Para la interpolación y preprocesamiento de las imágenes



Integracion del modelo a Streamlit

Creación de archivo .pth

```
torch.save(model.state_dict(), "modelo_entrenado.pth")
```

“51 MB”

Datos de ruta en google drive

```
MODEL_ID = '1r5EWxoBiCMF7ug6jly-3Oma4C9N4ZhGi'
MODEL_PATH = 'modelo_entrenado.pth' # Reemplaza
```

Carga de modelo

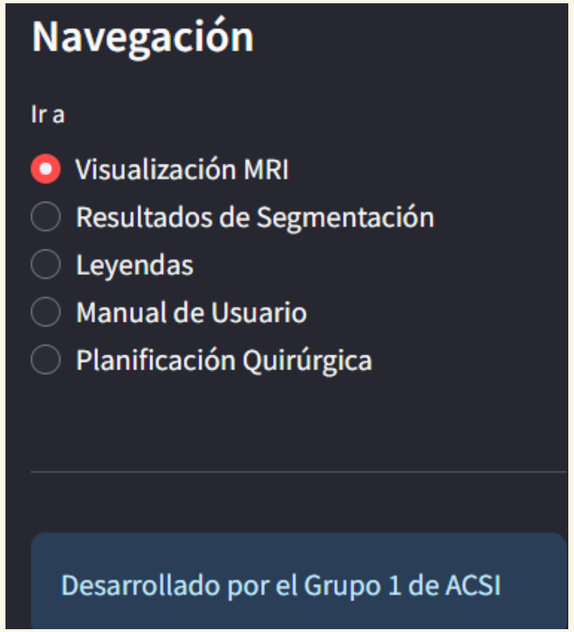
```
state_dict = torch.load(MODEL_PATH, map_location=torch.device("cpu"))
modelo.load_state_dict(state_dict)
modelo.eval()
st.success("Modelo cargado correctamente.")
return modelo
```

Resultados

Modelo U-Net

```
Epoch 1/10, Loss: 1.3347, Dice Score: 0.7752
Epoch 2/10, Loss: 1.1509, Dice Score: 0.9736
Epoch 3/10, Loss: 1.0642, Dice Score: 0.9871
Epoch 4/10, Loss: 1.0059, Dice Score: 0.9858
Epoch 5/10, Loss: 0.9567, Dice Score: 0.9863
Epoch 6/10, Loss: 0.9159, Dice Score: 0.9861
Epoch 7/10, Loss: 0.8781, Dice Score: 0.9928
Epoch 8/10, Loss: 0.8495, Dice Score: 0.9909
Epoch 9/10, Loss: 0.8295, Dice Score: 0.9858
Epoch 10/10, Loss: 0.8000, Dice Score: 0.9945
```

Interfaz



Link de StreamLite

https://mriplaneacionapp.streamlit.app

Resultados: Streamlit

Funciones de la aplicación

Visualización

- Cargar los 4 tipos de MRI en .nii
- Exploración de cortes

Segmentación

- Cargar imágenes apiladas de MRI en .nii
- Segmentación del tumor por corte (exploración)

