Anexo 5

1. Instalaciones necesarias

```
%%capture
!pip install nnunetv2
```

```
import torch
import nibabel as nib
import numpy as np
import pandas as pd
import os
import pathlib
import json
```

```
# Montar Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

2. Preprocesamiento de imagenes

```
import os
os.environ["nnUNet_raw"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_raw"
os.environ["nnUNet_preprocessed"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed"
os.environ["nnUNet_results"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_results"
```

```
!nnUNetv2_plan_and_preprocess -d 002 --verify_dataset_integrity
```

```
Fingerprint extraction...
Dataset002_ToothFairy2
Using <class 'nnunetv2.imageio.simpleitk_reader_writer.SimpleITKIO'> as reader/writer

####################
verify_dataset_integrity Done.
If you didn't see any error messages then your dataset is most likely OK!
####################

Using <class 'nnunetv2.imageio.simpleitk_reader_writer.SimpleITKIO'> as reader/writer
100% 547/547 [00:31<00:00, 17.50it/s]
Experiment planning...

############################
INFO: You are using the old nnU-Net default planner. We have updated our recommendations. Please consider using those instead! Read
############################

2D U-Net configuration:
{'data_identifier': 'nnUNetPlans_2d', 'preprocessor_name': 'DefaultPreprocessor', 'batch_size': 16, 'patch_size': (448, 448), 'medi

Using <class 'nnunetv2.imageio.simpleitk_reader_writer.SimpleITKIO'> as reader/writer
Plans were saved to /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Dataset002_ToothFairy2/nnUNetPlans.json
Preprocessing...
Preprocessing dataset Dataset002_ToothFairy2
Configuration: 2d...
100% 547/547 [00:39<00:00, 13.92it/s]
Configuration: 3d_fullres...
INFO: Configuration 3d_fullres not found in plans file nnUNetPlans.json of dataset Dataset002_ToothFairy2. Skipping.
Configuration: 3d_lowres...
INFO: Configuration 3d_lowres not found in plans file nnUNetPlans.json of dataset Dataset002_ToothFairy2. Skipping.
```

3. Entrenamiento

3.1 Clase de entrenador personalizada

Para solucionar el problema de tener un número predeterminado de 1000 épocas en nnUNet, se crea una nueva clase de entrenador personalizada llamada "nnUNetTrainer_1epoch". Esta clase hereda las propiedades de la clase original, pero sobrescribe el número de épocas, estableciéndolo en 1.

```
from nnunetv2.training.nnUNetTrainer.nnUNetTrainer import nnUNetTrainer

class nnUNetTrainer_1epoch(nnUNetTrainer):
    def __init__(self, plans: dict, configuration: str, fold: int, dataset_json: dict, unpack_dataset: bool = True,
                 device: torch.device = torch.device('cuda')):
        """used for debugging plans etc"""
        super().__init__(plans, configuration, fold, dataset_json, unpack_dataset, device)
        self.num_epochs = 1
```

3.2 Entrenamiento con folds 0-4

En nnUNet no se utiliza un conjunto de validación tradicional. En lugar de eso, se emplea una validación cruzada con 5 folds. Esto significa que el entrenamiento se ejecuta cinco veces, cada vez utilizando un fold diferente como conjunto de validación y los otros cuatro folds como conjunto de entrenamiento. Este enfoque ayuda a evaluar la robustez y la generalización del modelo.

Al ejecutar el entrenamiento, especificamos cada uno de los folds. Además, indicamos el parámetro –npz, que le dice a nnUNet que almacene las salidas softmax durante la validación final. Esto es necesario porque, después del entrenamiento, vamos a pedirle a nnUNet que elija la mejor configuración.

Fold 0

```
# Set up environment variables
os.environ["nnUNet_raw"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_raw"
os.environ["nnUNet_preprocessed"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed"
os.environ["nnUNet_results"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_results"

!nnUNetv2_train Dataset002_ToothFairy2 2d 0 --npz -tr nnUNetTrainer_1epoch
```

```
############################
INFO: You are using the old nnU-Net default plans. We have updated our recommendations. Please consider using those instead! Read
############################

Using device: cuda:0

#######################################################################
Please cite the following paper when using nnU-Net:
Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep le
#######################################################################

2024-05-28 15:02:49.677143: do_dummy_2d_data_aug: False
2024-05-28 15:02:49.707032: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:02:49.711857: The split file contains 5 splits.
2024-05-28 15:02:49.713997: Desired fold for training: 0
2024-05-28 15:02:49.727013: This split has 437 training and 110 validation cases.
using pin_memory on device 0
using pin_memory on device 0
2024-05-28 15:03:05.180514: Using torch.compile...
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:28: UserWarning: The verbose parameter is deprecated. Please
  warnings.warn("The verbose parameter is deprecated. Please use get_last_lr() "

This is the configuration used by this training:
Configuration name: 2d
 {'data_identifier': 'nnUNetPlans_2d', 'preprocessor_name': 'DefaultPreprocessor', 'batch_size': 16, 'patch_size': [448, 448], 'm

These are the global plan.json settings:
 {'dataset_name': 'Dataset002_ToothFairy2', 'plans_name': 'nnUNetPlans', 'original_median_spacing_after_transp': [999.0, 1.0, 1.0

