Style Transfer GAN with Diffusion

**Overview**

This project implements a Style Transfer GAN (Generative Adversarial Network) with diffusion-based image generation. It combines modern deep learning techniques to transfer artistic styles from reference images to content images while maintaining high quality and detail.

**Features**

- Diffusion-based image generation

- Style encoding and transfer

- Batch processing support

- Real-time training visualization

- Checkpoint management

- Command-line interface for inference

- Comprehensive model evaluation tools

**Requirements**

python >= 3.8

torch >= 1.9.0

torchvision

pillow

tqdm

matplotlib

numpy

**Project Structure**

project/

│

├── data/

│ ├── content\_images/ # Content images (CelebA dataset)

│ └── style\_images/ # Style reference images

│

├── checkpoints/ # Model checkpoints

├── samples/ # Generated samples during training

└── evaluation\_results/ # Model evaluation metrics

```

**Configuration**

Key parameters can be modified in the Config class:

class Config:

IMAGE\_SIZE = 128

BATCH\_SIZE = 32

NUM\_EPOCHS = 50

G\_LEARNING\_RATE = 2e-4

D\_LEARNING\_RATE = 1e-4

LATENT\_DIM = 100

STYLE\_DIM = 512

NUM\_TIMESTEPS = 1000

Installation

1. Install dependencies:

```bash

pip install -r requirements.txt

```

3. Set up your data directories as specified in the Config class

**Usage**

**Training**

# Start new training

trainer = Trainer()

trainer.train()

# Resume training from checkpoint

trainer = Trainer()

trainer.load\_checkpoint('checkpoints/checkpoint\_best.pt')

trainer.train()

**Inference**

# Single image processing

inference = StyleGANInference("checkpoints/checkpoint\_best.pt")

result = inference.generate\_from\_style(

"content\_images/image.jpg",

"style\_images/style.jpg"

)

# Using the Command Line Interface

python main.py

**Model Evaluation**

evaluator = ModelEvaluator("checkpoints/checkpoint\_best.pt", device)

evaluator.run\_full\_evaluation()

**Model Architecture**

**Generator**

- Style encoder network

- Time embedding module

- Progressive upsampling blocks

- Final image generation layer

**Discriminator**

- Convolutional architecture

- Multiple discrimination scales

- Label smoothing support

**Diffusion Scheduler**

- Progressive noise addition

- Controlled image generation process

- Timestep embedding

**Training Process**

1. Content and style image batching

2. Progressive diffusion

3. Style encoding

4. Adversarial training

5. Periodic evaluation and checkpointing

**Evaluation Metrics**

- PSNR (Peak Signal-to-Noise Ratio)

- SSIM (Structural Similarity Index)

- FID (Fréchet Inception Distance)

- Inference speed measurements

- Model parameter statistics

**Command Line Interface**

The project includes a user-friendly CLI with the following options:

1. Single Image Style Transfer

2. Batch Processing

3. View Model Stats

4. Exit

**Performance Metrics**

- Average inference time: ~0.0023 seconds per image

- Images per second: ~436

- Total parameters: ~45.5M

- Generator: 34.3M parameters

- Discriminator: 11.2M parameters

**Memory Requirements**

- Minimum GPU RAM: 6GB

- Recommended GPU RAM: 8GB+

- Training batch size can be adjusted based on available memory

Known Limitations

- Image size fixed at 128x128 pixels

- Style transfer quality depends on style image similarity

- GPU memory intensive during training

- Requires consistent image formats and sizes

Troubleshooting

1. Out of Memory Errors:

- Reduce batch size

- Decrease image dimensions

- Enable gradient checkpointing

2. Training Instability:

- Adjust learning rates

- Modify discriminator update frequency

- Check input data normalization

3. Poor Style Transfer:

- Ensure style images are representative

- Adjust style dimension parameters

- Increase training duration

**Acknowledgments**

- CelebA dataset for content images

- PyTorch team for the deep learning framework

- Diffusion models research community