Leyenda

- Muestra información sobre la segmentación del tumor

Planificación quirúrgica

- Muestra procedimientos y recomendaciones de la planificación quirúrgica

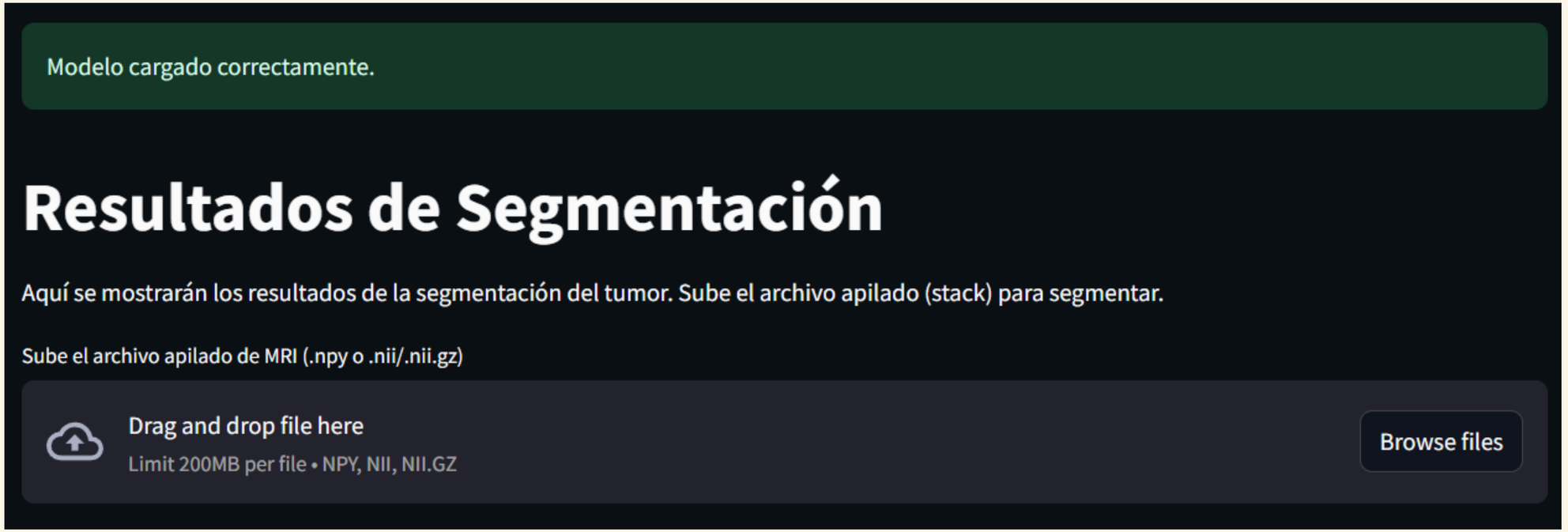
“Manual de uso”

