

PRESENTACIÓN PARCIAL DEL SOFTWARE

APLIACIONES CLINICAS A PORCESAMIENTO DE
SEÑALES E IMÁGENES



PRESENTED BY : GRUPO 3



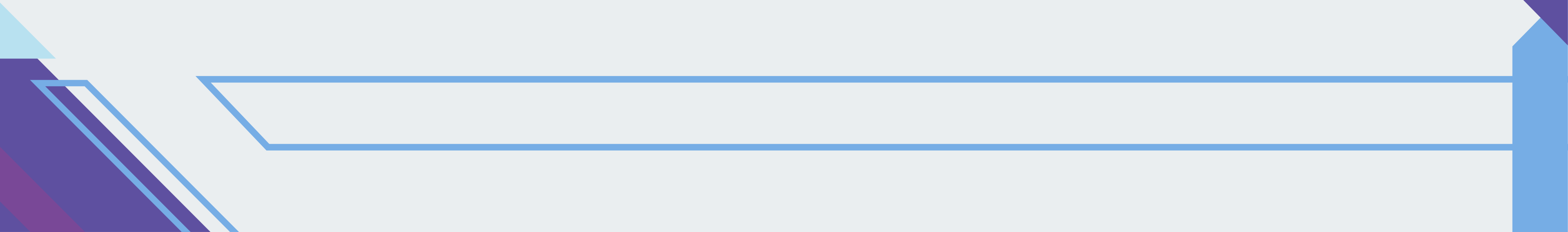
INDICE



01 **PROBLEMATICA**

02 **ALGORITMO**

03 **FRONTEND**



PROBLEMÁTICA

La **centralización de recursos y especialistas**, junto con la falta de un sistema de incentivos, genera una desigualdad en el acceso a servicios médicos especializados, como el diagnóstico por ultrasonido, entre áreas urbanas y rurales.

2020

114 Pediatras
en Cusco



354 médicos en el
Sector público



La **colecistitis** es la formación de cálculos biliares en la vesícula biliar.

95%

Enfermedades de la vía
biliar se atribuyen a
colecistitis

7%

Incidencia en la
ciudad del Cusco

[1] "Instituto Nacional de Estadística e Informática", Gob.pe. [En línea]. Disponible en: <https://www.gob.pe/institucion/inei/informes-publicaciones/5379323-compendio-estadistico-cusco-2023>. [Consultado: 25-mar-2024].

[2] Y. L. Condori Chillihuani, "CORRELACIÓN ECOGRÁFICA Y HALLAZGOS QUIRÚRGICOS EN PACIENTES COLECISTECTOMIZADOS EN LOS HOSPITALES DE LA CIUDAD DEL CUSCO DURANTE EL PERÍODO 2012-2016", Univ. Andin. Del Cusco, Cusco, 2018. Accedido el 14 de abril de 2024. [En línea]. Disponible: <https://hdl.handle.net/20.500.12557/1672>

INSUFICIENCIA DE PERSONAL CAPACITADO PARA INTERPRETAR LAS IMÁGENES DE ULTRASONIDO



VARIABLES



VARIABLES DEL DATASET

- Imágenes de TC
- Imágenes de ultrasonido



VARIABLE TARGET

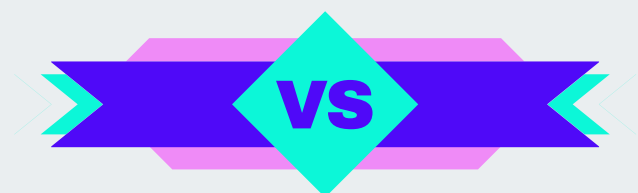
- Imágenes de ultrasonido sintéticas



ALGORITMO UTILIZADO... S-CYCLEGAN

Un algoritmo S-CycleGAN es una **versión mejorada** de **CycleGAN** diseñada para tareas de **traducción de imágenes** entre dominios (por ejemplo, de tomografía computarizada a ultrasonido), **preservando detalles semánticos** importantes.

CARACTERISTICAS IMPORTANTES:



GENERADORES Y DISCRIMINADORES ADVERSARIALES

2 generadores convierten imágenes de un dominio a otro, mientras que otros 2 discriminadores intentan distinguir entre imágenes reales y sintéticas

updates

SEGMENTADORES SEMÁNTICOS

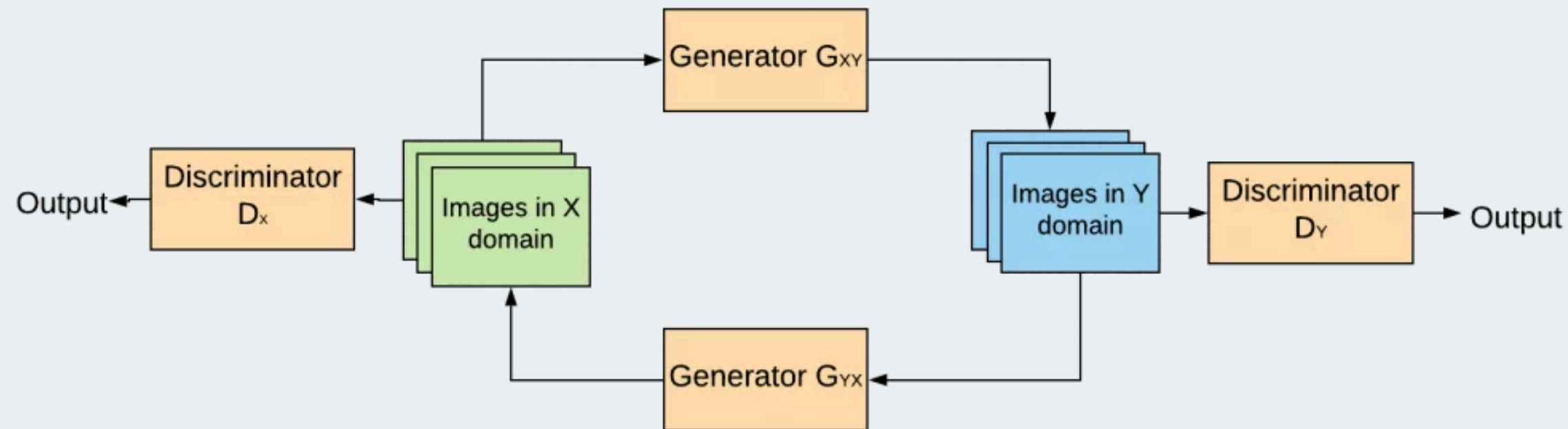
S-CycleGAN incluye redes de segmentación que actúan como discriminadores semánticos.



COMO FUNCIONA?

Generadores:

- G_{xy} : Es el generador que transforma imágenes del dominio X al dominio Y.
- G_{yx} : Este generador realiza la operación inversa, transformando imágenes del dominio Y al dominio X.



Discriminadores:

- D_x : El discriminador que se encarga de verificar si una imagen en el dominio X es real o ha sido generada por G_{yx} .
- D_y : Similarmente, este discriminador determina si una imagen en el dominio Y es real o ha sido generada por G_{xy} .



