GAN IMPLEMENTATION

Deep Convolutional GAN

Wasserstein GAN-GP

DATASET:

CalebA for Wasserstein GAN-GP.

CelebA Dataset

- **Description**: A large-scale dataset for face attributes and facial recognition.
- **Images**: 178x218 pixel color images (Resized to 64x64).
- **Number of Classes**: 40 attribute labels, plus 10,177 unique identities.
- **Total Images**: 202,599 images.(Subset of 5000 used)
- **Source**: The CelebA dataset was collected from celebrity images on the web.
- **Usage**: Used for training models in facial recognition, attribute prediction, and face generation tasks.
- **Format**: Images in .jpg format, annotations in text files with attributes and bounding boxes.

Model Architecture (Wasserstein GAN-GP)

Discriminator:

- **Layers:**
    - Conv2d(channels_img, features_d, kernel_size=4, stride=2, padding=1)
    - LeakyReLU(0.2)
    - _block(features_d, features_d * 2, 4, 2, 1)
    - _block(features_d * 2, features_d * 4, 4, 2, 1)
    - _block(features_d * 4, features_d * 8, 4, 2, 1)
    - Conv2d(features_d * 8, 1, kernel_size=4, stride=2, padding=0)
- **Purpose:** Discriminates real vs. generated images.

Generator:

- **Layers:**

    - _block(channels_noise, features_g * 16, 4, 1, 0)
    - _block(features_g * 16, features_g * 8, 4, 2, 1)
    - _block(features_g * 8, features_g * 4, 4, 2, 1)
    - _block(features_g * 4, features_g * 2, 4, 2, 1)
    - ConvTranspose2d(features_g * 2, channels_img, kernel_size=4, stride=2, padding=1)
    - Tanh()
- **Purpose:** Generates images from noise.

Optimizers:

- **Adam:** lr=1e-4, betas=(0.0, 0.9) for both Generator and Discriminator.

Loss Function:

- **Gradient Penalty:** Regularizes the discriminator to satisfy the Lipschitz constraint.

Data Pipeline:

- **Dataset:** CelebA
- **Transforms:**
    - Resize((64, 64))
    - ToTensor()
    - Normalize((0.5), (0.5))
- **DataLoader:** Batch size 64, shuffle=True.

Justification:

- **LeakyReLU:** Avoids dead neurons.
- **Adam Optimizer:** Efficient and adaptive.
- **Tanh Activation:** Normalizes image output to [-1, 1].
- **Gradient Penalty:** Enforces Lipschitz constraint for stable training.

EVALUATION

Evaluating Generative Adversarial Networks (GANs) presents significant challenges due to their inherently subjective nature. Unlike traditional machine learning models, GANs lack a straightforward evaluation metric. As a result, assessing their performance often involves indirect methods such as using pretrained models to compute metrics like Inception Score (IS) or Fréchet Inception Distance (FID).

In my project, I have opted for a more direct approach by utilizing TensorBoard logs to monitor the quality of the generated images. This method involves:

- TensorBoard Logs: Each epoch's generated images are logged and visualized using TensorBoard. This provides a clear view of how the GAN's outputs evolve over time and allows for a more intuitive assessment of image quality.
- Visual Inspection: By examining the images produced at different training stages, I can gauge the progress and improvements in the generated results.

This approach enables a detailed and visual evaluation of the GAN's performance, offering insights into its effectiveness through direct observation of the generated samples.

Conclusion

The project has been a rewarding exploration into the realm of Generative Adversarial Networks (GANs), providing valuable insights and hands-on experience with advanced generative models. Here's a summary of the key aspects of the project:

Key Learnings:

- **Architectural Insights**: Implementing and fine-tuning GAN architectures has deepened my understanding of how generator and discriminator networks interact and the importance of architectural choices such as layer types and activation functions.
- **Training Dynamics**: I have gained practical experience in managing the training complexities of GANs, including balancing the generator and discriminator, and addressing common issues like mode collapse and non-convergence.

Challenges Overcome:

- **Stabilizing Training**: Navigating the training instability of GANs was a significant challenge. Implementing techniques like gradient penalty and using TensorBoard for visualization helped in managing and stabilizing the training process.
- **Evaluating Outputs**: Given the subjective nature of GAN evaluation, I tackled the challenge of performance assessment by leveraging TensorBoard logs. This method provided a direct and intuitive way to track the progress and quality of generated images.

Future Improvements and Extensions:

- **Enhanced Evaluation Metrics**: Future work could involve incorporating more sophisticated evaluation metrics such as Inception Score (IS) or Fréchet Inception Distance (FID) to provide a more quantitative assessment of the generated images.
- **Model Refinements**: Exploring variations in GAN architectures and hyperparameters, such as different types of normalization or alternative loss functions, could lead to further improvements in output quality.
- **Broader Applications**: Extending the project to different datasets or applications could provide additional insights into the versatility and robustness of the GAN model.

Overall, this project has been an excellent learning experience, highlighting both the potential and challenges of working with GANs. The insights gained and the obstacles overcome will serve as a solid foundation for future research and experimentation in the field of generative modeling.