12 Implement a Deep convolutional GAN to generate complex color image.

Aim:

Train a Deep convolutional Generative Adversial network to generate high quality, realistic color images by learning the distribution of a dataset of color images

Algorithm:

* Prepare dataset, load and preprocess images.
* Define Models:

  Generator: transposed convolutional network that upsamples a latent vector

  Discrimination: convolutional network that down samples the image to scalar logit.

* Initialize weights.
* Training loop,

  Update D: Maximize $\log(D(x)) + \log(1 - D(G(z)))$

  using BCE loss

  Train on a batch of real images

  Train on a batch of fake images produced by G(z)

  Update G: minimize BCE with logits loss $(D(G(z)), 1)$

* Logging & checkpoints: save samples periodically and checkpoint models weights per epoch

Pseudocode:

```
SET hyperParameters (nz, ngf, ndf, nc, image_size,
    (r, beta1, batch_size, epochs)
Prepare dataset with transforms:
  Resize (image_size), CenterCrop, ToTensor(),
  Normalize (0.5)*3, (0.5)*3)
Create DataLoader (dataset, batch_size, shuffle=True)
Define Generator G:
    Input : z (nz x 1 x 1)
    sequence of convTranspose2d layers to upsample
  to image_size
  BatchNorm2d + ReLU between layers
  Final layer: convTranspose2d -> Tanh -> output
  image (nc x H x w)
Define Discriminator D:
  Input : image (nc x H x w)
  Sequence of conv2d layers to downsample to
  4 x 4
  BatchNorm2d + LeakyReLU between layers
  Final layer: conv 2d -> output logit
  Flatten logit to shape (batch,)
Initialize weights (normal, mean=0, std=0.02)
set optimizers: Adam (G.Parameters(), (r, betas =
                (beta1, 0.999)
fixed_noise = random tensor (sample_size, nz, 1, 1)
```

```
For epoch in range(1, epochs):
    For each batch real_images in dataloader:
        #1. update D
        D.zero_grad()
        labels_real = ones(batch_size)
        output_real = D(real images)
        loss_real = BCE with logits loss(output_real, labels_re
        loss_real.backward()
        noise = random normal(batch_size, nz, 1, 1)
        fake imges = G(noise)
        labels_fake = zeros(batch_size)
        output_fake = D(fake_images.detach())
        loss_fake.backword()
        optimizerD.stepD()
```

Observation:

* The generator gradually learned to Produce realistic color images
* The discrimination improved in distinguishing real and fake images
* Generator images were visually similar to the training dataset after sufficient epochs

Result:

The DCGAN successfully generated realistic looking color images, Proving the effectiveness of Adversial learning.

GAN Architecture

Output:

Epoch [1/10] D Loss : 1.0992 , G Loss : 0.5678

Epoch [2/10] D Loss : 0.506 , G Loss : 2.7113

Epoch [3/10] D Loss : 1.2431 , G Loss : 3.9827

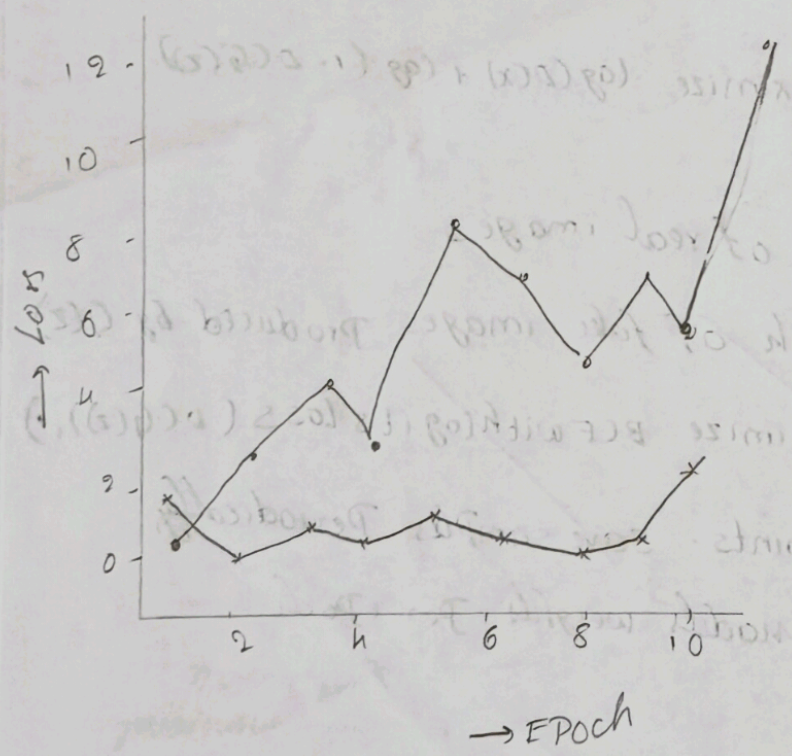
Epoch [4/10] D Loss : 0.4224 , G Loss : 2.2684