CODIGO CYCLEGAN EXPLICACION



```
!git clone https://github.com/thepochynsons/pytorch-CycleGAN-and-pix2pix.git
```



```
Cloning into 'pytorch-CycleGAN-and-pix2pix'...  
remote: Enumerating objects: 2381, done.  
remote: Total 2381 (delta 0), reused 0 (delta 0), pack-reused 2381 (from 1)  
Receiving objects: 100% (2381/2381), 8.14 MiB | 29.24 MiB/s, done.  
Resolving deltas: 100% (1514/1514), done.
```

PROVEN TRACK
RECORD

```
import os  
import pickle  
os.chdir('pytorch-CycleGAN-and-pix2pix/')  
# install dependencies  
!pip install -r requirements.txt
```


PROVEN TRACK RECORD



```
# SHOW ABDOMINAL_US DATASET IMAGES
```

```
from os import path
```

```
from PIL import Image
```

```
import matplotlib.pyplot as plt
```

```
images = ['c37', 'j26', 'c15']
```

```
fig = plt.figure()
```

```
# for each input image
```

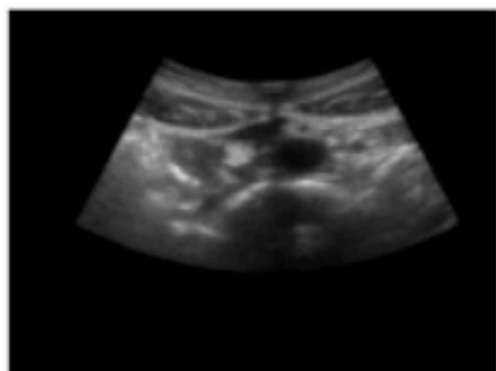
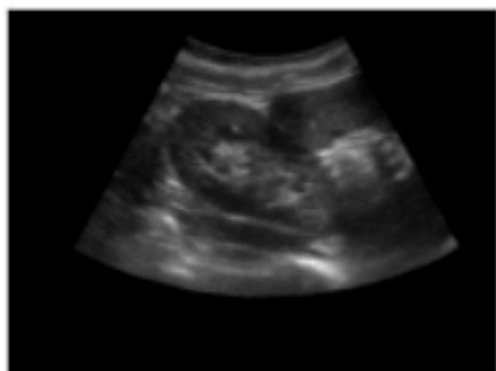
```
for idx in range(len(images)):
```

```
    # add a subplot
```

```
    ax = fig.add_subplot(1, len(images), idx+1, xticks=[], yticks=[])
```

```
    # display the image
```

```
    plt.imshow(Image.open(path.join("/content/drive/MyDrive/ACSI/abdo
```



```
# SHOW AUS2RUS DATASET IMAGES
```

```
rus_example = Image.open('/content/drive/MyDrive/ACSI/aus2rus/trainB/rotated_resize
```

```
aus_example = Image.open('/content/drive/MyDrive/ACSI/aus2rus/trainA/resized_ct14-1
```

```
fig = plt.figure()
```

```
ax = fig.add_subplot(1, 2, 1, xticks=[], yticks=[], title='Real Ultrasound')
```

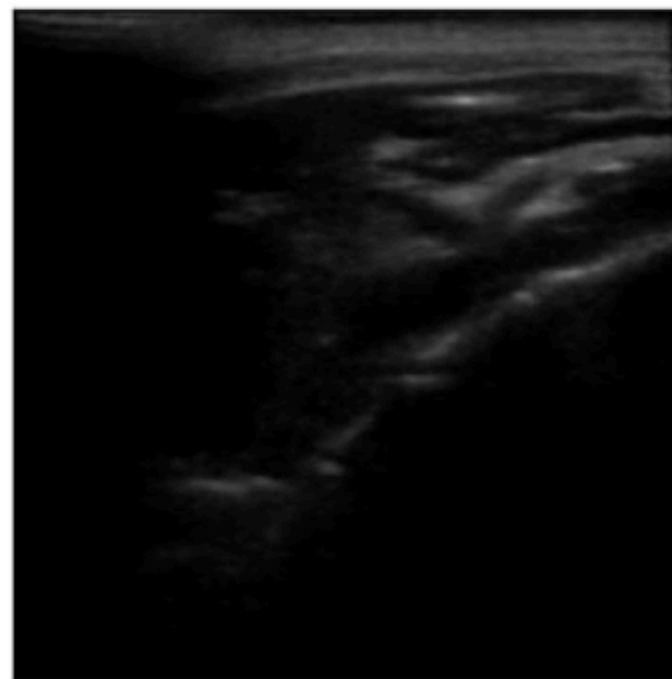
```
plt.imshow(rus_example)
```

```
ax = fig.add_subplot(1, 2, 2, xticks=[], yticks=[], title='Artificial Ultrasound')
```

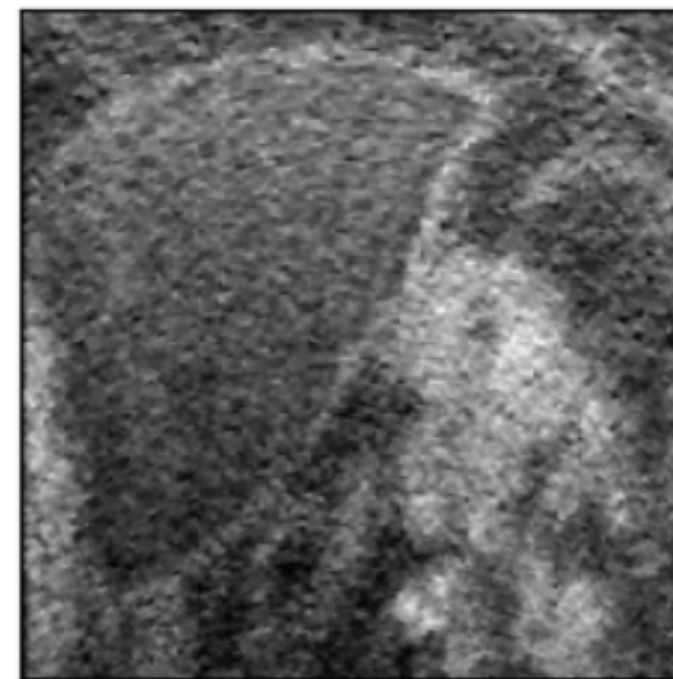
```
plt.imshow(aus_example)
```

```
plt.show()
```

Real Ultrasound



Artificial Ultrasound



PROVEN TRACK RECORD



```
import os
```

```
if not os.path.exists('./checkpoints'):  
    os.mkdir('./checkpoints')
```

```
!cp -r /content/drive/MyDrive/ACSI/pretrained_model_aus2rus/aus2rus ./checkpoints/aus2rus
```

```
!PYTHON TRAIN.PY --DATAROOT /CONTENT/DRIVE/MYDRIVE/ACSI/AUS2RUS --NAME NEW_AUS2RUS --MODEL CYCLE_GAN --
GAN_MODE VANILLA --NORM INSTANCE --NETG UNET_256 --PREPROCESS NONE --INPUT_NC 1 --OUTPUT_NC 1 --N_EPOCHS 1 --
N_EPOCHS_DECAY 0
```

```
init_type: normal
input_nc: 1
isTrain: True
lambda_A: 10.0
lambda_B: 10.0
lambda_identity: 0.5
load_iter: 0
load_size: 256
lr: 0.0002
lr_decay_iters: 50
lr_policy: linear
max_dataset_size: inf
model: cycle_gan
n_epochs: 1
n_epochs_decay: 0
n_layers_D: 3
name: new_au2rus
ndf: 64
```

```
----- End -----
dataset [UnalignedDataset] was created
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dat
warnings.warn(_create_warning_msg(
The number of training images = 633
cycleganmodel
initialize network with normal
initialize network with normal
initialize network with normal
initialize network with normal
model [CycleGANModel] was created
----- Networks initialized -----
[Network G_A] Total number of parameters : 54.403 M
[Network G_B] Total number of parameters : 54.403 M
[Network D_A] Total number of parameters : 2.763 M
[Network D_B] Total number of parameters : 2.763 M
-----
create web directory ./checkpoints/new_au2rus/web...
(epoch: 1, iters: 100, time: 0.292, data: 0.138) D_A: 0.190
(epoch: 1, iters: 200, time: 0.286, data: 0.002) D_A: 0.189
(epoch: 1, iters: 300, time: 0.298, data: 0.002) D_A: 0.088
(epoch: 1, iters: 400, time: 0.532, data: 0.002) D_A: 0.269
(epoch: 1, iters: 500, time: 0.306, data: 0.003) D_A: 0.180
(epoch: 1, iters: 600, time: 0.310, data: 0.002) D_A: 0.666
End of epoch 1 / 1      Time Taken: 130.22780919075012 sec
learning rate = 0.0000000
```