2024-05-28 15:03:07.968251: unpacking dataset...
2024-05-28 15:03:22.434384: unpacking done...
2024-05-28 15:03:22.452129: Unable to plot network architecture: nnUNet_compile is enabled!
2024-05-28 15:03:22.463289:
2024-05-28 15:03:22.480005: Epoch 0
2024-05-28 15:03:22.482608: Current learning rate: 0.01
2024-05-28 15:07:55.238821: train_loss -0.0573
2024-05-28 15:07:55.244973: val_loss -0.3553
2024-05-28 15:07:55.249099: Pseudo dice [0.5998, 0.2457, 0.6148, 0.0]
2024-05-28 15:07:55.254471: Epoch time: 272.78 s
2024-05-28 15:07:55.258180: Yayy! New best EMA pseudo Dice: 0.3651
2024-05-28 15:07:59.099033: Training done.
2024-05-28 15:07:59.977786: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:08:00.004975: The split file contains 5 splits.
2024-05-28 15:08:00.008453: Desired fold for training: 0
2024-05-28 15:08:00.011396: This split has 437 training and 110 validation cases.
2024-05-28 15:08:00.016630: predicting ToothFairy2F_002009
2024-05-28 15:08:00.045122: ToothFairy2F_002009, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:08:18.669225: predicting ToothFairy2F_002010
2024-05-28 15:08:18.705705: ToothFairy2F_002010, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:08:18.758248: predicting ToothFairy2F_002011
```

```
2024-05-28 15:08:18.798063: ToothFairy2F_002011, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:08:18.861753: predicting ToothFairy2F_002027
2024-05-28 15:08:18.898897: ToothFairy2F_002027, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:08:19.046290: predicting ToothFairy2F_002028
2024-05-28 15:08:19.077284: ToothFairy2F_002028, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:08:19.259306: predicting ToothFairy2F_002033
```

Fold 1

```
# Set up environment variables
os.environ["nnUNet_raw"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_raw"
os.environ["nnUNet_preprocessed"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed"
os.environ["nnUNet_results"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_results"

!nnUNetv2_train Dataset002_ToothFairy2 2d 1 --npz -tr nnUNetTrainer_1epoch
```

```
############################
INFO: You are using the old nnU-Net default plans. We have updated our recommendations. Please consider using those instead! Read
############################

Using device: cuda:0

#######################################################################
Please cite the following paper when using nnU-Net:
Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep le
#######################################################################

2024-05-28 15:11:07.537136: do_dummy_2d_data_aug: False
2024-05-28 15:11:07.576179: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:11:07.580638: The split file contains 5 splits.
2024-05-28 15:11:07.582639: Desired fold for training: 1
2024-05-28 15:11:07.599647: This split has 437 training and 110 validation cases.
using pin_memory on device 0
using pin_memory on device 0
2024-05-28 15:11:41.800439: Using torch.compile...
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:28: UserWarning: The verbose parameter is deprecated. Please
  warnings.warn("The verbose parameter is deprecated. Please use get_last_lr() "

This is the configuration used by this training:
Configuration name: 2d
 {'data_identifier': 'nnUNetPlans_2d', 'preprocessor_name': 'DefaultPreprocessor', 'batch_size': 16, 'patch_size': [448, 448], 'm

These are the global plan.json settings:
 {'dataset_name': 'Dataset002_ToothFairy2', 'plans_name': 'nnUNetPlans', 'original_median_spacing_after_transp': [999.0, 1.0, 1.0

2024-05-28 15:11:43.402150: unpacking dataset...
2024-05-28 15:11:56.007377: unpacking done...
2024-05-28 15:11:56.026522: Unable to plot network architecture: nnUNet_compile is enabled!
2024-05-28 15:11:56.037762:
2024-05-28 15:11:56.040071: Epoch 0
2024-05-28 15:11:56.042820: Current learning rate: 0.01
2024-05-28 15:15:44.335642: train_loss -0.031
2024-05-28 15:15:44.342568: val_loss -0.2852
2024-05-28 15:15:44.349296: Pseudo dice [0.5555, 0.0, 0.5902, 0.0001]
2024-05-28 15:15:44.356447: Epoch time: 228.3 s
2024-05-28 15:15:44.380624: Yayy! New best EMA pseudo Dice: 0.2865
2024-05-28 15:15:49.469850: Training done.
2024-05-28 15:15:49.753139: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:15:49.784560: The split file contains 5 splits.
2024-05-28 15:15:49.787868: Desired fold for training: 1
2024-05-28 15:15:49.790601: This split has 437 training and 110 validation cases.
2024-05-28 15:15:49.795860: predicting ToothFairy2F_002003
2024-05-28 15:15:49.823298: ToothFairy2F_002003, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:16:03.383966: predicting ToothFairy2F_002004
2024-05-28 15:16:03.405037: ToothFairy2F_002004, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:16:03.492874: predicting ToothFairy2F_002017
2024-05-28 15:16:03.521540: ToothFairy2F_002017, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:16:03.583148: predicting ToothFairy2F_002018
2024-05-28 15:16:03.628292: ToothFairy2F_002018, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:16:03.756297: predicting ToothFairy2F_002019
2024-05-28 15:16:03.819294: ToothFairy2F_002019, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:16:04.013247: predicting ToothFairy2F_002020
```

Fold 2

```
# Set up environment variables
os.environ["nnUNet_raw"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_raw"
os.environ["nnUNet_preprocessed"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed"
os.environ["nnUNet_results"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_results"

!nnUNetv2_train Dataset002_ToothFairy2 2d 2 --npz -tr nnUNetTrainer_1epoch
```

```
############################
INFO: You are using the old nnU-Net default plans. We have updated our recommendations. Please consider using those instead! Read
############################

Using device: cuda:0

#######################################################################
Please cite the following paper when using nnU-Net:
Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep le
#######################################################################