Resultados

Interfaz

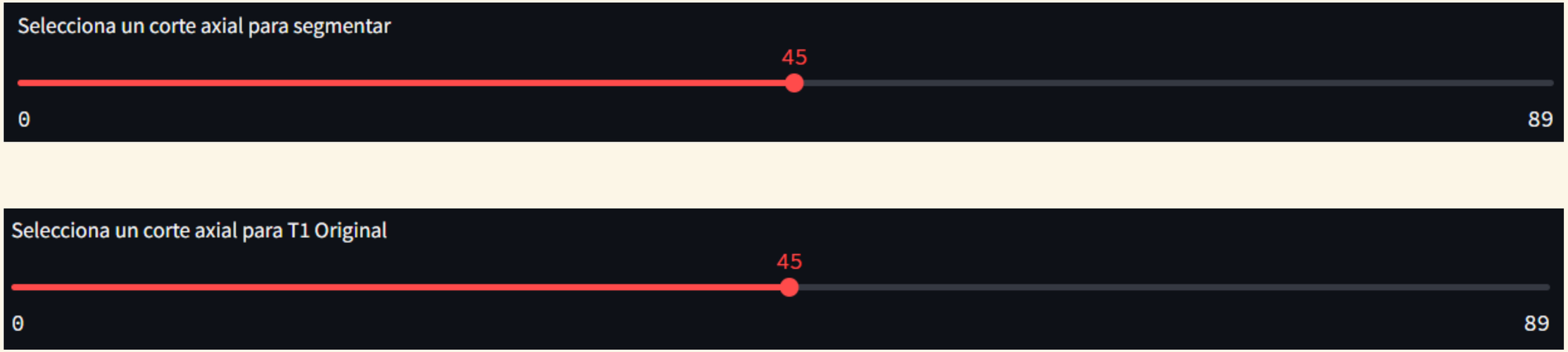
Resultados

Interfaz



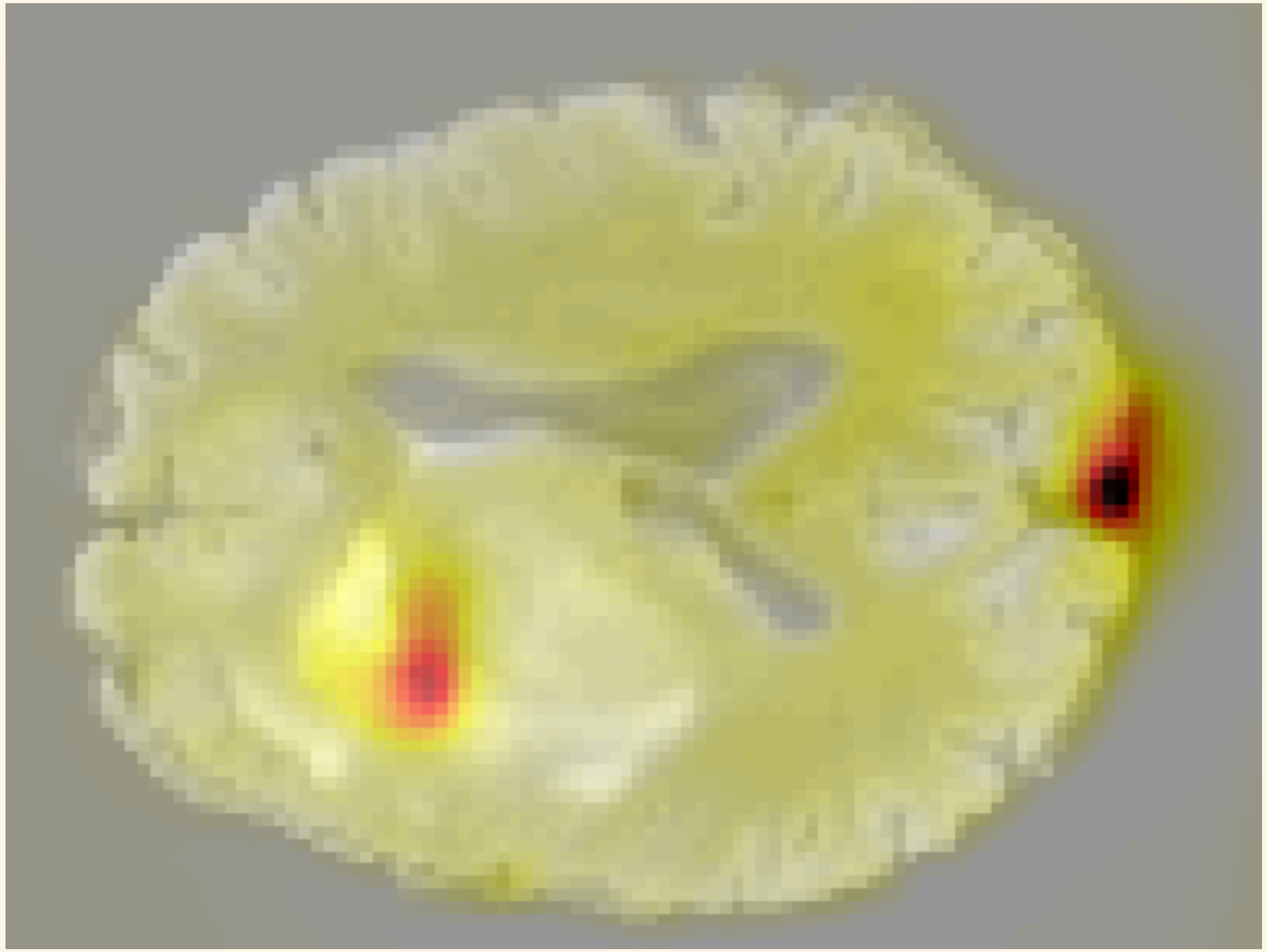
Resultados

Segmentación

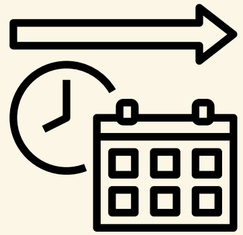


Resultados

Segmentación



Trabajo futuro



- Aumentar datos de entrenamiento
- Reducir la presencia de overfitting
- Mejoras en la interfaz
- Añadir nuevas funciones
- Solo se suban los MRI.nii.gz

GRACIAS