***DataPreprocessingForStyleGAN3FineTuneAndFeatureEnhancement.py***

**Overview**

This script preprocesses facial expression images for fine-tuning StyleGAN3 models, specifically focused on disgust/anger expressions. It performs image resizing, color enhancement, and contrast adjustments to create a standardized dataset.

**Features**

- Resizes images to 256x256 pixels using Lanczos resampling

- Applies CLAHE (Contrast Limited Adaptive Histogram Equalization) for improved lighting

- Enhances image contrast using a configurable factor

- Batch processes all images in a source directory

**Usage**

1. Set the `SOURCE\_PATH` to your input images directory

2. Set the `TARGET\_PATH` to your desired output directory

3. Adjust the `CONTRAST\_FACTOR` if needed (default: 1.5)

## Output

Processed images are saved as PNG files with sequential naming (angry\_00000.png, angry\_00001.png, etc.)

**Notes**

- This preprocessing pipeline is optimized for facial expression datasets

- The script maintains image quality with a quality setting of 100

- Progress updates are printed every 100 processed images

***StyleGAN3FullTrainingG&D.py***

**Overview**

This script implements a comprehensive training pipeline for fine-tuning StyleGAN3 models on custom datasets, supporting both full GAN training (Generator + Discriminator) and generator-only training modes.

**Features**

- Fine-tunes pre-trained StyleGAN3 models on custom datasets

- Supports both full GAN training and generator-only training

- Selective layer freezing/unfreezing for precise control over training

- Adaptive batch size to prevent memory issues

- Automatic progress tracking with image generation

- Style mixing regularization

- R1 gradient penalty for discriminator

- Dataset weight adjustment for controlled style transfer

**Requirements**

- Python 3.8+

- PyTorch 1.9+

- CUDA 11.0+

- StyleGAN3 repository

- Pillow (PIL)

- NumPy

- OpenCV (cv2)

**Key Arguments**

- `--pretrained\_model`: Path to the pretrained model pickle file

- `--output\_dir`: Output directory for training results

- `--dataset\_path`: Path to custom dataset directory

- `--batch\_size`: Batch size for training

- `--train\_mode`: Training mode (`GAN` or `generator\_only`)

- `--g\_unfreeze\_blocks`: Comma-separated list of generator blocks to unfreeze

- `--d\_unfreeze\_blocks`: Comma-separated list of discriminator blocks to unfreeze

- `--total\_kimg`: Total training length in thousands of images

**Output**

- Model snapshots saved periodically

- Progress images generated throughout training

- Training metrics and logs

- Final model checkpoint

**Notes**

- The script automatically adapts to available GPU memory

- Progress images are generated every 5 minutes to visualize training

- Detailed logs are saved in the output directory

***StyleGAN3SynImgGen.py***

**Overview**

This script generates synthetic images using a pre-trained or fine-tuned StyleGAN3 model. It allows for batch generation of images across a range of random seeds, with automatic saving of results.

**Features**

- Generates high-quality synthetic images from a StyleGAN3 model

- Supports batch generation across a specified seed range

- Handles both conditional and unconditional StyleGAN3 models

- Includes memory management to prevent CUDA out-of-memory errors

- Provides error handling to continue generation despite occasional failures

- Supports truncation parameter for controlling variation/quality tradeoff

**Requirements**

- Python 3.8+

- PyTorch 1.9+

- CUDA-compatible GPU

- StyleGAN3 repository

- Pillow (PIL)

- NumPy

- Matplotlib

- IPython (for optional display functionality)

**Usage**

1. Set the path to your StyleGAN3 model in the script:

```python

with open('PATH\_TO\_YOUR\_MODEL.pkl', 'rb') as f:

2. Configure the seed range and output directory:

SEED\_FROM = 37000

SEED\_TO = 38000

SAVE\_DIRECTORY = "PATH\_TO\_OUTPUT\_DIRECTORY/"

3. Customization

Adjust truncation\_psi (default 1.0) to control variation (lower values = less variation, higher quality)

Set noise\_mode to 'const', 'random', or 'none' to control stochastic noise

For conditional models, specify class\_idx to generate images from a particular class

**Output**

- Generated images are saved as PNG files in the specified output directory with filenames based on their seed values.

**Notes**

- The script automatically clears CUDA cache every 10 images to prevent memory issues

- For conditional models, label handling is included but requires setting the appropriate class index

- Failed generations are skipped rather than stopping the entire batch