2024-05-28 15:17:50.083950: do_dummy_2d_data_aug: False
2024-05-28 15:17:50.121242: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:17:50.126234: The split file contains 5 splits.
2024-05-28 15:17:50.128554: Desired fold for training: 2
2024-05-28 15:17:50.130855: This split has 438 training and 109 validation cases.
using pin_memory on device 0
using pin_memory on device 0
2024-05-28 15:18:02.216896: Using torch.compile...
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:28: UserWarning: The verbose parameter is deprecated. Please
  warnings.warn("The verbose parameter is deprecated. Please use get_last_lr() "

This is the configuration used by this training:
Configuration name: 2d
 {'data_identifier': 'nnUNetPlans_2d', 'preprocessor_name': 'DefaultPreprocessor', 'batch_size': 16, 'patch_size': [448, 448], 'm

These are the global plan.json settings:
 {'dataset_name': 'Dataset002_ToothFairy2', 'plans_name': 'nnUNetPlans', 'original_median_spacing_after_transp': [999.0, 1.0, 1.0

2024-05-28 15:18:03.779040: unpacking dataset...
2024-05-28 15:18:17.495768: unpacking done...
2024-05-28 15:18:17.519878: Unable to plot network architecture: nnUNet_compile is enabled!
2024-05-28 15:18:17.544964:
2024-05-28 15:18:17.547827: Epoch 0
2024-05-28 15:18:17.550184: Current learning rate: 0.01
2024-05-28 15:22:12.860379: train_loss -0.038
2024-05-28 15:22:12.867361: val_loss -0.2718
2024-05-28 15:22:12.872639: Pseudo dice [0.5693, 0.2024, 0.5371, 0.0]
2024-05-28 15:22:12.877551: Epoch time: 235.32 s
2024-05-28 15:22:12.881779: Yayy! New best EMA pseudo Dice: 0.3272
2024-05-28 15:22:16.954332: Training done.
2024-05-28 15:22:17.212164: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:22:17.247596: The split file contains 5 splits.
2024-05-28 15:22:17.251655: Desired fold for training: 2
2024-05-28 15:22:17.257004: This split has 438 training and 109 validation cases.
2024-05-28 15:22:18.691901: predicting ToothFairy2F_002007
2024-05-28 15:22:18.737743: ToothFairy2F_002007, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:22:31.814950: predicting ToothFairy2F_002008
2024-05-28 15:22:31.835139: ToothFairy2F_002008, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:22:31.897326: predicting ToothFairy2F_002013
2024-05-28 15:22:31.953832: ToothFairy2F_002013, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:22:32.028689: predicting ToothFairy2F_002014
2024-05-28 15:22:32.062359: ToothFairy2F_002014, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:22:32.176317: predicting ToothFairy2F_002016
2024-05-28 15:22:32.204754: ToothFairy2F_002016, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:22:32.375396: predicting ToothFairy2F_002021
```

Fold 3

```
# Set up environment variables
os.environ["nnUNet_raw"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_raw"
os.environ["nnUNet_preprocessed"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed"
os.environ["nnUNet_results"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_results"

!nnUNetv2_train Dataset002_ToothFairy2 2d 3 --npz -tr nnUNetTrainer_1epoch
```

```
############################
INFO: You are using the old nnU-Net default plans. We have updated our recommendations. Please consider using those instead! Read
############################

Using device: cuda:0

#######################################################################
Please cite the following paper when using nnU-Net:
Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep le
#######################################################################

2024-05-28 15:23:41.564083: do_dummy_2d_data_aug: False
2024-05-28 15:23:41.607317: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:23:41.618785: The split file contains 5 splits.
2024-05-28 15:23:41.621963: Desired fold for training: 3
2024-05-28 15:23:41.625435: This split has 438 training and 109 validation cases.
using pin_memory on device 0
using pin_memory on device 0
```

```
2024-05-28 15:23:55.561326: Using torch.compile...
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:28: UserWarning: The verbose parameter is deprecated. Please
  warnings.warn("The verbose parameter is deprecated. Please use get_last_lr() "

This is the configuration used by this training:
Configuration name: 2d
 {'data_identifier': 'nnUNetPlans_2d', 'preprocessor_name': 'DefaultPreprocessor', 'batch_size': 16, 'patch_size': [448, 448], 'm

These are the global plan.json settings:
 {'dataset_name': 'Dataset002_ToothFairy2', 'plans_name': 'nnUNetPlans', 'original_median_spacing_after_transp': [999.0, 1.0, 1.0

2024-05-28 15:23:57.554782: unpacking dataset...
2024-05-28 15:24:11.285285: unpacking done...
2024-05-28 15:24:11.308355: Unable to plot network architecture: nnUNet_compile is enabled!
2024-05-28 15:24:11.319821:
2024-05-28 15:24:11.322197: Epoch 0
2024-05-28 15:24:11.349436: Current learning rate: 0.01
2024-05-28 15:28:05.527112: train_loss -0.0088
2024-05-28 15:28:05.531959: val_loss -0.263
2024-05-28 15:28:05.534993: Pseudo dice [0.6227, 0.0, 0.3778, 0.0]
2024-05-28 15:28:05.539384: Epoch time: 234.21 s
2024-05-28 15:28:05.542523: Yayy! New best EMA pseudo Dice: 0.2501
2024-05-28 15:28:09.499666: Training done.
2024-05-28 15:28:09.812057: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:28:09.836602: The split file contains 5 splits.
2024-05-28 15:28:09.839795: Desired fold for training: 3
2024-05-28 15:28:09.842266: This split has 438 training and 109 validation cases.
2024-05-28 15:28:09.847421: predicting ToothFairy2F_002001
2024-05-28 15:28:09.875448: ToothFairy2F_002001, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:28:24.135622: predicting ToothFairy2F_002002
2024-05-28 15:28:24.151436: ToothFairy2F_002002, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:28:24.215823: predicting ToothFairy2F_002015
2024-05-28 15:28:24.249116: ToothFairy2F_002015, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:28:24.335299: predicting ToothFairy2F_002024
2024-05-28 15:28:24.361186: ToothFairy2F_002024, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:28:24.482627: predicting ToothFairy2F_002025
2024-05-28 15:28:24.552043: ToothFairy2F_002025, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:28:24.714817: predicting ToothFairy2F_002026
```

Fold 4

