RA2311047010040

Index

மரம் வளர்ப்போம்
மழை வளம் காப்போம்

Superior® Note Books

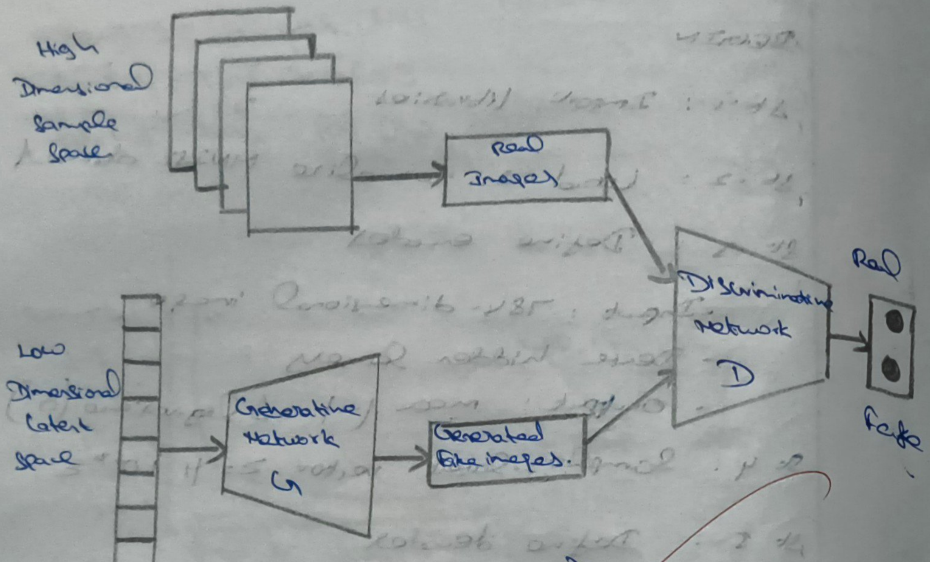
Name : Vishwa S

Subject : DL

Std. : .... Sec. : .... Roll No. ....

School : ....

No.	Date	Title	Marks	Signature
1	24/7/25	Exploring the DL platforms.		
2	31/7/25	Implement a classifier using open source dataset.		
3	7/8/25	Study the classifier with respect to statistical parameters.		1/8/25
4	14/8/25	Build a simple feed forward neural network to Recognize Handwritten Characters		22/8/25
5	22/8/25	Study of Activation functions & Their role.		9/9/25
6	9/9/25	Implement Gradient Descent & Backpropagation in Deep neural Network.		1/10/25
7	16/9/25	Build a CNN model to classify cat & dog image.		23/9/25
8 Ex. 9 2	30/9/25	Build a RNN Network.		30/9
9 Exp	9/10/25	Experiment using LSTM		1/10/25
10	9/10/25	AutoEncoder		
11	9/10/25	VARIATIONAL AUTOENCODER		
12	17/10/25	Deep convolutional GAN		



(GAN Architecture)

17/10/25

Lab 12: Implement a Deep Convolutional GAN to generate complex color image

Aim:

To implement a deep convolutional Generative Adversarial Network to generate complex color images from random noise.

Objectives:

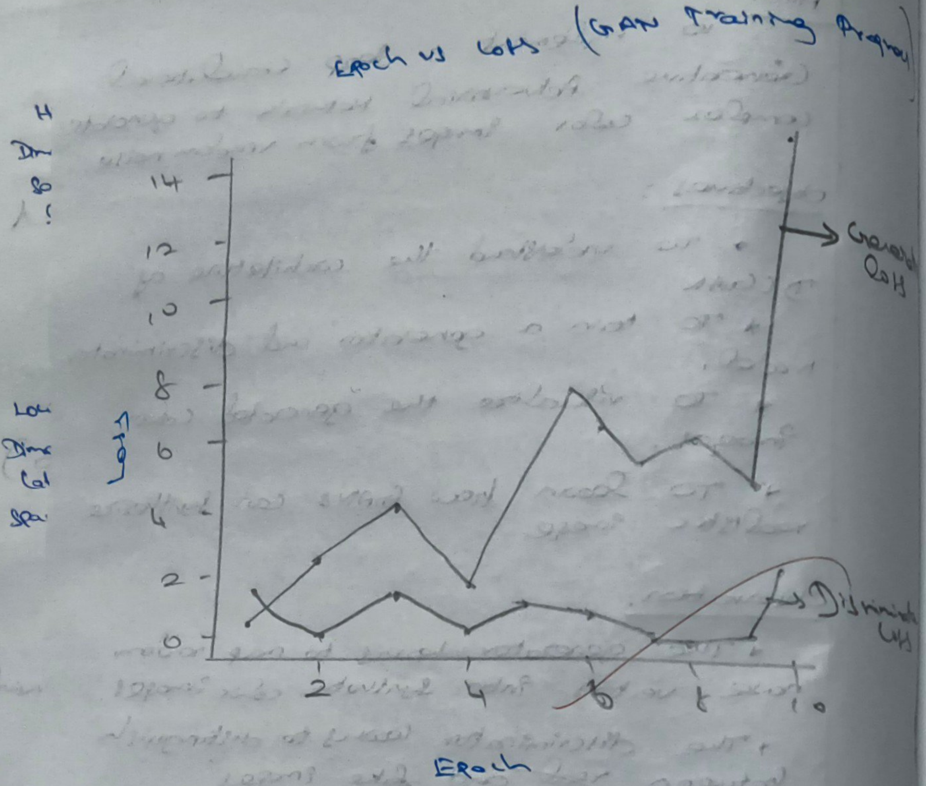
* To understand the architecture of DCGAN.
* To train a generator and discriminator model.
* To visualize the generated color images.
* To learn how GANs can synthesize realistic image.