⋮

Epoch [8/10] D Loss : 0.0043 , G Loss : 6.3371

Epoch [9/10] D Loss : 0.1380 , G Loss : 4.73

Epoch [10/10] D Loss : 2.3294 , G Loss : 1.403

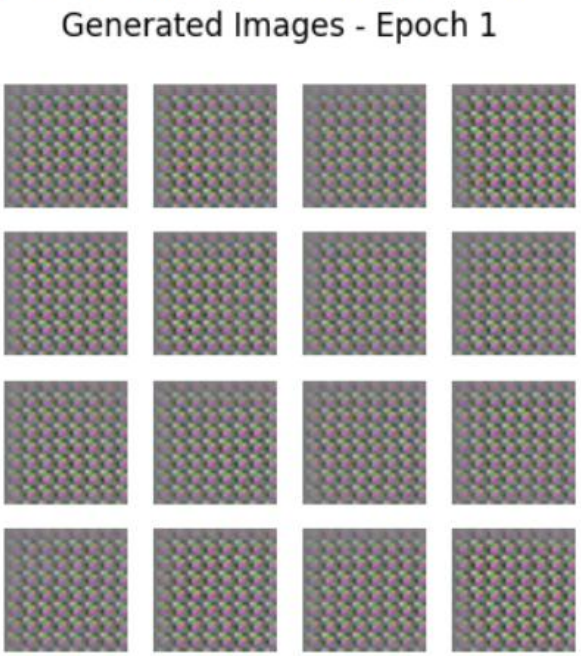


```python
# ✅ Lab 12 (Fast Version): DCGAN to Generate Color Images (CIFAR-10)
import tensorflow as tf
from tensorflow.keras import layers
import matplotlib.pyplot as plt
import numpy as np
import time

# 1️⃣ Load & Preprocess CIFAR-10 (subset for speed)
(x_train, _), (_, _) = tf.keras.datasets.cifar10.load_data()
x_train = (x_train.astype("float32") - 127.5) / 127.5  # normalize to [-1,1]
x_train = x_train[:5000]  # only 5k images for faster training

BUFFER_SIZE = 5000
BATCH_SIZE = 32
train_dataset = tf.data.Dataset.from_tensor_slices(x_train).shuffle(BUFFER_SIZE).batch(BATCH_SIZE)

# 2️⃣ Generator
def build_generator():
    model = tf.keras.Sequential([
        layers.Input(shape=(100,)),
        layers.Dense(8*8*128, use_bias=False),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
        layers.Reshape((8, 8, 128)),
        layers.Conv2DTranspose(64, (5,5), strides=(2,2), padding="same", use_bias=False),
        layers.BatchNormalization(),
        layers.LeakyReLU(),
```

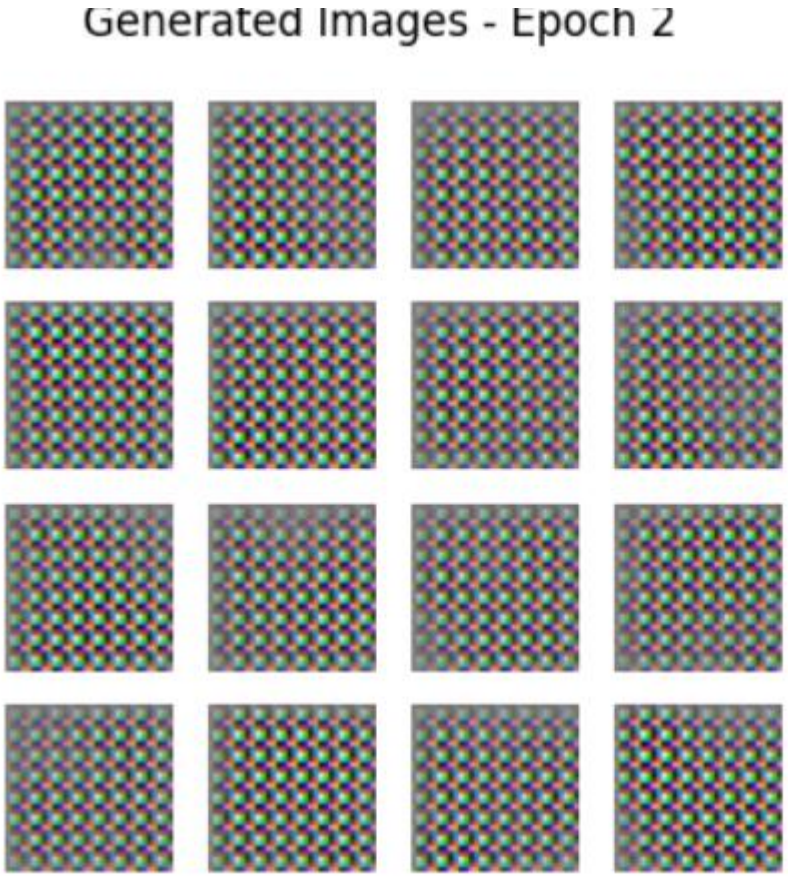
```python
# 3️⃣ Discriminator
def build_discriminator():
    model = tf.keras.Sequential([
        layers.Input(shape=(32, 32, 3)),
        layers.Conv2D(64, (5,5), strides=(2,2), padding="same"),
        layers.LeakyReLU(),
        layers.Dropout(0.3),
        layers.Conv2D(128, (5,5), strides=(2,2), padding="same"),
        layers.LeakyReLU(),
        layers.Dropout(0.3),
        layers.Flatten(),
        layers.Dense(1)
    ])
    return model

generator = build_generator()
discriminator = build_discriminator()

# 4️⃣ Loss + Optimizers
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
gen_optimizer = tf.keras.optimizers.Adam(1e-4)
disc_optimizer = tf.keras.optimizers.Adam(1e-4)

@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, 100])
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)
```

