Lab 4: Conditional Generative Adversarial Network (cGAN)

1 Introduction

Today, you will be building and training a Conditional Generative Adversarial Network (cGAN) to generate images of human faces, conditioned on specific facial attributes. You will be using the CelebA dataset.

2 Instructions

2.1 Dataset and Preprocessing

1. **Dataset Loading:** Load the CelebA dataset using the `torchvision.datasets` module in PyTorch. load the training split of the dataset.
2. **Image Transformations:** Apply the following image transformations using `torchvision.transforms`:
   - Resize all images to a consistent size of 64x64 pixels.
   - Crop the center of each image to ensure the final size is 64x64.
   - Normalize the pixel values of the images to the range [-1, 1].

2.2 Model Architecture

1. **Generator Network (G):** Design a generator network with the following specifications:
   - **Input:** The generator should take two inputs:
     (a) A latent vector, **z**, of a specified dimension (e.g., 100).
     (b) An attribute vector, **a**, representing the desired facial attributes. This vector should have the same dimension as the number of attributes in the CelebA dataset (40). The values in this vector should represent the presence (1) or absence (0) of each attribute.
   - **Architecture:** Use a series of transposed convolutional layers (`nn.ConvTranspose2d`) to upsample the input (concatenated latent and attribute vectors) to the desired output image size (64x64x3) or you can use MLPs.
2. **Discriminator Network (D):** Design a discriminator network with the following specifications:
   - **Input:** The discriminator should take two inputs:
     (a) An image, either a real image from the dataset or a generated image from the generator.
     (b) An attribute vector, **a**, representing the facial attributes.
   - **Architecture:** Use a series of convolutional layers to downsample the input image or you can use MLPs.
   - **Output:** The final layer should be a single output neuron with a Sigmoid activation function, representing the probability that the input image is real (given the provided attributes).
   - **Attribute Handling (Important):** The discriminator must be conditioned on the attributes. A recommended approach is:
     (a) Reshape the attribute vector into a "feature map" with dimensions (batch_size, num_attributes, 1, 1).

(b) Expand this feature map to have the same spatial dimensions (height and width) as the image being processed by the discriminator.

(c) Concatenate the expanded attribute map with the image along the channel dimension. This provides the discriminator with spatially-aware attribute information.

2.3 Evaluation and Visualization

1. **Image Generation:** After training, generate a set of sample images using the trained generator.
2. **Attribute Control:** Test the conditional nature of your cGAN by generating specific images. For example, generate images of faces with and without the Smiling attribute.
3. **Loss Plotting:** Plot the generator and discriminator losses over the training progress.
4. **Save Images:** Save the generated images to files after each epoch to track the evolution of the generated images during training.