Observation:

* The generator learns to map random noise vectors into synthetic color images.
* The discriminator learns to distinguish between real and fake images.
* During training, the generator gradually improves its output quality while the discriminator's accuracy decreases, indicating effective learning.

Pseudocode:

1) Import necessary libraries
2) Load dataset
3) Normalize images to [-1, 1]
4) Define the Generator network:
   - Transposed Conv layers + Batch Norm + ReLU



5) Define the Discriminator network:
   - Conv layers + Leaky ReLU + Dropout
6) Define loss function (Binary Cross Entropy)
7) Train the model:
   for each epoch:
   a. Train Discriminator with real and fake images.
   b. Train Generator to fool discriminator
   c. Display generated images every few epochs.
8) Save the final generated color images

Output:

Epoch 1 D Loss: 0.7148, G Loss: 2.9591
Epoch 2 D Loss: 0.5373, G Loss: 3.1537
Epoch 3 D Loss: 0.6086, G Loss: 1.5696
Epoch 4 D Loss: 0.6072, G Loss: 2.1970
Epoch 5 D Loss: 0.7659, G Loss: 1.9037
Epoch 6 D Loss: 0.3328, G Loss: 7.1927
Epoch 7 D Loss: 0.1849, G Loss: 5.0234
Epoch 8 D Loss: 0.5666, G Loss: 4.7413
Epoch 9 D Loss: 0.0875, G Loss: 6.0503
Epoch 10 D Loss: 0.0141, G Loss: 5.8257.

Result:

Successfully implemented the GAN to generated the color image.

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Connect

```
!pip install torch torchvision matplotlib
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: torchvision in /usr/local/lib/python3.12/dist-packages (0.23.0+cu126)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch) (3.20.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in /usr/local/lib/python3.12/dist-packages (from torch) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-packages (from torch) (3.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from torchvision) (2.0.2)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.12/dist-packages (from torchvision) (11.3.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.3)
```

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import torchvision.utils as vutils
import matplotlib.pyplot as plt
import numpy as np

# 1. Hyperparameters
batch_size = 128
latent_dim = 100
epochs = 10
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```python
# 1. Hyperparameters
batch_size = 128
latent_dim = 100
epochs = 10
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# 2. Data loading (CIFAR-10 color images)
transform = transforms.Compose([
    transforms.Resize(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
dataset = datasets.CIFAR10(root='./data', download=True, transform=transform)
dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=True)

# 3. Define Generator
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.model = nn.Sequential(
            nn.ConvTranspose2d(100, 512, 4, 1, 0, bias=False),
            nn.BatchNorm2d(512), nn.ReLU(True),
            nn.ConvTranspose2d(512, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256), nn.ReLU(True),
            nn.ConvTranspose2d(256, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128), nn.ReLU(True),
            nn.ConvTranspose2d(128, 64, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64), nn.ReLU(True),
            nn.ConvTranspose2d(64, 3, 4, 2, 1, bias=False),
            nn.Tanh()
        )
    def forward(self, x):
        return self.model(x)

# 4. Define Discriminator
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3, 64, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(64, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128), nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(128, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256), nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(256, 512, 4, 2, 1, bias=False),
            nn.BatchNorm2d(512), nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(512, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )
    def forward(self, x):
        return self.model(x).view(-1, 1)
```

dltlab12.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Connect

```python
# 5. Initialize models
netG, netD = Generator().to(device), Discriminator().to(device)
criterion = nn.BCELoss()
optimizerG = optim.Adam(netG.parameters(), lr=0.0002, betas=(0.5, 0.999))
optimizerD = optim.Adam(netD.parameters(), lr=0.0002, betas=(0.5, 0.999))