train.py x



```
#Translate AUS to RUS
!python ultrasound_test.py --dataroot /content/drive/MyDrive/ACSI/aus2rus --name aus2rus --model cycle_gan
```

```
#Translate RUS to AUS
!python ultrasound_test.py --dataroot /content/drive/MyDrive/ACSI/aus2rus --name aus2rus --model cycle_gan
```

```
# Original images
rus = Image.open('/content/drive/MyDrive/ACSI/aus2rus/testB/rotated_rezized_pacienteA1.jpg')
aus = Image.open('/content/drive/MyDrive/ACSI/aus2rus/testA/rezized_ct11-7.png')

# Fake images
fake_aus = Image.open('/content/drive/MyDrive/ACSI/results/fakeAUS/rotated_rezized_pacienteA1.png')
fake_rus = Image.open('/content/drive/MyDrive/Dataset_ACSI/US_acsi/aus2rus/aus2rus/results/fakeRUS/r

fig = plt.figure()

ax = fig.add_subplot(2, 2, 1, xticks=[], yticks=[], title='RUS')
plt.imshow(rus)

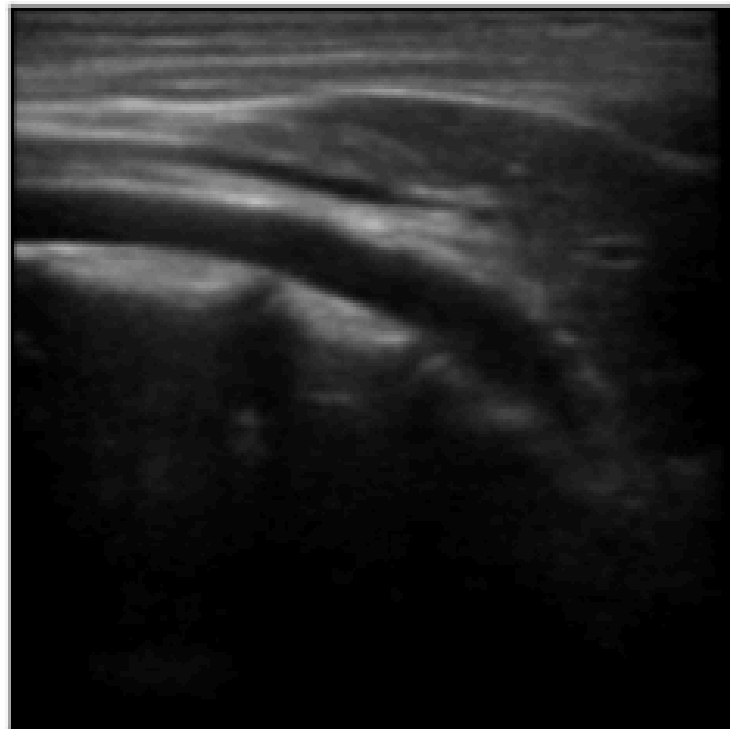
ax = fig.add_subplot(2, 2, 2, xticks=[], yticks=[], title='AUS')
plt.imshow(aus)

ax = fig.add_subplot(2, 2, 3, xticks=[], yticks=[], title='Fake_AUS')
plt.imshow(fake_aus)

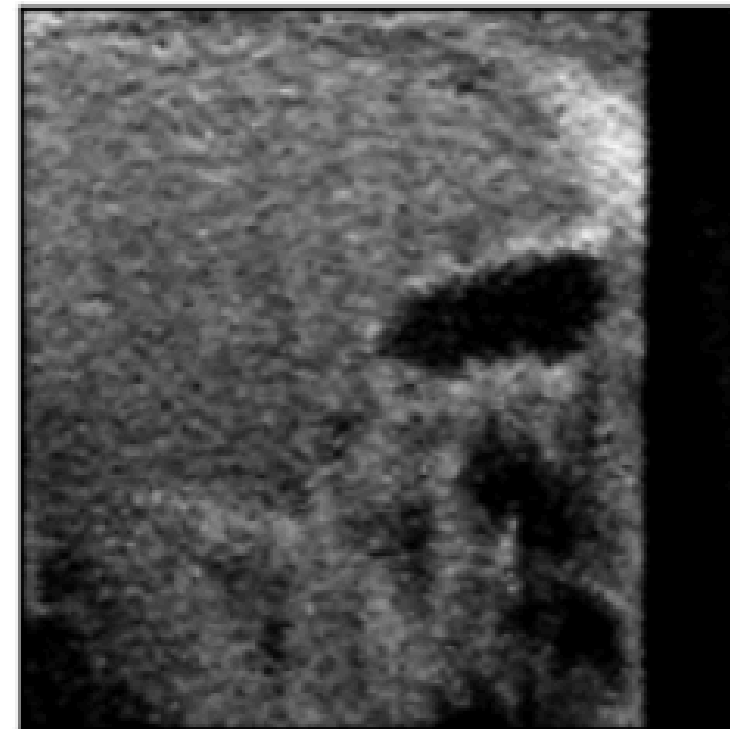
ax = fig.add_subplot(2, 2, 4, xticks=[], yticks=[], title='Fake_RUS')
plt.imshow(fake_rus)
```

```
<matplotlib.image.AxesImage at 0x7a70c0ac9de0>
```

