USN NUMBER:1RVU22BSC110
NAME: YASHWANT RAJ

Ex No: 7 Date: 18-09-24	Generative Adversarial Networks (GAN) Using MNIST Dataset

Objective:

The aim of this project was to create a Generative Adversarial Network (GAN) that generates new images resembling handwritten digits based on the MNIST dataset.

Descriptions:

A **Generative Adversarial Network (GAN)** consists of two neural networks: a **generator** and a **discriminator**. These two models are trained simultaneously. The generator tries to create realistic images, while the discriminator attempts to distinguish real images from fake ones.

- **Generator:** This network generates new images from random noise.
- **Discriminator:** This network evaluates the authenticity of the images, outputting a probability that the image is real (from the dataset) or fake (generated by the generator).

The **MNIST dataset** contains 28x28 grayscale images of handwritten digits from 0 to 9. The dataset includes 60,000 training images and 10,000 test images.

Methodology:

1. Dataset Preparation:

The MNIST dataset was loaded and preprocessed. Each image was reshaped to 28x28 pixels and normalized to have pixel values between -1 and 1 for better convergence during training.

2. Model Architecture:

- **Generator:** The generator model was designed to convert a random noise vector of size 100 into a 28x28 image. It used **transposed convolutions**, **batch normalization**, and **LeakyReLU** activations.

USN NUMBER:1RVU22BSC110
NAME: YASHWANT RAJ

```python
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)  # Note: None is the batch size

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    return model
```

- **Discriminator:** The discriminator took in a 28x28 image as input and classified it as real or fake. It used **convolutions**, **dropout**, and **LeakyReLU** activations to prevent overfitting.

```python
def make_discriminator_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                            input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model
```

3. Loss Functions:

The loss functions were based on binary cross-entropy, where the discriminator tried to correctly classify real and fake images, and the generator tried to "fool" the discriminator into classifying fake images as real.

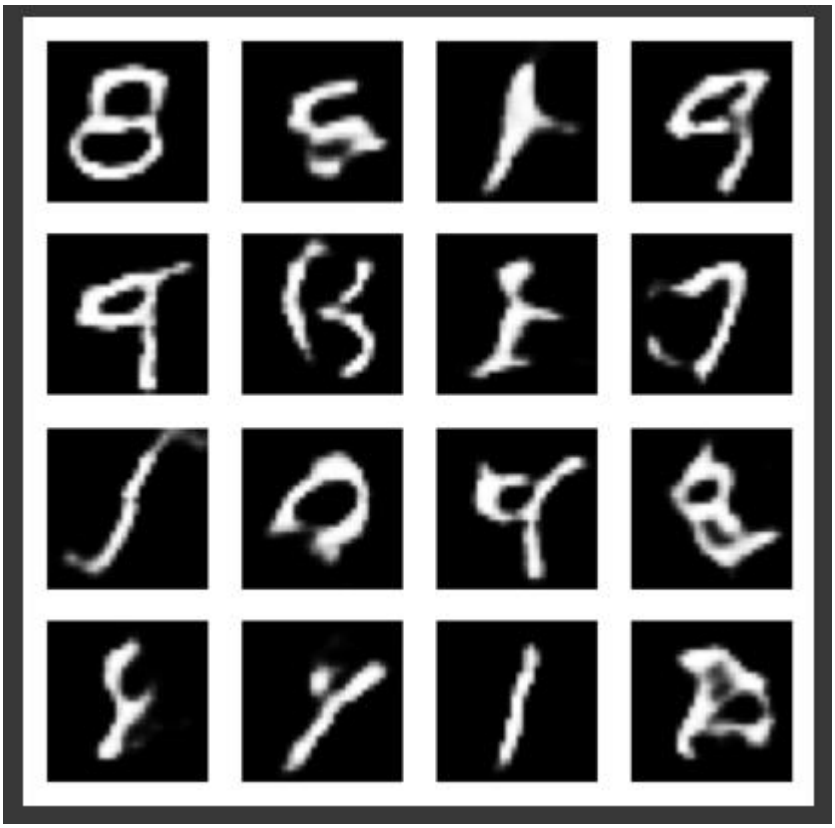
- **Generator Loss:** This measured how well the generator could fool the discriminator.
- **Discriminator Loss:** This measured how well the discriminator distinguished

between real and fake images.

4. Training:

The GAN was trained for 50 epochs. For each epoch:

- The generator created a batch of fake images.
- The discriminator classified both real and fake images.
- The generator and discriminator were updated based on their respective losses.



Results:

- **Generated Images:** The generator was able to produce images resembling handwritten digits from random noise. Initially, the generated images were indistinguishable from random noise. However, as training progressed, the images became clearer and more like digits from the MNIST dataset.

- **Training Time:** Each epoch took approximately 30 seconds to complete, and the model trained for 50 epochs.

- **GIF of Generated Images:** A GIF was created to visualize the progression of image generation across different epochs.

USN NUMBER:1RVU22BSC110
NAME: YASHWANT RAJ

GitHub Link: https://github.com/Yashr22/Lab-7-GANs.git