```
# Set up environment variables
os.environ["nnUNet_raw"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_raw"
os.environ["nnUNet_preprocessed"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed"
os.environ["nnUNet_results"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_results"

!nnUNetv2_train Dataset002_ToothFairy2 2d 4 --npz -tr nnUNetTrainer_1epoch
```

```
############################
INFO: You are using the old nnU-Net default plans. We have updated our recommendations. Please consider using those instead! Read
############################

Using device: cuda:0

#######################################################################
Please cite the following paper when using nnU-Net:
Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep le
#######################################################################

2024-05-28 15:29:32.964084: do_dummy_2d_data_aug: False
2024-05-28 15:29:33.006796: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:29:33.016733: The split file contains 5 splits.
2024-05-28 15:29:33.019301: Desired fold for training: 4
2024-05-28 15:29:33.021438: This split has 438 training and 109 validation cases.
using pin_memory on device 0
using pin_memory on device 0
2024-05-28 15:29:47.179456: Using torch.compile...
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:28: UserWarning: The verbose parameter is deprecated. Please
  warnings.warn("The verbose parameter is deprecated. Please use get_last_lr() "

This is the configuration used by this training:
Configuration name: 2d
 {'data_identifier': 'nnUNetPlans_2d', 'preprocessor_name': 'DefaultPreprocessor', 'batch_size': 16, 'patch_size': [448, 448], 'm

These are the global plan.json settings:
 {'dataset_name': 'Dataset002_ToothFairy2', 'plans_name': 'nnUNetPlans', 'original_median_spacing_after_transp': [999.0, 1.0, 1.0

2024-05-28 15:29:49.060031: unpacking dataset...
2024-05-28 15:30:02.288567: unpacking done...
2024-05-28 15:30:02.323434: Unable to plot network architecture: nnUNet_compile is enabled!
2024-05-28 15:30:02.334962:
2024-05-28 15:30:02.337354: Epoch 0
2024-05-28 15:30:02.340218: Current learning rate: 0.01
2024-05-28 15:34:11.461390: train_loss -0.033
2024-05-28 15:34:11.467136: val_loss -0.2593
2024-05-28 15:34:11.472549: Pseudo dice [0.5205, 0.0057, 0.511, 0.0001]
```

```
2024-05-28 15:34:11.478359: Epoch time: 249.13 s
2024-05-28 15:34:11.483985: Yayy! New best EMA pseudo Dice: 0.2593
2024-05-28 15:34:15.582025: Training done.
2024-05-28 15:34:15.868290: Using splits from existing split file: /content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed/Datase
2024-05-28 15:34:15.890554: The split file contains 5 splits.
2024-05-28 15:34:15.894067: Desired fold for training: 4
2024-05-28 15:34:15.917919: This split has 438 training and 109 validation cases.
2024-05-28 15:34:15.923771: predicting ToothFairy2F_002000
2024-05-28 15:34:15.953591: ToothFairy2F_002000, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:34:29.759189: predicting ToothFairy2F_002005
2024-05-28 15:34:29.793425: ToothFairy2F_002005, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:34:29.853173: predicting ToothFairy2F_002006
2024-05-28 15:34:29.905691: ToothFairy2F_002006, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:34:29.966186: predicting ToothFairy2F_002012
2024-05-28 15:34:30.028902: ToothFairy2F_002012, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:34:30.145255: predicting ToothFairy2F_002023
2024-05-28 15:34:30.184034: ToothFairy2F_002023, shape torch.Size([1, 1, 410, 410]), rank 0
2024-05-28 15:34:30.312616: predicting ToothFairy2F_002029
```

3.3 Encontrar la mejor configuración de entrenamiento

```
import os
os.environ["nnUNet_raw"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_raw"
os.environ["nnUNet_preprocessed"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_preprocessed"
os.environ["nnUNet_results"]="/content/drive/MyDrive/saros/nnUNet/nnUNet_results"
```

```
!nnUNetv2_find_best_configuration Dataset002_ToothFairy2 -c 2d -tr nnUNetTrainer_1epoch
```

```
***All results:***
nnUNetTrainer_1epoch__nnUNetPlans__2d: 0.12887671802268819

*Best*: nnUNetTrainer_1epoch__nnUNetPlans__2d: 0.12887671802268819

***Determining postprocessing for best model/ensemble***
Removing all but the largest foreground region did not improve results!
Removing all but the largest component for 1 did not improve results! Dice before: 0.40021 after: 0.26004
Removing all but the largest component for 2 did not improve results! Dice before: 0.06179 after: 0.04279
Removing all but the largest component for (11, 12, 13, 14, 15, 16, 17, 18, 21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 33, 34, 35, 36,
Removing all but the largest component for (3, 4, 5, 6, 7, 8, 9, 10, 18, 19, 20, 29, 30, 38, 39, 40) did not improve results! Dice