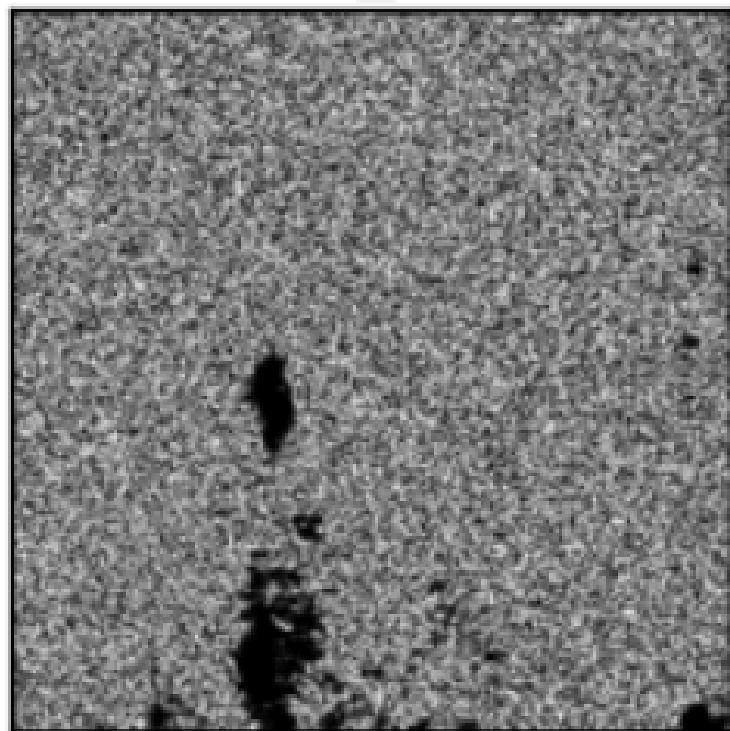
RUS



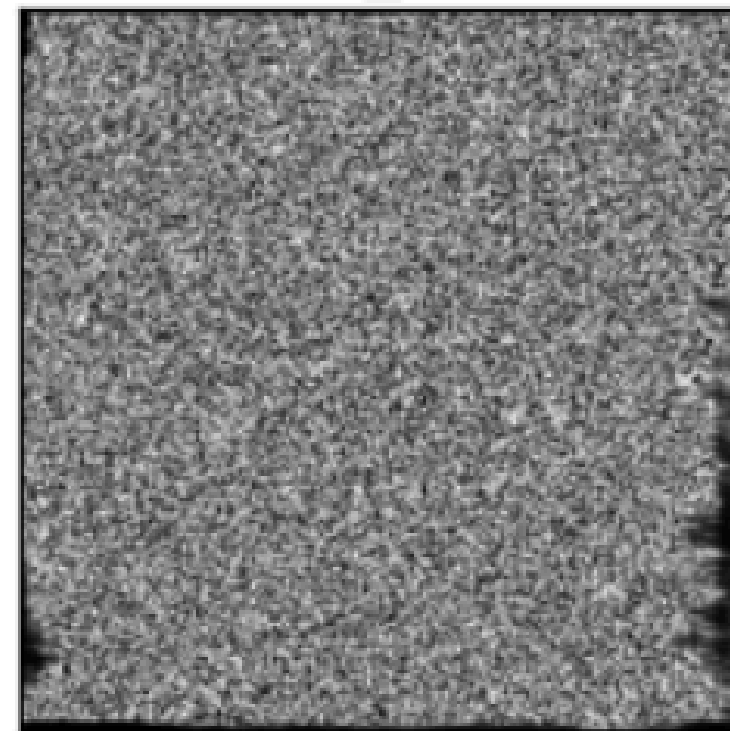
AUS



Fake_AUS



Fake_RUS



EXPLICACIÓN DEL CÓDIGO UTILIZADO

PREPARACIÓN DE IMÁGENES DE CT Y
SU TRADUCCIÓN A IMÁGENES DE US



PREPARACIÓN DE ARCHIVOS DE CT

01

DESCARGA DE
ARCHIVOS .NII

En este caso se tomaron
como referencia 5
archivos .NII

03

CROP DE IMÁGENES
EN FORMA DE
TRAPECIO

05

IMÁGENES
LISTAS PARA
PROBARSE EN EL
MODELO

02

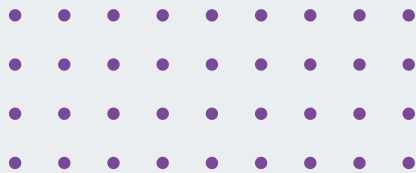
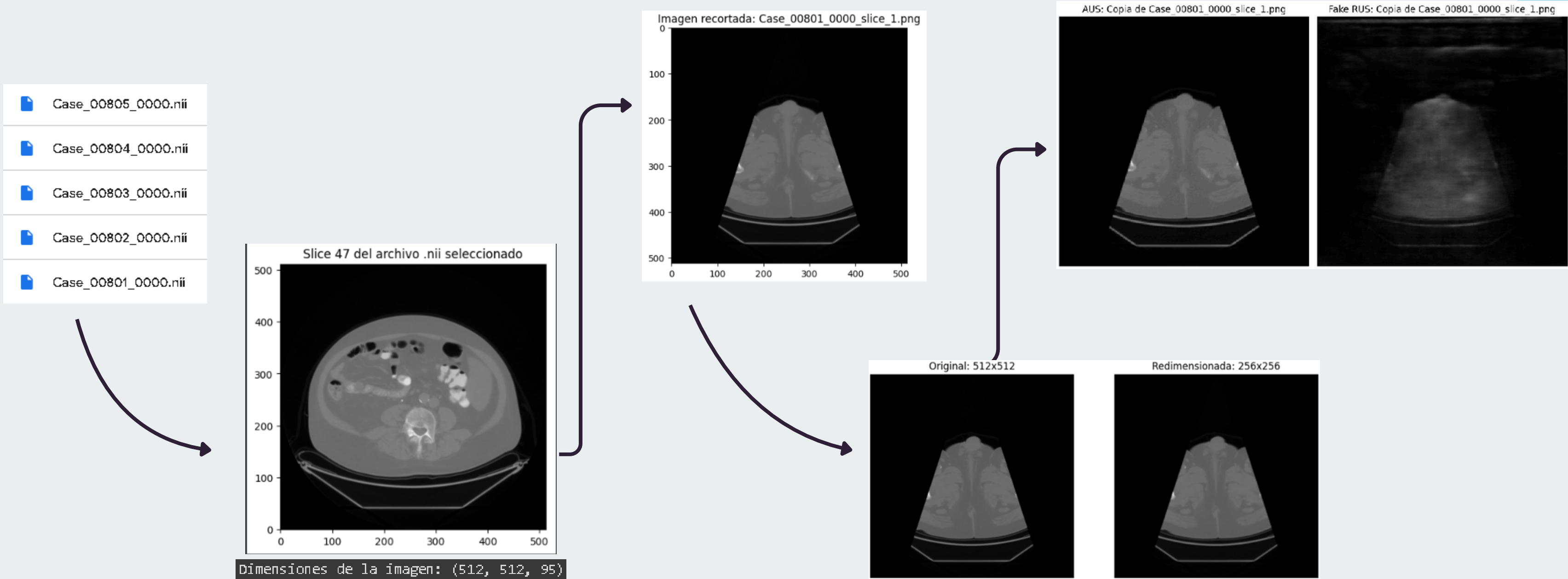
DESCOMPRESIÓN
DE ARCHIVOS .NII
A .PNG