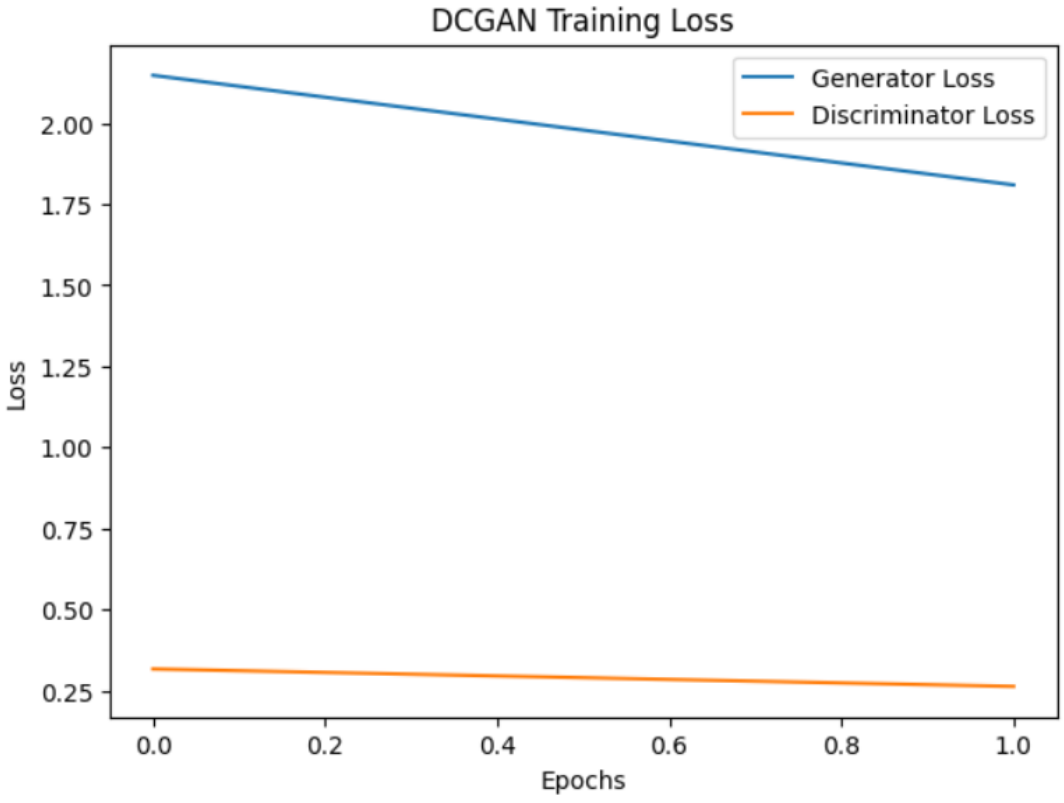
```python
# 5️⃣ Training (Quick Version)
EPOCHS = 2
seed = tf.random.normal([16, 100])
gen_losses, disc_losses = [], []

for epoch in range(EPOCHS):
    start = time.time()
    for image_batch in train_dataset.take(100):  # only 100 batches for speed
        g_loss, d_loss = train_step(image_batch)
    gen_losses.append(g_loss.numpy())
    disc_losses.append(d_loss.numpy())

    print(f"Epoch {epoch+1}/{EPOCHS} | Gen Loss: {g_loss:.4f} | Disc Loss: {d_loss:.4f} | Time: {time.time()-start:.2f}s")

    # Show generated samples
    predictions = generator(seed, training=False)
    fig = plt.figure(figsize=(4, 4))
    for i in range(predictions.shape[0]):
        plt.subplot(4, 4, i + 1)
        plt.imshow((predictions[i] * 0.5 + 0.5))
        plt.axis("off")
    plt.suptitle(f"Generated Images - Epoch {epoch+1}")
    plt.show()

# 6️⃣ Plot Loss Graph
plt.figure(figsize=(7,5))
plt.plot(gen_losses, label="Generator Loss")
plt.plot(disc_losses, label="Discriminator Loss")
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
Epoch 1/2 | Gen Loss: 2.1477 | Disc Loss: 0.3164 | Time: 6.95s
```




```
Epoch 2/2 | Gen Loss: 1.8096 | Disc Loss: 0.2626 | Time: 0.75s
```


Generated Images - Epoch 2


DCGAN Training Loss
Generator Loss
Discriminator Loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
Loss
0.0
0.2
0.4
0.6
0.8
1.0
Epochs