EX·NO:12
27·10·25

IMPLEMENT A DEEP CONVOLUTIONAL GAN TO GENERATE COMPLEX COLOR IMAGES

AIM

To implement a Deep convolutional gan to generate complex color images.

Objective

To understand GAN architecture
To train model on color image dataset
To observe training dynamics and quality of generating images.

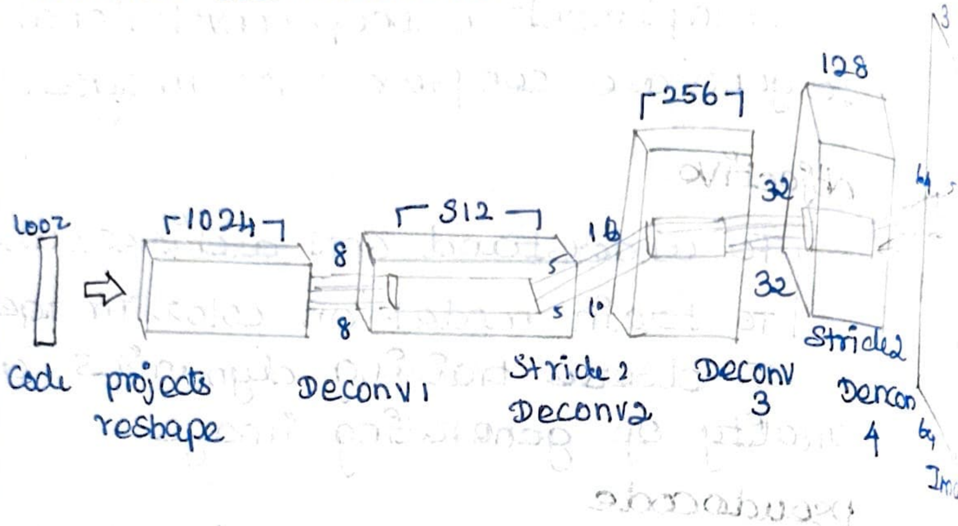
Pseudocode

import libraries and set device
Load CIFAR-10 dataset, normalize to [-1,1], create dataloader
Define DCGAN generator: series of convtranspose 2D, BatchNorm 2d, Relu
Initialize weights (Normal with mean = 0, std = 0.02)
Define Loss & optimizers
for each epochs
a) Train Discriminator real images
b) Train generator to fool Discriminator
c) Save generator batch of generator image for visualization.
Monitor losses

Observation

The discriminator and generator losses usally fluctuate this is expected in GAN training as one improves, other responds.

DCGAN architecture



Output

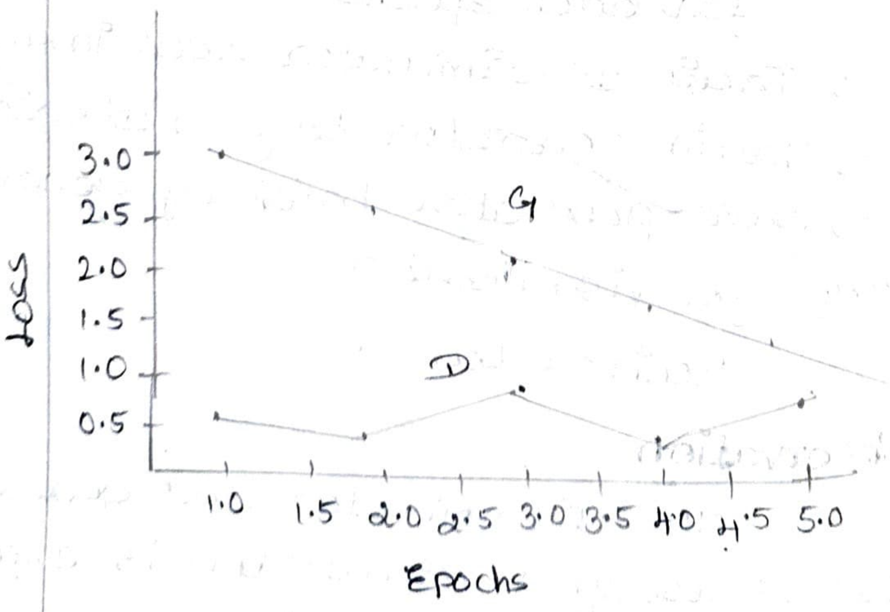
Epoch [1/5]   LOSS D : 0.4934   LOSS G : 2.9012

Epoch [2/5]   LOSS D : 0.5894   LOSS G : 266182

Epoch [3/5]   LOSS D : 0.7406   LOSS G : 2.3580

Epoch [4/5]   LOSS D : 0.5590   LOSS G : 2.0222

Epoch (5/5)   LOSS D : 1.3862   LOSS G : 1.5855



initially generator images are noisy random patterns

over epochs starts showing structure (colors, shape)

saving images each epoch helps visualize progression from noise → structure → clearer images.

Result

successfully implemented DCGAN.

EX·NO:13

27·10·25

UNDERSTANDING THE ARCHITECTURE OF PRE-TRAINED MODEL

Aim

TO understand the architecture of Pre-trained model.

objective

TO learn the concept of transfer learning and how it reduces training time.

TO explore the layer wise architecture of pretrained models such as VGG16, Alexnet

TO identify the role

TO visualize how these models extract features.

pseudocode

import libraries

Load a pre-trained model

~~Model~~ Display it

observe : input and output shapes layers names, types and no.of parameter

visualize the architecture using

Load a sample input image and preprocess it for model

pass image through the model and get prediction

optionally visualize activation.

output

Linear (in-features = 2048, out-features = 10,
bias = True)

VGG (
(features): sequential (
(0): Conv 2d (3, 64, kernel_size = (3,3), stride = (1,1)
padding = (1,1))
(1) : ReLU (inplace = True)
(2) : conv 2d (64, 64, kernel size = (3,3),
stride = (1,1)
padding (1,1)
(3) : ReLU (inplace = True)
(4) maxpool 2d (kernel_size = 2,
stride = 2 padding = 0,
dilation = 1 ceil = mode = false)

⋮

(30) : maxpool2d (kernel_size = 2,
stride = 2,
padding = 0, dilation = 1, ceil = mode = false)
(avg pool) = Adaptive Avg pool (output_size
(7,7))
(0) : linear
(1) : ReLU (inplace = True)
(2) : Dropout (p = 0.5, inplace = false)

Total Trainable parameters : 138357544

Observation

The pre-trained model contains millions of parameter spread across convolutional of dense layers.

Early layer extract edges and texture while deeper layers capture high level feature such as object parts.

The use of pre-trained weights drastically reduces training time and improves model accuracy for new tasks.

Result

Successfully Implemented pre-trained model.