04

REDIMENSION
AMIENTO DE
512X512 A
256X256



PREPARACIÓN DE ARCHIVOS DE CT



PREPARACIÓN DE ARCHIVOS DE CT

02

```
import os
import random
import nibabel as nib
import numpy as np
import matplotlib.pyplot as plt

# Define el directorio donde se encuentran los archivos .nii
nii_directory = '/content/drive/MyDrive/Dataset_ACSI/CT_acsi/'

# Listar todos los archivos .nii en el directorio
nii_files = [f for f in os.listdir(nii_directory) if f.endswith('.nii')]

# Seleccionar un archivo al azar
random_nii_file = random.choice(nii_files)
print(f"Archivo seleccionado: {random_nii_file}")

# Cargar el archivo .nii seleccionado
nii_path = os.path.join(nii_directory, random_nii_file)
img = nib.load(nii_path)

# Obtener los datos de la imagen en formato numpy
img_data = img.get_fdata()

# Mostrar información del archivo cargado
print(f"Dimensiones de la imagen: {img_data.shape}")

# Seleccionar una rebanada en el eje z para visualizar
slice_index = img_data.shape[2] // 2 # Rebanada central
slice_2d = img_data[:, :, slice_index]

# Mostrar la rebanada seleccionada
plt.imshow(slice_2d.T, cmap='gray', origin='lower')
plt.title(f'Slice {slice_index} del archivo .nii seleccionado')
plt.show()
```

```
import os
import nibabel as nib
import numpy as np
import matplotlib.pyplot as plt

# Define el directorio donde están los archivos .nii y donde se guardarán los .png
nii_directory = '/content/drive/MyDrive/Dataset_ACSI/CT_acsi/'
output_directory = '/content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1/'

# Crear el directorio de salida si no existe
os.makedirs(output_directory, exist_ok=True)

# Listar todos los archivos .nii en el directorio
nii_files = [f for f in os.listdir(nii_directory) if f.endswith('.nii')]

# Procesar cada archivo .nii
for nii_file in nii_files:
    # Cargar el archivo .nii
    nii_path = os.path.join(nii_directory, nii_file)
    img = nib.load(nii_path)

    # Obtener los datos de la imagen en formato numpy
    img_data = img.get_fdata()

    # Iterar sobre todas las rebanadas del volumen en el eje z
    for i in range(img_data.shape[2]):
        slice_2d = img_data[:, :, i]

        # Crear la ruta de salida para el archivo PNG
        output_file = os.path.join(output_directory, f"{nii_file.split('.')[0]}_slice_{i}.png")

        # Guardar la rebanada como archivo PNG
        plt.imsave(output_file, slice_2d.T, cmap='gray', origin='lower')

    print(f"Procesado {nii_file} y guardado en {output_directory}")

print("¡Todos los archivos .nii han sido procesados y convertidos a .png!")
```

```
Procesado Case_00801_0000.nii y guardado en /content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1/
Procesado Case_00805_0000.nii y guardado en /content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1/
Procesado Case_00804_0000.nii y guardado en /content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1/
Procesado Case_00803_0000.nii y guardado en /content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1/
Procesado Case_00802_0000.nii y guardado en /content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1/
¡Todos los archivos .nii han sido procesados y convertidos a .png!
```

PREPARACIÓN DE ARCHIVOS DE CT

03

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Ruta a la carpeta de las imágenes en Google Drive
image_folder = '/content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1/'

# Listar todos los archivos en la carpeta
image_files = [f for f in os.listdir(image_folder) if os.path.isfile(os.path.join(image_folder, f))]

# Función para aplicar la máscara en forma de trapecio
def trapezoid_mask(image, top_width_ratio=0.05, bottom_width_ratio=0.70, height_ratio=1.0):
    h, w = image.shape[:2]

    # Definir las proporciones del trapecio
    top_width = int(w * top_width_ratio) # Ancho de la parte superior del trapecio
    bottom_width = int(w * bottom_width_ratio) # Ancho de la parte inferior del trapecio
    height = int(h * height_ratio) # Altura del trapecio (en este caso, la altura completa de la imagen)

    # Coordenadas de los cuatro vértices del trapecio
    top_left = (w // 2 - top_width // 2, 0)
    top_right = (w // 2 + top_width // 2, 0)
    bottom_left = (w // 2 - bottom_width // 2, height)
    bottom_right = (w // 2 + bottom_width // 2, height)

    # Crear una máscara negra (de fondo)
    mask = np.zeros((h, w), dtype=np.uint8)

    # Definir el trapecio
    points = np.array([[top_left, top_right, bottom_right, bottom_left]], dtype=np.int32)

    # Rellenar el trapecio en la máscara
    cv2.fillPoly(mask, points, 255)

    # Aplicar la máscara a la imagen
    result = cv2.bitwise_and(image, image, mask=mask)

    return result
```

```
# Carpeta de salida para las imágenes recortadas
output_folder = '/content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1_cropped/'
os.makedirs(output_folder, exist_ok=True)

# Aplicar el recorte tipo trapecio a todas las imágenes
for image_file in image_files:
    image_path = os.path.join(image_folder, image_file)
    try:
        # Cargar la imagen
        image = cv2.imread(image_path)

        if image is None:
            print(f"Error al cargar la imagen {image_file}.")
            continue

        # Aplicar el recorte tipo trapecio
        cropped_image = trapezoid_mask(image)

        # Guardar la nueva imagen
        output_path = os.path.join(output_folder, image_file)
        cv2.imwrite(output_path, cropped_image)

        # Mostrar la imagen recortada (opcional)
        plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
        plt.title(f"Imagen recortada: {image_file}")
        plt.show()

    except Exception as e:
        print(f"Error al procesar {image_file}: {e}")
```

PREPARACIÓN DE ARCHIVOS DE CT

04

```
#from PIL import Image
import os

# Ruta de la carpeta de imágenes originales de 512x512
ruta_origen = '/content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1_cropped'

# Ruta de la carpeta destino para guardar las imágenes redimensionadas (puedes cambiarla)
ruta_destino = '/content/drive/MyDrive/Dataset_ACSI/CT_acsi_proce_p1_cropped_256x256'

# Crear la carpeta destino si no existe
if not os.path.exists(ruta_destino):
    os.makedirs(ruta_destino)

# Obtener una lista de archivos en la carpeta
imagenes = os.listdir(ruta_origen)

# Filtrar por archivos de imagen comunes (puedes ajustar las extensiones según sea necesario)
imagenes = [img for img in imagenes if img.endswith(('.png', '.jpg', '.jpeg'))]

# Tamaño de salida deseado
nuevo_tamano = (256, 256)

# Redimensionar todas las imágenes
for imagen in imagenes:
    # Ruta completa de la imagen original
    ruta_imagen_original = os.path.join(ruta_origen, imagen)

    # Ruta donde se guardará la imagen redimensionada
    ruta_imagen_nueva = os.path.join(ruta_destino, imagen)

    # Abrir la imagen original
    with Image.open(ruta_imagen_original) as img:
        # Redimensionar la imagen
        img_redimensionada = img.resize(nuevo_tamano)

        # Guardar la imagen redimensionada en la ruta de destino
        img_redimensionada.save(ruta_imagen_nueva)

    print(f'Imagen {imagen} redimensionada y guardada en {ruta_imagen_nueva}')

print('Todas las imágenes han sido redimensionadas correctamente.')
```

TRADUCCIÓN DE IMÁGENES CT A US

01 PROBAR LA TRADUCCIÓN CON UNA IMÁGEN CT ALEATORIA

```
import os
import random
from PIL import Image
import matplotlib.pyplot as plt

# Define los directorios para imágenes originales y generadas
dir_aus = '/content/drive/MyDrive/Dataset_ACSI/US_acsi/aus2rus/aus2rus/testA_CT/'
dir_fake_rus = '/content/drive/MyDrive/Dataset_ACSI/US_acsi/aus2rus/aus2rus/results/fakeRUS_CT_Test/'

# Selecciona una imagen aleatoria de cada directorio
aus_image = random.choice([f for f in os.listdir(dir_aus) if f.endswith('.jpg') or f.endswith('.png')])
fake_rus_image = aus_image.replace('.jpg', '.png') # Suponiendo que los nombres son iguales