***Run inference like this:***

nnUNetv2_predict -d Dataset002_ToothFairy2 -i INPUT_FOLDER -o OUTPUT_FOLDER -f  0 1 2 3 4 -tr nnUNetTrainer_1epoch -c 2d -p nnUNetP

***Once inference is completed, run postprocessing like this:***

nnUNetv2_apply_postprocessing -i OUTPUT_FOLDER -o OUTPUT_FOLDER_PP -pp_pkl_file /content/drive/MyDrive/saros/nnUNet/nnUNet_results/
```

3.4. Resultados de entrenamiento

NnUNet no proporciona una función específica para visualizar métricas de entrenamiento en tiempo real, pero guarda estos datos en archivos de registro. Por ejemplo los datos en el archivo "summary.json" en la carpeta "crossval_results_folds_0_1_2_3_4" contienen las métricas de evaluación del entrenamiento cruzado.

```
directorio = '/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/nnUNetTrainer_1epoch__nnUNetPlans__2d/crossval_

# Comprobar si el directorio existe
os.path.exists(directorio)
```

```
True
```

```
# Ruta al archivo summary.json
nombre_archivo = 'summary.json'
# Unir directorio y nombre del archivo
summary_file = os.path.join(directorio, nombre_archivo)


def load_summary(json_path):
    with open(json_path, 'r') as file:
        summary = json.load(file)
    return summary

def display_metrics_table(metrics):
    df = pd.DataFrame(metrics, index=[0])
    return df

# Cargar las métricas del archivo JSON
summary = load_summary(summary_file)

# Extraer las métricas de foreground_mean
foreground_mean = summary['foreground_mean']

# Mostrar las métricas en una tabla
metrics_table = display_metrics_table(foreground_mean)
metrics_table
```

	Dice	FN	FP	IoU	TN	TP	n_pred
0	0.128877	2645.494059	1214.254113	0.085621	163551.708867	688.542962	1902.797075

4. Predicción utilizando conjunto de test

4.1 inferencia

Realizamos predicciones utilizando un modelo previamente entrenado en el conjunto de datos utilisando clase de entrenador personalizado. Guardamos los resultados en la carpeta "inferencia"

```
!nnUNetv2_predict -d Dataset002_ToothFairy2 -i /content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/imagesTs -o /conte
```

```
#######################################################################
Please cite the following paper when using nnU-Net:
Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). nnU-Net: a self-configuring method for deep le
#######################################################################

There are 274 cases in the source folder
I am process 0 out of 1 (max process ID is 0, we start counting with 0!)
There are 274 cases that I would like to predict

Predicting ToothFairy2F_051000:
perform_everything_on_device: True
100% 1/1 [00:01<00:00,  1.09s/it]
100% 1/1 [00:00<00:00, 25.22it/s]
100% 1/1 [00:00<00:00, 28.02it/s]
100% 1/1 [00:00<00:00, 24.36it/s]
100% 1/1 [00:00<00:00, 12.49it/s]
sending off prediction to background worker for resampling and export
done with ToothFairy2F_051000

Predicting ToothFairy2F_051001:
perform_everything_on_device: True
100% 1/1 [00:00<00:00, 27.24it/s]
100% 1/1 [00:00<00:00, 27.42it/s]
100% 1/1 [00:00<00:00, 25.21it/s]
100% 1/1 [00:00<00:00, 26.33it/s]
100% 1/1 [00:00<00:00, 23.48it/s]
sending off prediction to background worker for resampling and export
done with ToothFairy2F_051001

Predicting ToothFairy2F_051002:
perform_everything_on_device: True
100% 1/1 [00:00<00:00, 28.97it/s]
100% 1/1 [00:00<00:00, 31.16it/s]
100% 1/1 [00:00<00:00, 29.58it/s]
100% 1/1 [00:00<00:00, 25.92it/s]
100% 1/1 [00:00<00:00, 25.37it/s]
sending off prediction to background worker for resampling and export
done with ToothFairy2F_051002

Predicting ToothFairy2F_051003:
perform_everything_on_device: True
```

```
100% 1/1 [00:00<00:00, 21.07it/s]
100% 1/1 [00:00<00:00, 29.60it/s]
100% 1/1 [00:00<00:00, 29.74it/s]
100% 1/1 [00:00<00:00, 30.41it/s]
100% 1/1 [00:00<00:00, 29.85it/s]
sending off prediction to background worker for resampling and export
done with ToothFairy2F_051003

Predicting ToothFairy2F_051004:
perform_everything_on_device: True
100% 1/1 [00:00<00:00, 22.61it/s]
100% 1/1 [00:00<00:00, 27.61it/s]
100% 1/1 [00:00<00:00, 27.13it/s]
100% 1/1 [00:00<00:00, 26.70it/s]
100% 1/1 [00:00<00:00, 28.48it/s]
```

4.2 Postprocesamiento

Ahora ejecutemos el paso de postprocesamiento para generar los archivos de segmentación final utilizando el mejor modelo determinado por nnUNet en función del puntaje DICE. La carpeta de entrada es nuestra carpeta de "inferencia", y la carpeta de salida es una carpeta "postprocesamiento".

