Generative Adversarial Networks (GANs) in Image Generation

Jakub Oleszczuk 260410
Paweł Wieszczeczyński 260414
Jan Ruciński 260291

1. Justification

Innovative Potential:
GANs offer a cost-effective alternative to networks like DALL-E, making them appealing to industries seeking efficient AI-driven content creation.
Market Demand:
Industries like gaming, film, and marketing increasingly rely on AI-generated content to save time and reduce costs.
Applications:

- **Art:** Creation of unique digital artwork.
- **Content Creation:** Quick generation of gaming and media assets.
- **Data Augmentation:** Enhancing datasets for training other AI models.

Despite GANs' competitive limitations compared to modern genAI solutions, this project aims to explore and optimize their performance using current techniques and resources.

2. Project Scope

Primary Focus:
Build a GAN-based system for generating high-quality images, exploring architecture optimizations to improve performance.
Secondary Focus:

- Investigate modern alternatives like TransGAN.
- Evaluate results using limited computational resources.
  Research Objectives:
  Study the impact of different GAN architectures on image quality and diversity.
  Practical Applications:
- AI-driven art creation.
- Synthetic data generation for training other AI models.

3. Goal / Aims

Main Objective:
Develop a high-performance GAN capable of generating photorealistic images.
Sub-Goals:

- Enhance training stability and minimize artifacts in generated images.

- Compare the performance of various GAN architectures.
- Implement a user interface to generate images.

4. Features to be Implemented

Primary Features (Must-Have):

- **GAN Architecture Implementation:** Use standard or DCGAN architectures.
- **Dataset Preparation:** Preprocess and train on a large dataset.
- **Training and Tuning:** Set up training pipelines with hyperparameter tuning.
- **Evaluation Metrics:** Measure image quality using Inception Score and FID.

Secondary Features (Nice-to-Have):

- Explore **TransGAN** architecture.
- Develop an easy-to-use UI for generating images.

5. Techniques & Technologies

- **Core Technology:** GAN, WGAN, TransGAN, each network trained for roughly 12 hours
- **Programming Language:** Python.
- **Frameworks:** PyTorch
- **Infrastructure:** Google Collab, and Local Learning
- **Datasets:** 39,000 source images depicting cats

6. Short State of the Art

Competitive Solutions:

- **DCGANs:** Higher-quality results using convolutional layers.
- **Conditional GANs (cGANs):** Enables generation based on labels or conditions (e.g., specific objects).
- **StyleGAN:** Advanced model capable of detailed photorealistic generation.

Emerging Techniques:

- **Visual Transformers:** Adapted for 2D images, transforming AI generative tasks.
- **Diffusion Networks:** Primarily generative, contrasting traditional transformers.

Challenges:

- Mode collapse, training instability, dataset size, and high computational costs

7. Implementation Description

Development Steps:

1. Conducted a literature review on GANs and their existing implementations.
2. Prepared the dataset through preprocessing.
3. Implemented and trained the models using PyTorch.

4. Fine-tuned hyperparameters and evaluated model performance.
5. Prepared a UI for image generation.

Deployment Requirements:

- **Hardware:** High-performance GPU (e.g., NVIDIA).
- **Software:** PyTorch, Python, cloud computing services .
- **Installation Steps:**
  - Set up the development environment with required libraries.
  - Download code from repository (https://github.com/JanRucinski/MMCV_MON0730_GAN_IMAGE_GEN)
  - Train the model on the dataset.
  - Run the UI (UI.py)

8. Conclusions

- GAN training is heavily reliant on resources.
- Different architectures present various problems.