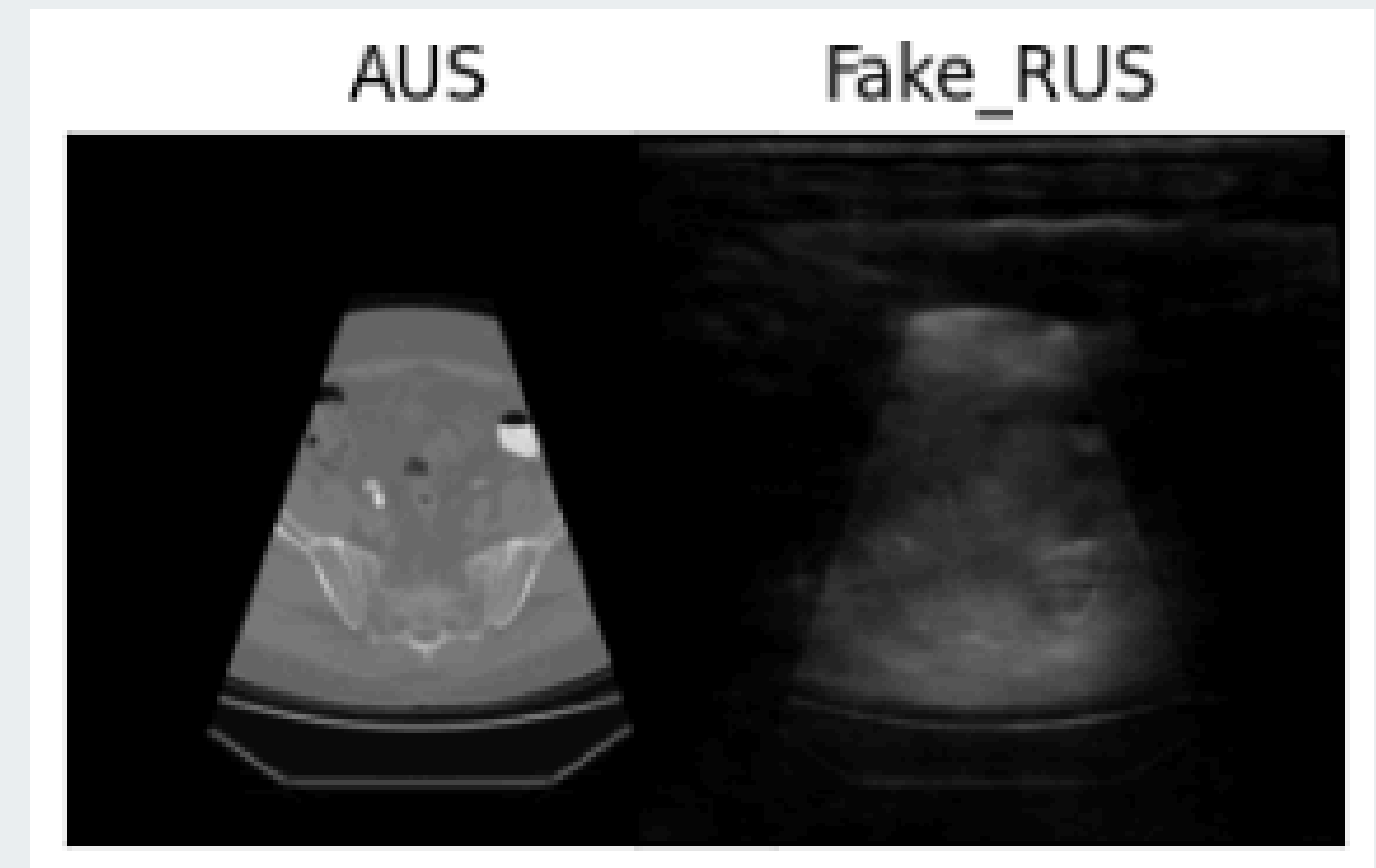
# Carga las imágenes
aus = Image.open(os.path.join(dir_aus, aus_image))
fake_rus = Image.open(os.path.join(dir_fake_rus, fake_rus_image))

# Crear la figura y agregar los subplots
fig = plt.figure()

ax = fig.add_subplot(2, 1, 1, xticks=[], yticks=[], title='AUS')
plt.imshow(aus)

ax = fig.add_subplot(2, 2, 2, xticks=[], yticks=[], title='Fake_RUS')
plt.imshow(fake_rus)

# Mostrar las imágenes
plt.show()
```



TRADUCCIÓN DE IMÁGENES CT A US

02 TRADUCCIÓN LAS IMÁGENES DE CT A US

```
import os
from PIL import Image
import matplotlib.pyplot as plt

# Define los directorios para imágenes originales y generadas
dir_aus = '/content/drive/MyDrive/Dataset_ACSI/US_acsi/aus2rus/aus2rus/testA_CT/'
dir_fake_rus = '/content/drive/MyDrive/Dataset_ACSI/US_acsi/aus2rus/aus2rus/results/fakeRUS_CT_Test/'

# Obtén la lista de imágenes en el directorio de testA_CT
aus_images = [f for f in os.listdir(dir_aus) if f.endswith('.jpg') or f.endswith('.png')]

# Crear una figura con subplots ajustables
fig, axes = plt.subplots(len(aus_images), 2, figsize=(10, len(aus_images) * 5))

# Recorrer todas las imágenes y mostrarlas en la figura
for i, aus_image in enumerate(aus_images):
    # Cargar la imagen original
    aus = Image.open(os.path.join(dir_aus, aus_image))

    # Suponiendo que los nombres son iguales, cambiar la extensión si es necesario
    fake_rus_image = aus_image.replace('.jpg', '.png')
    fake_rus_path = os.path.join(dir_fake_rus, fake_rus_image)

    # Cargar la imagen generada solo si existe en el directorio de imágenes generadas
    if os.path.exists(fake_rus_path):
        fake_rus = Image.open(fake_rus_path)
    else:
        fake_rus = None
```

```
# Mostrar la imagen original
axes[i, 0].imshow(aus)
axes[i, 0].set_title(f'AUS: {aus_image}')
axes[i, 0].axis('off')