```
!nnUNetv2_apply_postprocessing -i /content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/inference -o /content/drive
```

4.3. Resultados de test

```
!nnUNetv2_evaluate_folder -h
```

```
usage: nnUNetv2_evaluate_folder [-h] -djfile DJFILE -pfile PFILE [-o O] [-np NP] [--chill]
                                gt_folder pred_folder

positional arguments:
  gt_folder       folder with gt segmentations
  pred_folder     folder with predicted segmentations

options:
  -h, --help      show this help message and exit
  -djfile DJFILE  dataset.json file
  -pfile PFILE    plans.json file
  -o O            Output file. Optional. Default: pred_folder/summary.json
  -np NP          number of processes used. Optional. Default: 8
  --chill         dont crash if folder_pred does not have all files that are present in folder_gt
```

```
FOLDER_IMAGES_TEST='/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/imagesTs'
FOLDER_LABELS_TEST_GT='/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/labelsTs'
FOLDER_LABELS_TEST_PREDICTIONS='/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/postprocessing'
DJFILE_TEST_PREDICTIONS='/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/nnUNetTrainer_1epoch__nnUNetPlans__2d/
PFILE_TEST_PREDICTIONS='/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/nnUNetTrainer_1epoch__nnUNetPlans__2d/p

!nnUNetv2_evaluate_folder $FOLDER_LABELS_TEST_GT $FOLDER_LABELS_TEST_PREDICTIONS -djfile $DJFILE_TEST_PREDICTIONS -pfile $PFILE_TEST_PRED

# Ruta al archivo summary.json
nombre_archivo = 'summary.json'
# Unir directorio y nombre del archivo
summary_file = os.path.join(FOLDER_LABELS_TEST_PREDICTIONS, nombre_archivo)


def load_summary(json_path):
    with open(json_path, 'r') as file:
        summary = json.load(file)
    return summary

def display_metrics_table(metrics):
    df = pd.DataFrame(metrics, index=[0])
    return df

# Cargar las métricas del archivo JSON
summary = load_summary(summary_file)

# Extraer las métricas de foreground_mean
foreground_mean = summary['foreground_mean']

# Mostrar las métricas en una tabla
metrics_table = display_metrics_table(foreground_mean)
metrics_table
```

```
Using <class 'nnunetv2.imageio.simpleitk_reader_writer.SimpleITKIO'> as reader/writer
```

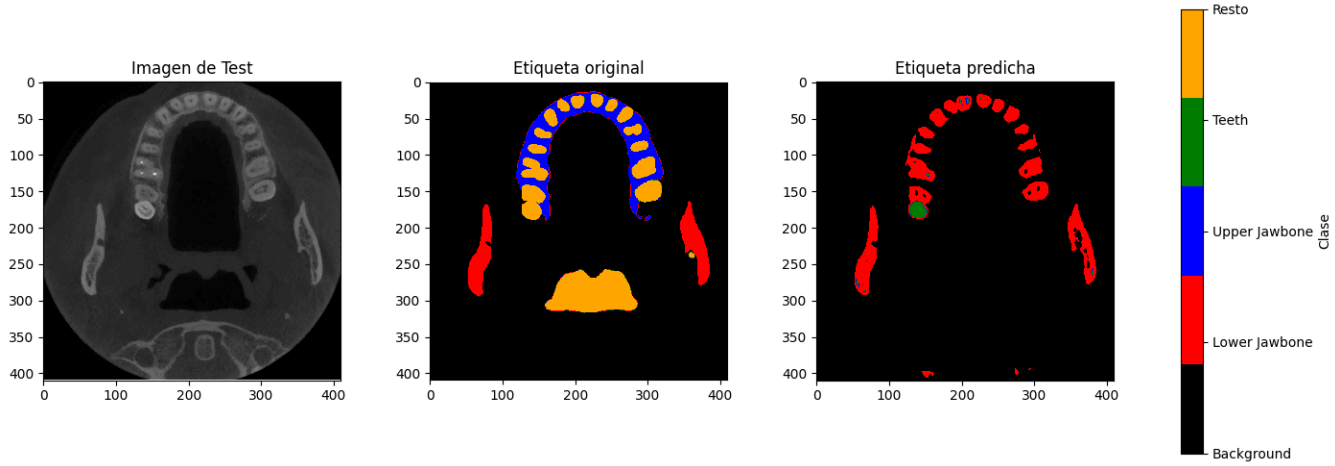
	Dice	FN	FP	IoU	TN	TP	n_pred	n_ref
0	0.161049	2931.186131	1037.612226	0.123197	162992.421533	1138.780109	2176.392336	4069.966241

```
import SimpleITK as sitk
import matplotlib.pyplot as plt
import os
from matplotlib.colors import ListedColormap

def show_two_nii_images(file_name1, file_name2, file_name3, ruta_imagen_test, ruta_label_test, ruta_label_predicho):
    """
    Muestra dos imágenes .nii.gz en una sola línea.

    Args
        file_name1 (str): imagen de test.nii.gz (sin ruta).
        file_name2 (str): label de test .nii.gz (sin ruta).
        file_name3 (str): label predicho .nii.gz (sin ruta).
        slice_index (int): Índice de la slice a mostrar (por defecto es 0).
    """
    # Directorios predeterminados
    image_dir1 = ruta_imagen_test
    image_dir2 = ruta_label_test
    image_dir3 = ruta_label_predicho

    # Construir las rutas completas de los archivos
    image_path1 = os.path.join(image_dir1, file_name1)
    image_path2 = os.path.join(image_dir2, file_name2)
    image_path3 = os.path.join(image_dir3, file_name3)

    print(image_path1)
    print(image_path2)
    print(image_path3)

    # Leer imágenes
    image1 = sitk.ReadImage(image_path1)
    image2 = sitk.ReadImage(image_path2)
    image3 = sitk.ReadImage(image_path3)

