Generate synthetic grayscale BUS images with tumors

All these codes are tested in Python 3.10.19, Tensorflow-gpu 2.10.0 and numpy 1.25.2.

1 Intitial Defination

Import the necessary libraries, define the variables, and define a function for the generator

```
import os
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
# intial defination
IMG_SIZE = 256
LATENT_DIM = 256
# The Generator
def build_generator():
    latent_inputs = layers.Input(shape=(LATENT_DIM,))
    start_res = IMG_SIZE // 8 # 32x32
    x = layers.Dense(start_res * start_res * 128)(latent_inputs)
    x = layers.Reshape((start_res, start_res, 128))(x)
    x = layers.Conv2DTranspose(128, 3, strides=2, padding='same')(x)
    x = layers.LeakyReLU()(x)
    x = layers.Conv2DTranspose(64, 3, strides=2, padding='same')(x)
    x = layers.LeakyReLU()(x)
    x = layers.Conv2DTranspose(32, 3, strides=2, padding='same')(x)
    x = layers.LeakyReLU()(x)
    output = layers.Conv2D(2, 3, activation='sigmoid', padding='same')(x)
    return models.Model(latent_inputs, output, name="generator")
generator = build_generator()
```

2 Load weights

The trained weights for the generator can be found in https://github.com/MohanBhandari/SynBreast-US/blob/main/generator_weights.weights.h5. One can download this and load the weights. It is publicly available

```
generator.load_weights("generator_weights.weights.h5")
```

3 Generate the samples

One can generate as many samples as needed based on the values of *num_samples*. Clinical validation is highly recommended, and we do not take responsibility for any diagnostic or clinical decisions made on the basis of the generated synthetic samples without independent medical oversight.

```
def generate_novel_samples(model_generator, num_samples=100):
    random_latent_vectors = tf.random.normal(shape=(num_samples,
        LATENT_DIM))
```

```
    synthetic_output = model_generator.predict(random_latent_vectors,
        verbose=0)
    syn_gray = synthetic_output[..., 0]
    return syn_gray
generated_images= generate_novel_samples(generator, num_samples=100)
```

4 Save the images

The generated images are now saved in drive.

```
import os
import numpy as np
from PIL import Image
def save_synthetic_data(gen_images, base_dir="Synthetic_Dataset"):
    # 1. Create the folder structure
    gray_dir = os.path.join(base_dir, "gray")
    os.makedirs(gray_dir, exist_ok=True)
    print(f"Saving {len(gen_images)} samples to {base_dir}...")
    for i in range(len(gen_images)):
        # 2. Convert arrays to 8-bit images (0-255)
        img_array = (gen_images[i] * 255).astype(np.uint8)
        # 3. Create PIL Image objects
        img_pil = Image.fromarray(img_array)
        # 4. Save with matching filenames
        img_pil.save(os.path.join(gray_dir, f"sample_{i:04d}.png"))
    print("Done!")
save_synthetic_data(generated_images)
```