# Lists to store losses
d_losses = []
g_losses = []

# 6. Training loop
for epoch in range(epochs):
    for i, (real_imgs, _) in enumerate(dataloader):
        real_imgs = real_imgs.to(device)
        batch_size = real_imgs.size(0)

        # Real labels = 1, Fake labels = 0
        real_labels = torch.ones(batch_size, 1, device=device)
        fake_labels = torch.zeros(batch_size, 1, device=device)

        # Train Discriminator
        z = torch.randn(batch_size, latent_dim, 1, 1, device=device)
        fake_imgs = netG(z)
        real_loss = criterion(netD(real_imgs), real_labels)
        fake_loss = criterion(netD(fake_imgs.detach()), fake_labels)
        d_loss = real_loss + fake_loss

        optimizerD.zero_grad()
        d_loss.backward()
        optimizerD.step()

        # Train Generator
        g_loss = criterion(netD(fake_imgs), real_labels)
        optimizerG.zero_grad()
        g_loss.backward()
        optimizerG.step()

    # Store losses
    d_losses.append(d_loss.item())
    g_losses.append(g_loss.item())

    print(f"Epoch [{epoch+1}/{epochs}] D Loss: {d_loss:.4f}, G Loss: {g_loss:.4f}")

# 7. Plot Epoch Graph (Loss vs Epoch)
plt.figure(figsize=(8,5))
plt.plot(range(1, epochs+1), d_losses, label='Discriminator Loss', marker='o')
plt.plot(range(1, epochs+1), g_losses, label='Generator Loss', marker='o')
plt.title("Epoch vs Loss (GAN Training Progress)")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()
```

dltlab12.ipynb

File Edit View Insert Runtime Tools Help

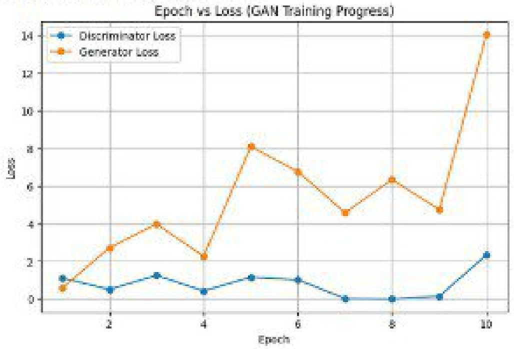
Commands + Code + Text Run all

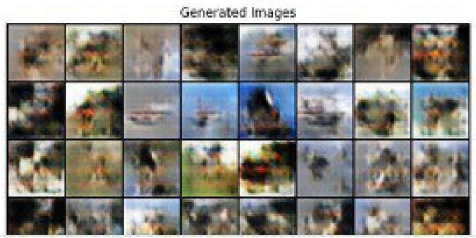
```python
        print(f"Epoch [{epoch+1}/{epochs}] D Loss: {d_loss:.4f}, G Loss: {g_loss:.4f}")

    # 7. Plot Epoch Graph (Loss vs Epoch)
    plt.figure(figsize=(8,5))
    plt.plot(range(1, epochs+1), d_losses, label='Discriminator Loss', marker='o')
    plt.plot(range(1, epochs+1), g_losses, label='Generator Loss', marker='o')
    plt.title("Epoch vs Loss (GAN Training Progress)")
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.legend()
    plt.grid(True)
    plt.show()

    # 8. Generate Samples
    z = torch.randn(64, latent_dim, 1, 1, device=device)
    fake_imgs = netG(z)
    vutils.save_image(fake_imgs, 'dcgan_generated.png', normalize=True)
    plt.figure(figsize=(8,8))
    plt.axis("off")
    plt.title("Generated Images")
    plt.imshow(np.transpose(vutils.make_grid(fake_imgs[:64], padding=2, normalize=True).cpu(), (1,2,0)))
    plt.show()
```

```
100%|██████████| 170M/170M [00:14<00:00, 11.5MB/s]
Epoch [1/10] D Loss: 1.0932, G Loss: 0.5678
Epoch [2/10] D Loss: 0.5060, G Loss: 2.7113
Epoch [3/10] D Loss: 1.2431, G Loss: 3.9827
Epoch [4/10] D Loss: 0.4229, G Loss: 2.2684
Epoch [5/10] D Loss: 1.1486, G Loss: 8.1007
Epoch [6/10] D Loss: 1.0081, G Loss: 6.7625
Epoch [7/10] D Loss: 0.0252, G Loss: 4.5718
Epoch [8/10] D Loss: 0.0043, G Loss: 6.3371
Epoch [9/10] D Loss: 0.1300, G Loss: 4.7357
Epoch [10/10] D Loss: 2.3294, G Loss: 14.0363
```





Variables Terminal