    # Convertir las imágenes a arrays numpy y reformatear
    image_array1 = sitk.GetArrayFromImage(image1)
    image_array2 = sitk.GetArrayFromImage(image2)
    image_array3 = sitk.GetArrayFromImage(image3)

    # Colores distintivos para las clases
    cmap = ListedColormap(['black','red', 'blue', 'green', 'orange'])

   # Mostrar las imágenes
    fig, axes = plt.subplots(1, 3, figsize=(18, 6))  # Ajusta el tamaño de la figura
    plt.subplots_adjust(wspace=0.3)  # Ajusta el espaciado entre subgráficos

    # Primera imagen
    axes[0].imshow(image_array1, cmap='gray')  # Muestra la imagen en escala de grises
    axes[0].set_title('Imagen de Test')

    # Segunda imagen
    im = axes[1].imshow(image_array2, cmap=cmap, vmin=0, vmax=4)  # Muestra la imagen de label de test
    axes[1].set_title('Etiqueta original')

    # Tercera imagen
    axes[2].imshow(image_array3, cmap=cmap, vmin=0, vmax=4)  # Muestra la imagen de label predicho
    axes[2].set_title('Etiqueta predicha')

    # Crear y mostrar leyenda independiente
    legend_labels = ['Background', 'Lower Jawbone', 'Upper Jawbone', 'Teeth', 'Resto']
    cbar = fig.colorbar(im, ax=axes.ravel().tolist(), ticks=[0, 1, 2, 3, 4], orientation='vertical', label='Clase')
    cbar.ax.set_yticklabels(legend_labels)

    plt.show()
```

4.4 Visualisaciones: imagen de test, etiqueta original y etiqueta predicha

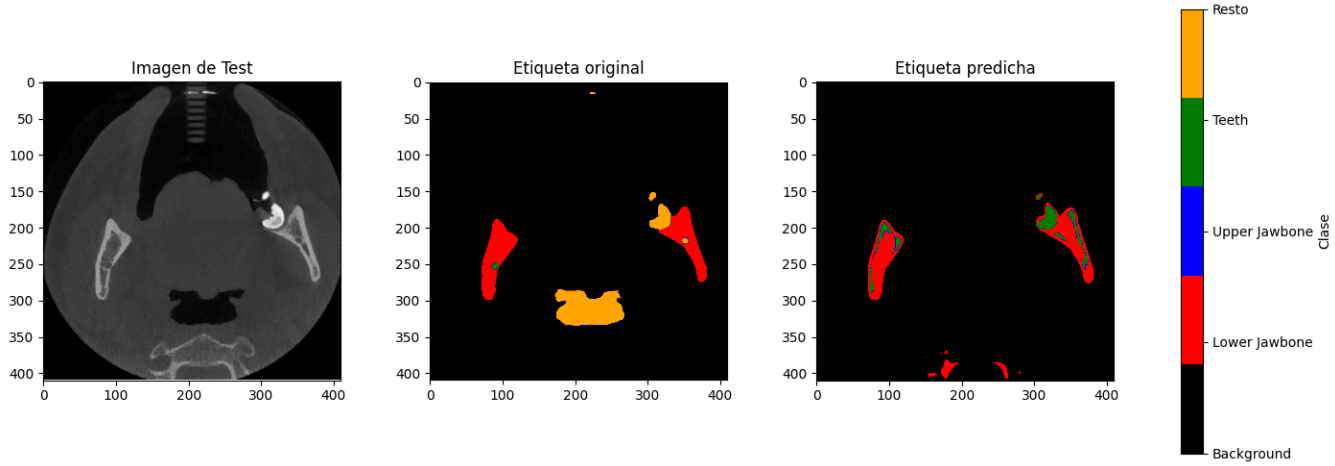
```
# Ejemplo de uso de la función
file_name1 = 'ToothFairy2F_051050_0000.nii.gz'
file_name2 = 'ToothFairy2F_051050.nii.gz'
file_name3 = 'ToothFairy2F_051050.nii.gz'
show_two_nii_images(file_name1, file_name2, file_name3, FOLDER_IMAGES_TEST, FOLDER_LABELS_TEST_GT, FOLDER_LABELS_TEST_PREDICTIONS)
```

```
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/imagesTs/ToothFairy2F_051050_0000.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/labelsTs/ToothFairy2F_051050.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/postprocessing/ToothFairy2F_051050.nii.gz
```



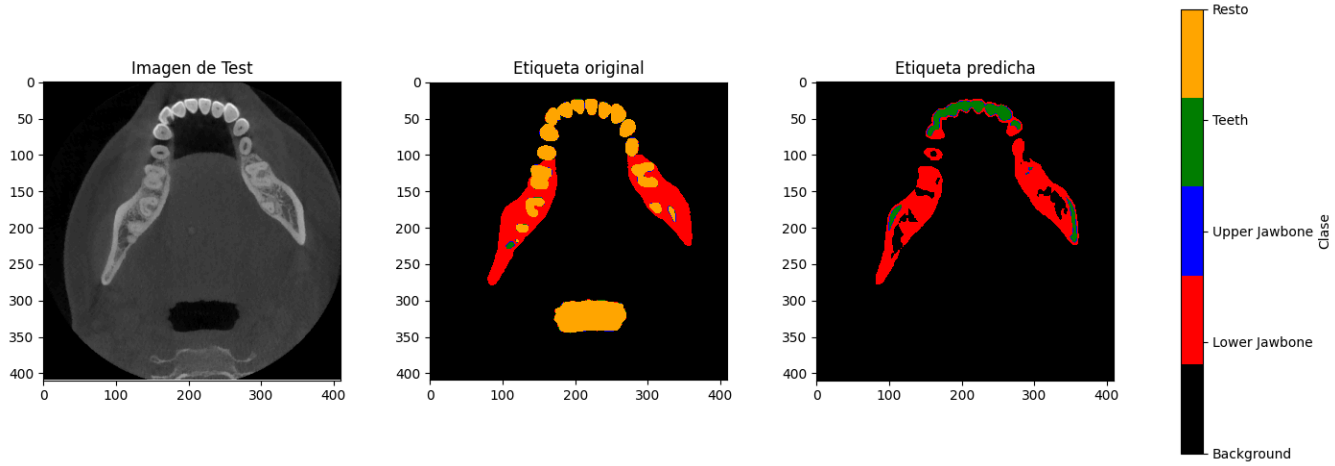