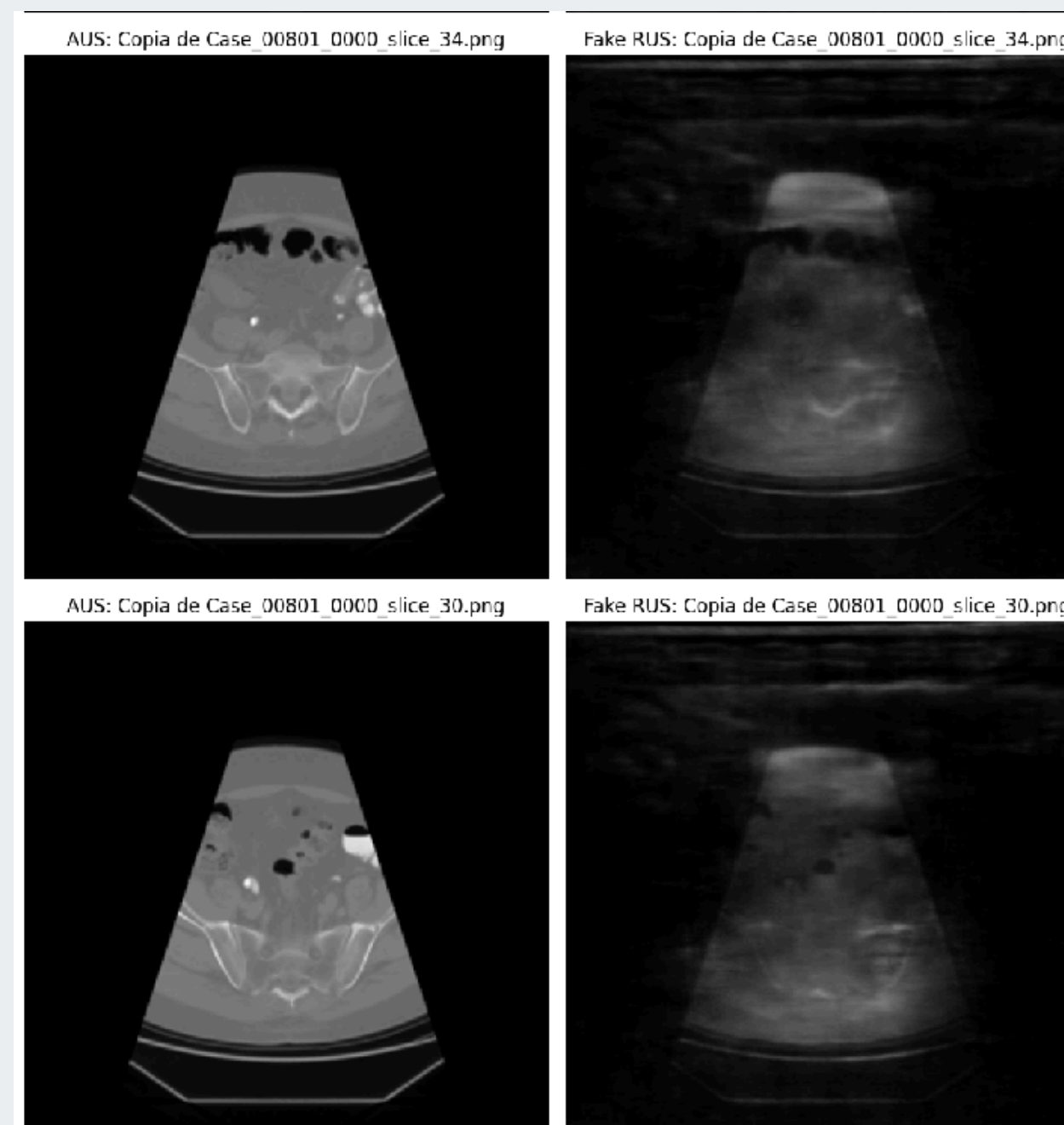
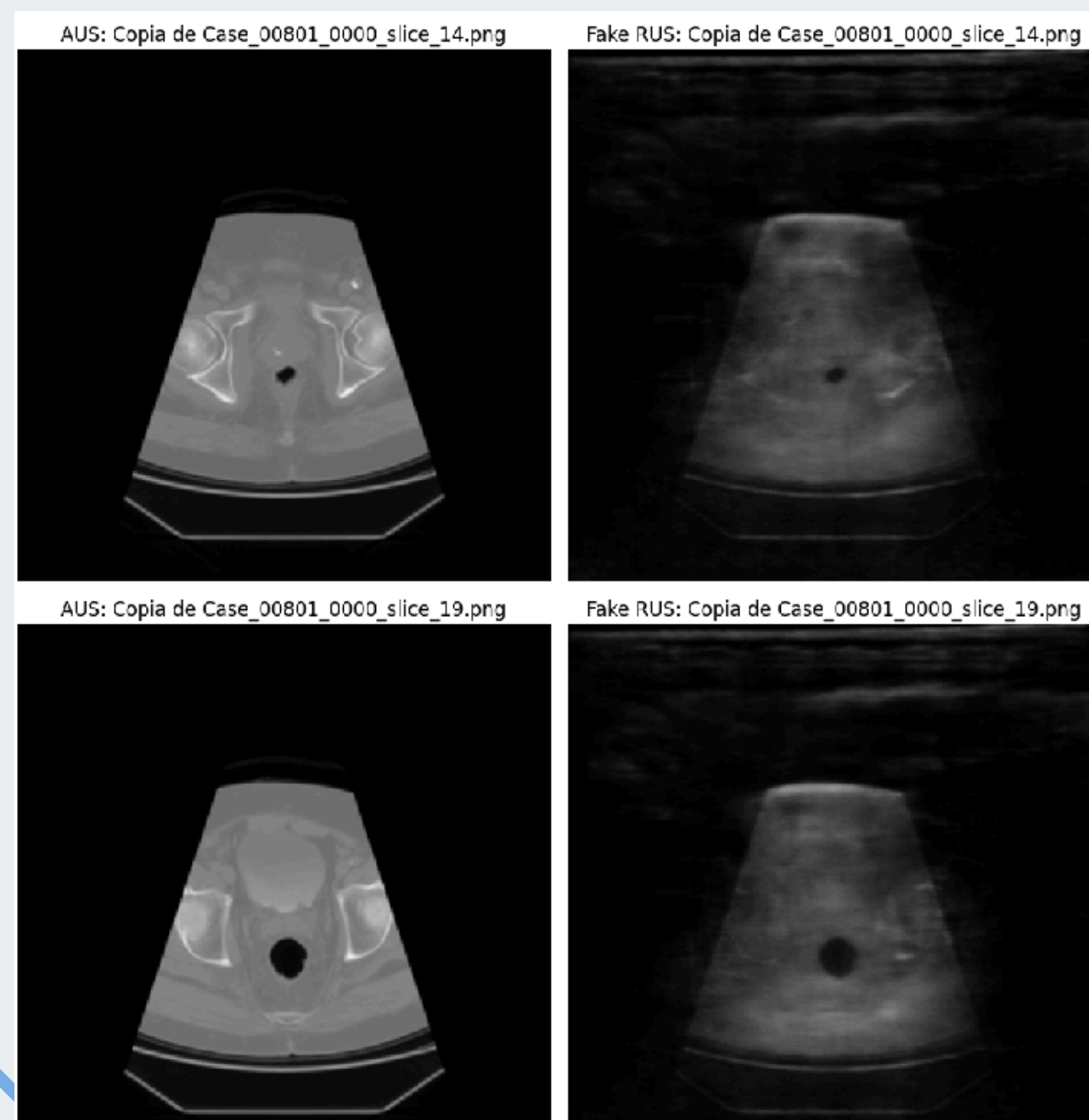
# Mostrar la imagen generada
if fake_rus is not None:
    axes[i, 1].imshow(fake_rus)
    axes[i, 1].set_title(f'Fake RUS: {fake_rus_image}')
else:
    axes[i, 1].set_title(f'Fake RUS: No image found for {aus_image}')

    axes[i, 1].axis('off')

