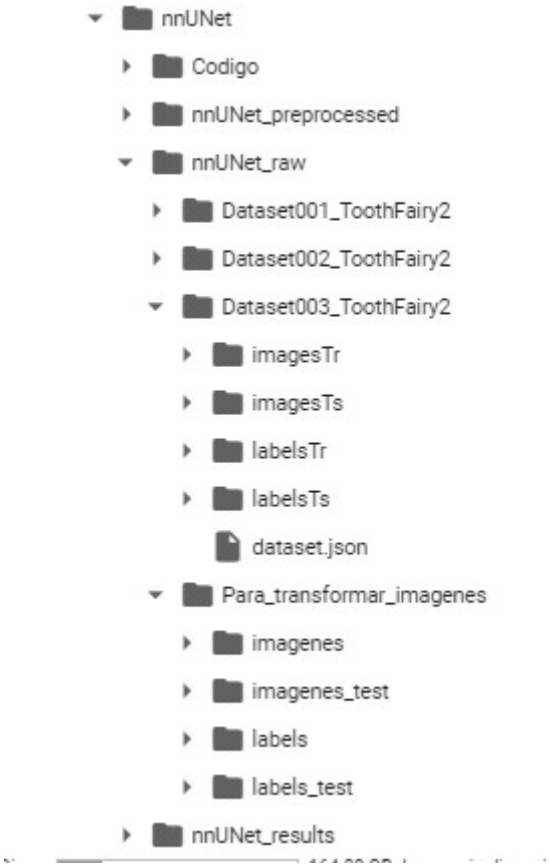
Anexo 3

Pasos para entrenar nn-Unet

1. Crear estructura de carpetas:



La carpeta “nnUnet” es obligatoria y sirve como la carpeta raíz.

Dentro de esta, la carpeta “nnUnet_raw” también es obligatoria y se utiliza para almacenar los conjuntos de datos que se emplean como entrada para el entrenamiento y las pruebas. Cada conjunto de datos se identifica con un ID de tres dígitos y un nombre (que se puede elegir libremente). Por ejemplo, Dataset003_ToothFairy2 tiene 'ToothFairy2' como nombre de conjunto de datos, con un ID de conjunto de datos 003.

Dentro de “nnUnet_raw”, se encuentran subcarpetas como imagesTr, imagesTs, labelsTr y labelsTs, todas ellas obligatorias, que se utilizan para depositar imágenes y etiquetas de entrenamiento y prueba. En mi caso, estas subcarpetas albergan imágenes transformadas de 3D a 2D utilizando un código de transformación.

La carpeta nnUnet_results también es obligatoria y se emplea para almacenar automáticamente los resultados de entrenamiento por fold y cross-validation para cada conjunto de datos. Estos resultados se guardan en la carpeta correspondiente al "entrenador". En mi caso, esta carpeta se llama nnUNetTrainer_20epochs__nnUNetPlans__2d y se crea automáticamente durante la ejecución del programa. Además, dentro de nnUnet_results, se genera

automáticamente la carpeta inference, donde se guardan los resultados de predicción antes de cualquier procesamiento adicional.

La carpeta nnUnet_preprocessed es obligatoria y se utiliza para almacenar automáticamente las predicciones finales realizadas por el modelo basadas en los datos de la carpeta inference.

Por último, la carpeta Para_transformar_imagenes es opcional y no se utiliza por nnunet. Esta carpeta fue creada para depositar imágenes en formato 3D .mha, transformarlas dividiéndolas en 274 slices cada una junto con sus etiquetas, y luego depositar los resultados en la carpeta nnUnet_raw que se utiliza por el modelo.

2. Crear un archivo denominado 'dataset.json' que contiene los metadatos necesarios para el entrenamiento de nnU-Net. Se deben tener en cuenta los siguientes parámetros:

- 'tensorImageSize': Debe indicarse '4D' para imágenes 3D como .mha y '3D' para imágenes 2D como nii.gz.
- 'labels': Se establece que el fondo siempre se etiqueta como 0, mientras que las etiquetas para el resto de clases son números enteros consecutivos o agrupados.
- 'regions_class_order': Este parámetro se añade únicamente si hay agrupaciones de clases.
- 'numTraining': Indicar el número de imágenes en el conjunto de entrenamiento.
- 'file_ending': Especificar el formato de archivo de las imágenes en el conjunto de datos. Además de '.mha' y 'nii.gz', es importante destacar que existen otros formatos compatibles. Para obtener más detalles sobre estos formatos adicionales, se puede consultar la documentación oficial en el siguiente enlace: https://github.com/MIC-DKFZ/nnUNet/blob/master/documentation/dataset_format.md."


```
{
  "name": "ToothFairy 2",
  "description": "Segmentation of maxillofacial CBCT volumes",
  "reference": "https://ditto.ing.unimore.it/toothfairy2/",
  "license": "CC-BY-SA 4.0",
  "release": "20/04/2024",
  "tensorImageSize": "3D",
  "labels": {
    "background": 0,
    "lower jawbone": 1,
    "upper jawbone": 2,
    "teeth": [11, 12, 13, 14, 15, 16, 17, 18, 21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 33, 34, 35, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47,
48],
    "resto": [3, 4, 5, 6, 7, 8, 9, 10, 19, 20, 29, 30, 39, 40]
  },
  "regions_class_order": [1, 2, 3, 4],
  "numTraining": 1620,
  "numTest": 0,
  "file_ending": ".nii.gz",
  "channel_names": {
    "0": "CBCT"
  }
}
```

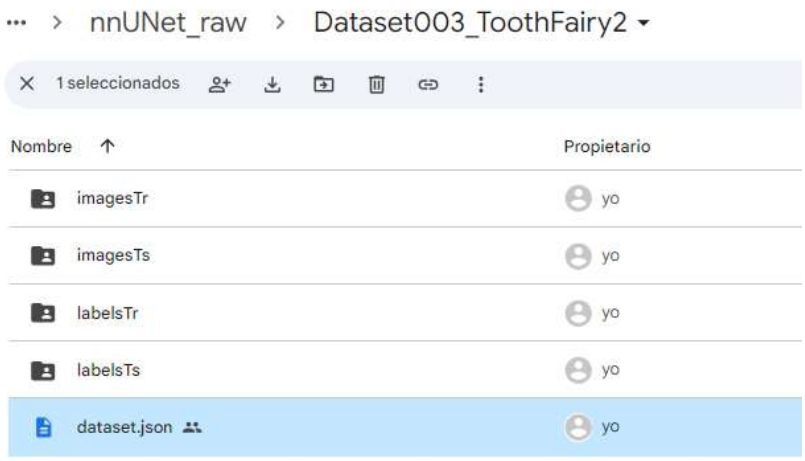

Ilustración 1. Ejemplo de 'dataset.json' para entrada de imágenes 2D".nii.gz" y clases agrupados

```
{
  "name": "ToothFairy 2",
  "description": "Segmentation of maxillofacial CBCT volumes",
  "reference": "https://ditto.ing.unimore.it/toothfairy2/",
  "license": "CC-BY-SA 4.0",
  "release": "20/04/2024",
  "tensorImageSize": "4D",
  "labels": {
    "background": 0,
    "Lower Jawbone": 1,
    "Upper Jawbone": 2,
    "Left Inferior Alveolar Canal": 3,
    "Right Inferior Alveolar Canal": 4,
    "Left Maxillary Sinus": 5,

    ..... .
    "Lower Right Second Molar": 47,
    "Lower Right Third Molar (Wisdom Tooth)": 48
  },
  "numTraining": 45,
  "numTest": 0,
  "file_ending": ".mha",
  "channel_names": {
    "0": "CBCT"
  }
}
```


Ilustración 2. Ejemplo de 'dataset.json' para entrada de imágenes 3D".mha" sin agrupación de clases.

3. Depositar 'dataset.json' en la carpeta 'nnUnet_raw - nombre del conjunto de datos', según se muestra en la imagen adjunta.



4. Depositar las imágenes de entrenamiento y prueba, junto con sus etiquetas, en las carpetas correspondientes según el tipo de entrada. Si se van a utilizar imágenes 3D para el entrenamiento, se puede descargar el conjunto de datos Tooth_Fairy2 y colocarlo directamente en las carpetas 'imagesTr', 'imagesTs', 'labelsTr' y 'labelsTs' dentro de la carpeta 'nnUnet_raw', que serán utilizadas por el modelo. Por otro lado, si se desea utilizar imágenes 2D para el entrenamiento, entonces es necesario depositar imágenes 3D en carpetas dentro de 'Para_transformar_imagenes' y ejecutar el código del Anexo 3 para su transformación y posterior guardado dentro de 'nnUnet_raw'. Se han probado ambas opciones.

5. Ejecutar el código para el entrenamiento y test