InfoGAN: Implementation and Results

Interpretable Representation Learning by Information Maximizing GANs

Nasykhova Anastasia
Slovyagina Anna
Tarasova Sofia

December 15, 2025

Project Motivation

Why InfoGAN?

- Unsupervised disentanglement of data factors
- Interpretable latent representations
- Control over generation process
- Information-theoretic approach

Key Innovation

Maximize mutual information between latent codes c and generated images $G(z, c)$ without supervision

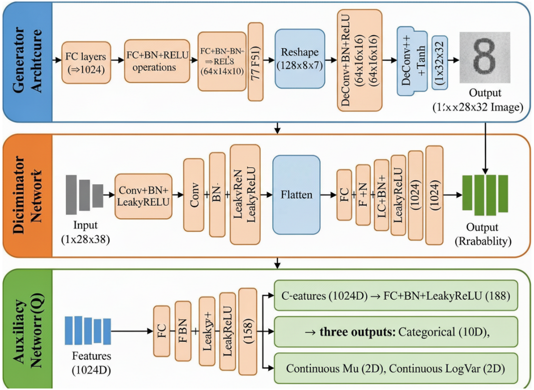
$$I(c; G(z, c)) \to \max$$

Our Goals:

- Reproduce paper results on MNIST
- Extend to new datasets (chairs, faces)
- Analyze learned representations
- Understand limitations

Expected Benefits

- Data augmentation
- Feature manipulation
- Better understanding of data structure



Design Decisions:

- DCGAN-based architecture for stability
- Shared layers between D and Q (minimal overhead)
- Batch normalization in generator
- LeakyReLU activations in discriminator

Theoretical Foundation

Standard GAN Objective

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

InfoGAN Extension

$$\min_{G,Q} \max_D V_{InfoGAN}(D, G, Q) = V(D, G) - \lambda I(c; G(z, c))$$

Variational Lower Bound:

$$\begin{aligned} I(c; G(z, c)) &\geq L_I(G, Q) \\ L_I(G, Q) &= \mathbb{E}_{c \sim P(c), x \sim G(z,c)}[\log Q(c|x)] + H(c) \end{aligned}$$

Where $Q(c|x)$ is an auxiliary network approximating the posterior distribution

Implementation and Training Details

Implementation Highlights

- Implemented in **Python using PyTorch**
- Full **GPU (CUDA) support**
  - Models and tensors transferred to GPU
  - Significant speedup compared to CPU training
- Trained on all **five datasets from the original InfoGAN paper**:
  - MNIST, SVHN, CelebA
  - Faces, Chairs
- Additional dataset:
  - **Fashion-MNIST**
  - Same architecture as MNIST
  - Stable training and interpretable factors

Training Configuration

- **Optimizer:** Adam
  - $\beta_1 = 0.5, \ \beta_2 = 0.999$
- **Learning Rates**
  - Generator (G): 1×10^{-3}
  - Discriminator (D): 2×10^{-4}
- **Batch Size**
  - MNIST: 128
  - Chairs: 32
  - Faces: 64
- **Epochs:** 15 – 150 (dataset-dependent)
- **MI Weight** (λ): 1.0

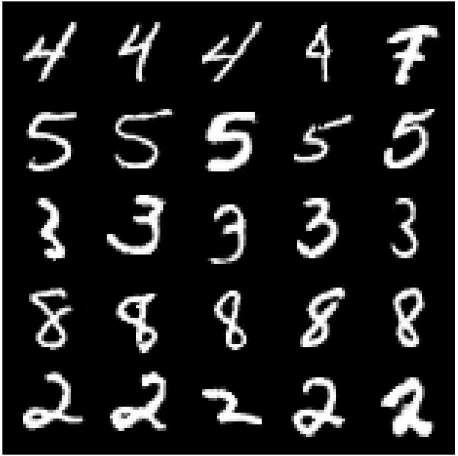
Fashion-MNIST Network Architecture

Table: Discriminator and generator for Fashion-MNIST.

Discriminator D / **Recognition** Q	**Generator** G
Input 28×28 Gray image	Input $\in \mathbb{R}^{d_z}$
4×4 conv. 64 Leaky-RELU, stride 2	FC 1024 Leaky-RELU + BatchNorm
4×4 conv. 128 Leaky-RELU, stride 2 + BatchNorm	FC $7 \times 7 \times 128$ Leaky-RELU + BatchNorm
FC 1024 lRELU + BatchNorm	4×4 upconv 64 Leaky-RELU, stride 2 + BatchNorm
FC output layer for D	4×4 upconv 1 Tanh, stride 2
FC 128 + BatchNorm + Leaky-RELU + FC output for Q	

Architecture mirrors MNIST, adapted for Fashion-MNIST images.

Generated Samples: MNIST and Fashion-MNIST



MNIST

Fashion-MNIST

Generated Samples: SVHN and CelebA

SVHN

CelebA

Generated Samples: Faces and Chairs

Faces

Chairs

Convergence Patterns:

- D/G losses stabilize after epoch 10-15
- Faces require more epochs

Quantitative Evaluation

Dataset	Image Quality	Disentanglement
MNIST	Excellent	Clear
Fashion-MNIST	Excellent	Clear
SVHN	Good	Moderate
CelebA	Good	Partial
Faces	Good	Partial
Chairs	Good	Moderate

Table: Performance metrics across datasets (qualitative assessment)

Observations:

- All datasets show results comparable to the original paper.
- SVHN and Faces perform slightly worse; some unsuccessful samples exist.
- CelebA color faces are challenging; disentanglement is harder.
- Fashion-MNIST confirms method generalizes to new datasets.
- More complex objects (Faces, Chairs, CelebA) require careful parameter tuning; disentanglement is more difficult.

Computational Limitations

Hardware Constraints

- Limited to Google Colab with T4 GPU (16GB VRAM)
- Training time: 4-8 hours for complex datasets
- Batch size limited to 32-128
- Could not train on full resolution (64×64 max)

Architectural Simplifications

- Simple DCGAN architecture (not state-of-the-art)
- Limited capacity for very complex patterns
- Trade-off between model size and training time

Future Work and Extensions

1. More Complex Datasets

- Natural images (ImageNet subsets)
- 3D objects (multi-view consistency)

2. Advanced Patterns and Attributes

- Beyond rotation: lighting direction, material properties
- Hierarchical attributes (coarse to fine)
- Compositional generation (combine multiple objects)
- Style transfer applications

3. Improved Architectures

- StyleGAN-based generators (better quality)
- Progressive growing for high-resolution images
- Transformer-based architectures

Summary of Achievements

What We Accomplished

- Successfully reproduced InfoGAN on all datasets from original paper
- Extended to one new datasets (Fashion-MNIST)
- Demonstrated unsupervised disentanglement
- Analyzed limitations and challenges
- Identified future research directions

Practical Takeaways

- InfoGAN works well for simple patterns (rotation, type)
- Complex attributes require larger models and more data
- Careful hyperparameter tuning is essential
- Trade-offs between quality, speed, and interpretability

InfoGAN demonstrates that information-theoretic principles can guide unsupervised learning of interpretable representations

Core Contribution

We validated InfoGAN's approach across three datasets with different complexity levels, showing both its potential and practical limitations when using modest computational resources.

Future Impact

As generative models become more powerful and computational resources more accessible, information-maximization approaches like InfoGAN will become increasingly valuable for controllable generation and representation learning.

Questions?

Thank you for your attention!

Code and Resources:

- Implementation available on GitHub