```
# Ejemplo de uso de la función
file_name1 = 'ToothFairy2F_051100_0000.nii.gz'
file_name2 = 'ToothFairy2F_051100.nii.gz'
file_name3 = 'ToothFairy2F_051100.nii.gz'
show_two_nii_images(file_name1, file_name2, file_name3, FOLDER_IMAGES_TEST, FOLDER_LABELS_TEST_GT, FOLDER_LABELS_TEST_PREDICTIONS)
```

```
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/imagesTs/ToothFairy2F_051100_0000.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/labelsTs/ToothFairy2F_051100.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/postprocessing/ToothFairy2F_051100.nii.gz
```



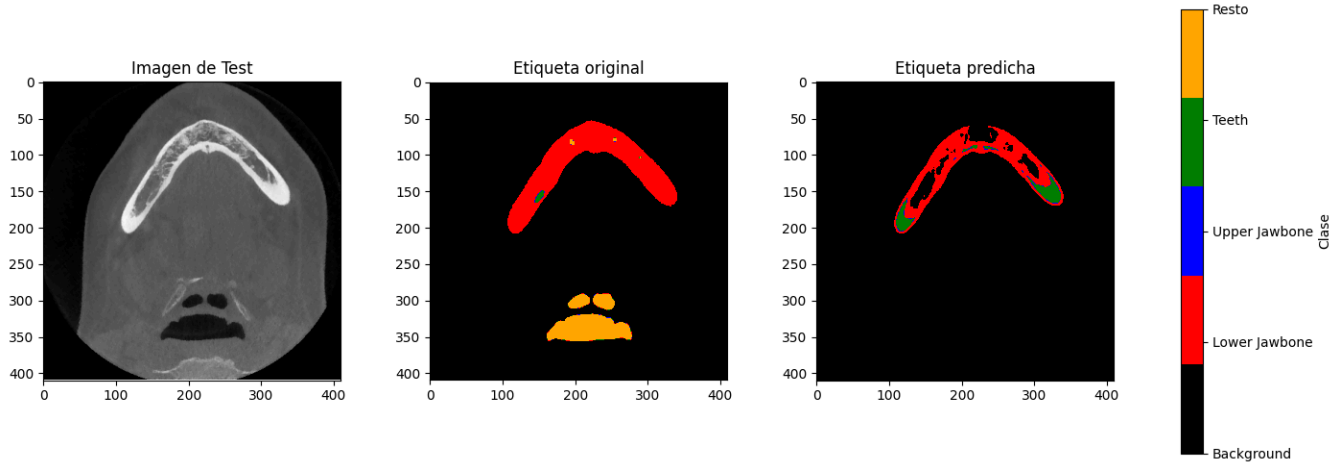

```
# Ejemplo de uso de la función
file_name1 = 'ToothFairy2F_051150_0000.nii.gz'
file_name2 = 'ToothFairy2F_051150.nii.gz'
file_name3 = 'ToothFairy2F_051150.nii.gz'
show_two_nii_images(file_name1, file_name2, file_name3, FOLDER_IMAGES_TEST, FOLDER_LABELS_TEST_GT, FOLDER_LABELS_TEST_PREDICTIONS)
```

```
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/imagesTs/ToothFairy2F_051150_0000.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/labelsTs/ToothFairy2F_051150.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/postprocessing/ToothFairy2F_051150.nii.gz
```




```
# Ejemplo de uso de la función
file_name1 = 'ToothFairy2F_051200_0000.nii.gz'
file_name2 = 'ToothFairy2F_051200.nii.gz'
file_name3 = 'ToothFairy2F_051200.nii.gz'
show_two_nii_images(file_name1, file_name2, file_name3, FOLDER_IMAGES_TEST, FOLDER_LABELS_TEST_GT, FOLDER_LABELS_TEST_PREDICTIONS)
```

```
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/imagesTs/ToothFairy2F_051200_0000.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/labelsTs/ToothFairy2F_051200.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/postprocessing/ToothFairy2F_051200.nii.gz
```




```
# Ejemplo de uso de la función
file_name1 = 'ToothFairy2F_051225_0000.nii.gz'
file_name2 = 'ToothFairy2F_051225.nii.gz'
file_name3 = 'ToothFairy2F_051225.nii.gz'
show_two_nii_images(file_name1, file_name2, file_name3, FOLDER_IMAGES_TEST, FOLDER_LABELS_TEST_GT, FOLDER_LABELS_TEST_PREDICTIONS)
```

```
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/imagesTs/ToothFairy2F_051225_0000.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_raw/Dataset002_ToothFairy2/labelsTs/ToothFairy2F_051225.nii.gz
/content/drive/MyDrive/saros/nnUNet/nnUNet_results/Dataset002_ToothFairy2/postprocessing/ToothFairy2F_051225.nii.gz
```