Jordyn Brooks
CPSC 8430 - Deep Learning
Homework 4
Github link: https://github.com/brooksjk/Deep_Learning - under folder HW4

Generative Adversarial Networks (GANs) have transformed generative modeling by learning to generate realistic images using adversarial training. In this homework, we implemented and trained three types of GANs using the CIFAR-10 dataset: DCGAN, WGAN, and ACGAN (bonus). The goal was to assess the performance of these models in terms of image quality and training stability, as well as to highlight the top ten photographs generated by each model.

CIFAR-10

The CIFAR-10 dataset is a widely used benchmark for image classification and generative tasks. It contains 60,000 32x32 color images across 10 classes, with 50,000 images for training and 10,000 for testing. Images were normalized to the range [-1, 1] to facilitate GAN training. No data augmentation techniques were applied.

Model Architectures

DCGAN

The Deep Convolutional GAN (DCGAN) consists of two main components:

- Generator:
    - Takes a random noise vector z sampled from a uniform distribution as input.
    - Uses a series of transposed convolutional layers to upsample the noise into a realistic image. Each layer is followed by BatchNorm and ReLU activation (except for the output layer, which uses Tanh to scale pixel values to [-1, 1]).
    - Architecture:
        - Input: z (100-dimensional vector)
        - Fully connected layer (reshaped to 4×4×512)
        - 4 transposed convolutional layers:
        - 512→256→128→64→output channels (e.g., 3 for RGB)
    - Output: 64×64 RGB image.
- Discriminator:
    - A convolutional network that downsamples an input image through strided convolutions to classify it as real or fake. LeakyReLU is used for activations, with no BatchNorm in the final layer.
    - Architecture:
        - Input: 64×64 RGB image

        - 4 convolutional layers with strides
        - 2×2, followed by a fully connected layer.
- Output: Probability that the image is real.

WGAN

The Wasserstein GAN modifies the traditional GAN to improve stability and address mode collapse:

- Generator:
    - Similar to DCGAN, with a noise vector z as input and a series of transposed convolutional layers.
    - Key difference: Uses the Wasserstein loss instead of the cross-entropy loss.
- Discriminator (Critic):
    - Similar to DCGAN but without BatchNorm.
    - Outputs a scalar score instead of a probability.
    - Uses weight clipping (e.g., $[-0.01, 0.01]$) to enforce the Lipschitz constraint.

ACGAN

The Auxiliary Classifier GAN introduces class conditioning:

- Generator:
    - Takes both noise z and a class label y as inputs.
    - Class labels are embedded into a dense vector and concatenated with z.
    - The combined input is processed through a similar architecture as DCGAN.
- Discriminator:
    - Outputs two values for each input: Real/Fake score.
    - Predicted class label (via a softmax layer).
    - Loss: Combines adversarial loss and classification loss.

Implementation Details

- Preprocessing:
    - Resized all images to 64×64 pixels.

    - Normalized pixel values to the range $[-1,1]$ to match the Tanh activation of the generator output.

- Hyperparameters:
    - Learning rate:
        - 2e-4 (DCGAN, ACGAN),
        - 5e-5 (WGAN).
    - Optimizer: Adam (DCGAN, ACGAN), RMSprop (WGAN).
    - Batch size: 8
    - Noise dimension z: 100
    - Number of Epochs: 500
- Loss Functions:
    - DCGAN: Binary Cross-Entropy Loss.
    - WGAN: Wasserstein Loss with weight clipping.
    - ACGAN: Combined Binary Cross-Entropy (for real/fake) and Cross-Entropy (for class labels).

Training Workflow

- Initialize the generator and discriminator for each model.
- For each batch:
    - Sample z (and class labels for ACGAN) and generate fake images.
    - Train the discriminator/critic with both real and fake images.
    - Update the generator based on the discriminator's feedback.
    - Log loss values for generator and discriminator after each epoch.

Results

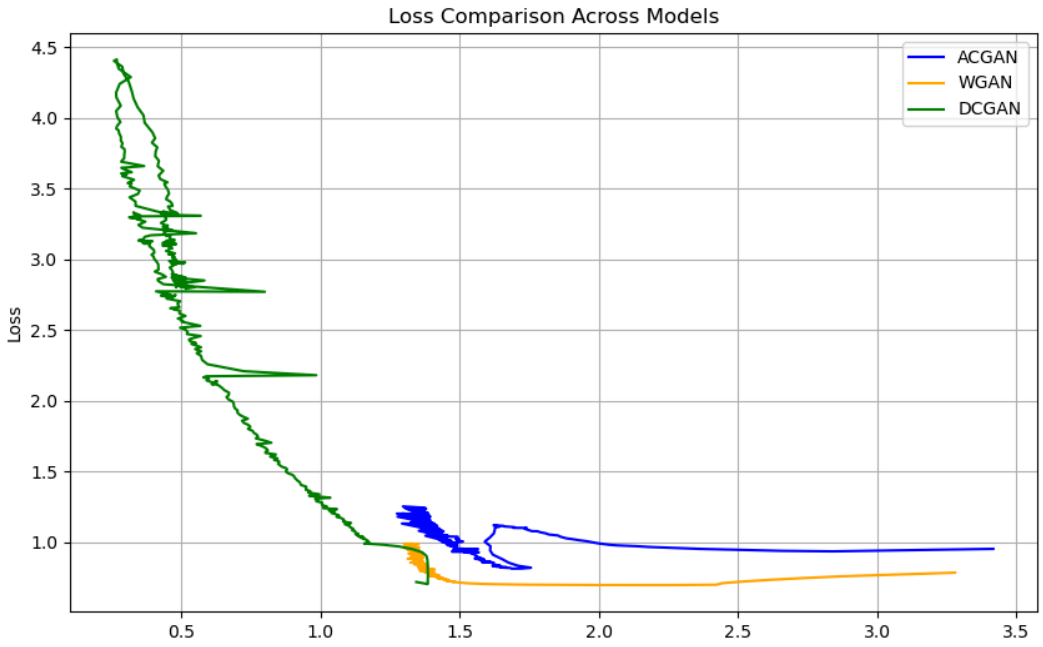
Generated Images

DCGAN

WGAN

ACGAN

Loss Comparison



The graph compares the loss trends of three models (ACGAN, WGAN, and DCGAN) over the training process. WGAN demonstrates the most stable and efficient performance, maintaining a low loss throughout and stabilizing quickly. This suggests that WGAN handles loss minimization effectively, likely due to its use of the Wasserstein loss and gradient penalty, which contribute to improved stability in adversarial training. ACGAN also performs well, showing a steady decrease in loss and eventual stabilization, though it takes slightly longer to reach the same level of stability as WGAN. In contrast, DCGAN exhibits the highest initial loss and significant fluctuations throughout training, indicating challenges with convergence and stability. These fluctuations may point to issues such as vanishing or exploding gradients, requiring additional regularization or hyperparameter tuning to improve performance. Overall, WGAN appears to be the most robust model, with ACGAN as a competitive alternative, while DCGAN may require further adjustments to achieve comparable results.