# Ajustar el espaciado de la figura
plt.tight_layout()
plt.show()
```


TRADUCCIÓN DE IMÁGENES CT A US

02 TRADUCIR LAS IMÁGENES DE CT A US



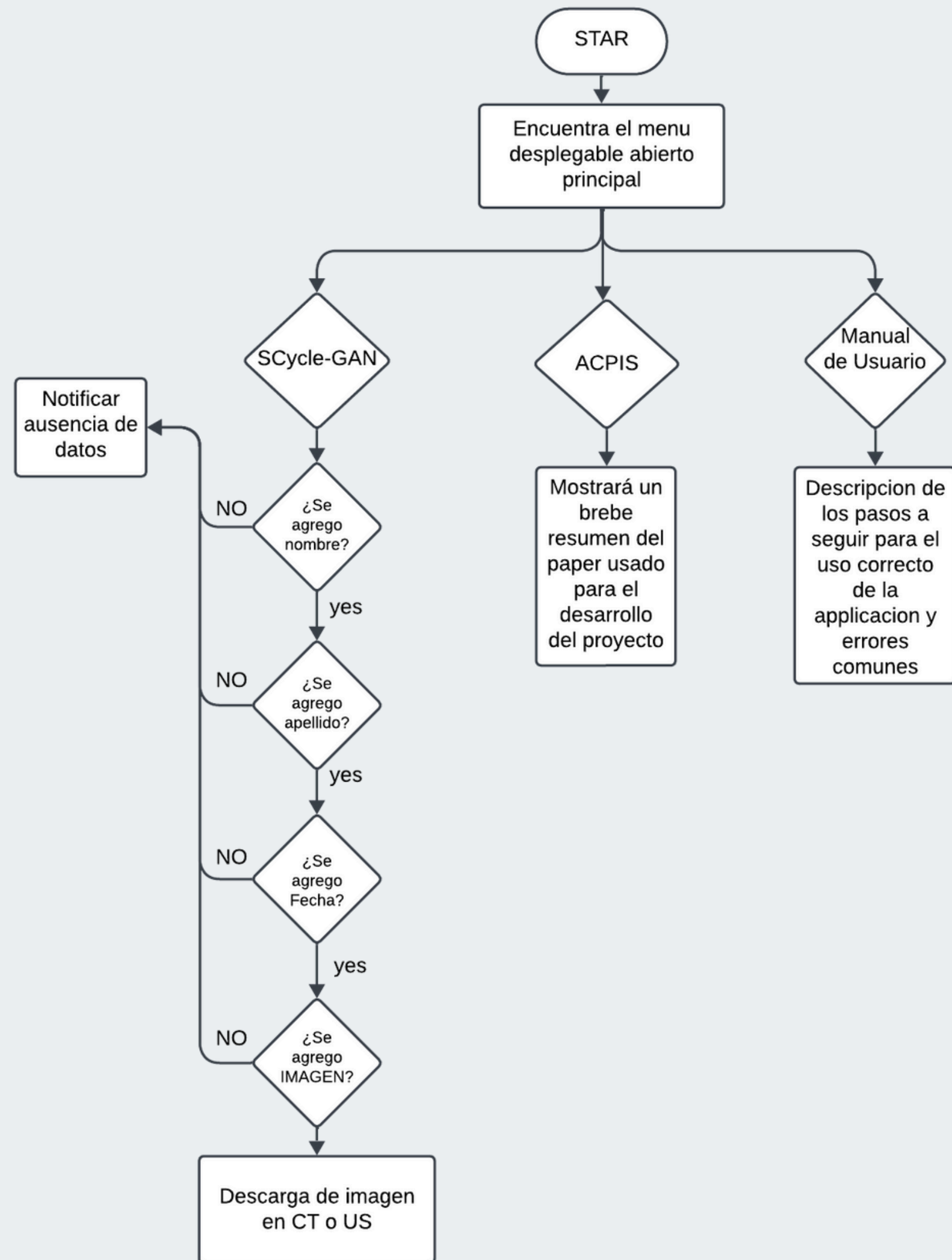
AVANCE DEL FRONTEND

EN BASE A STREAMLIT

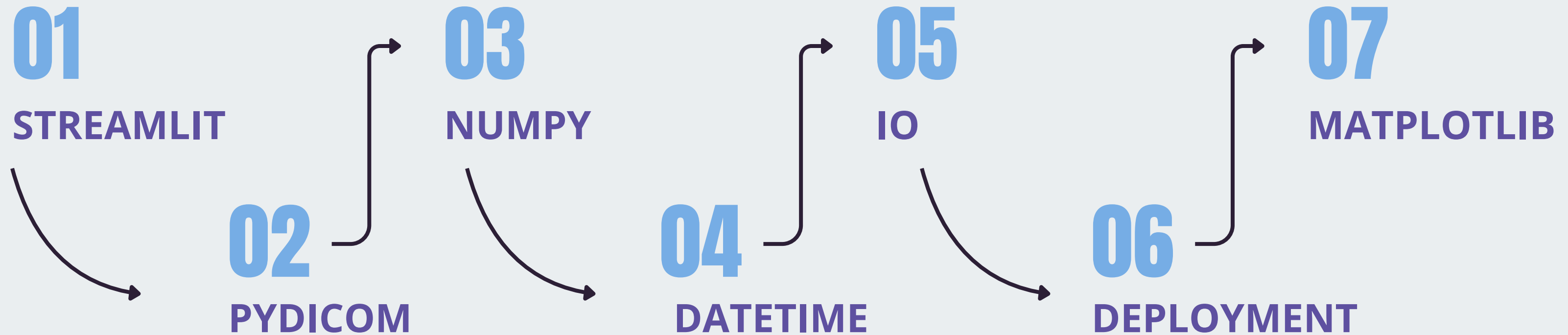


IDEA DE FLUJO

Actual mente la idea de flujo se mantiene simple para un control mas intuitivo por parte del usuario



CODIGO Y LIBRERIAS



A 4x10 grid of purple dots, consisting of 4 rows and 10 columns, totaling 40 dots.

[illegible]

CODIGO Y LIBRERIAS




MEDGAN


Disclaimer: Este programa ha sido diseñado con fines académicos por lo que no se recomienda el uso de las imágenes generadas como guía o resultados de algún tipo para el diagnóstico.


```
# Agregar el disclaimer al inicio de la aplicación
st.info("**Disclaimer:** Este programa ha sido diseñado con fines académicos por lo que no se \
recomienda el uso de las imágenes generadas como guía o resultados de algún tipo para el diagnóstico.")
```


CODIGO Y LIBRERIAS



 Menú

 SCYCLE-GAN

 **ACPIs**

 Manual de Usuario

MEDGAN

Disclaimer: Este programa ha sido diseñado con fines académicos por lo que no se recomienda el uso de las imágenes generadas como guía o resultados de algún tipo para el diagnóstico.

S-CycleGAN: Semantic Segmentation Enhanced CT-Ultrasound Image-to-Image Translation for Robotic Ultrasonography

Introducción

Referencias

Paper base: <https://arxiv.org/html/2406.01191v2>

Repositorio referencial: <https://github.com/yhsong98/ct-us-i2i-translation>

Dataset sugerido: <https://www.kaggle.com/datasets/ignaciorlando/ussimandsegm/data>

CODIGO Y LIBRERIAS



Menú

SCYCLE-GAN

ACPIS

Manual de Usuario


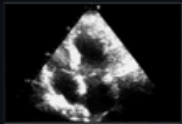
MEDGAN

Disclaimer: Este programa ha sido diseñado con fines académicos por lo que no se recomienda el uso de las imágenes generadas como guía o resultados de algún tipo para el diagnóstico.

Manual de Usuario

¿Para qué sirve?

Esta es una aplicación de la cual se encarga de transformar imágenes de CT a US y viceversa



REDIRECCION A LA APP

>

MEDGAN

Disclaimer: Este programa ha sido diseñado con fines académicos por lo que no se recomienda el uso de las imágenes generadas como guía o resultados de

FUTURO DEL FRONTEND



Subir Imagenes generadas a un repositorio

Deseamos experiencia del usuario sea optima y tenga todos sus datos almacenados en un solo lugar por lo que planeamos que se permita la conectividad con drive

Link de la carpeta drive donde desee descargar

<https://drive.google.com/drive/folders/1dY4DacdvUpc12AYOipNPpaFxRLvTYPqQ?usp=sharing>
Press Enter to apply

FUTURO DEL FRONTEND



Subir Imagenes generadas a un repositorio

Esto sera posible utilizando algunas liberias extras que nos permitiran conectar DRIVE con streamlit

- 1) PYDRIVE**
- 2) GOOGLE-AUTH**
- 3) GOOGLE-AUTH-OAUTHLIB**
- 4) GOOGLE-AUTH-HTTPLIB2**



**THANK
YOU**