InfoGAN Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Xi Chen†‡, Yan Duan†‡, Rein Houthooft†‡, John Schulman†‡, Ilya Sutskever‡, Pieter Abbeel†‡ († UC Berkeley, Department of Electrical Engineering and Computer Sciences‡ OpenAI)

Presented by – Sumit Dugar for the practical course – Deep learning in real world

TECHNISCHE UNIVERSITÄT MÜNCHEN

What is InfoGAN?

InfoGAN is an extension to Generative Adversarial Networks that learns disentangled (interpretable) representations of the latent variables in a completely unsupervised manner. These interpretable features can be used by many downdtream tasks such as classification, regression, visualization, and policy learning in reinforcement learning.

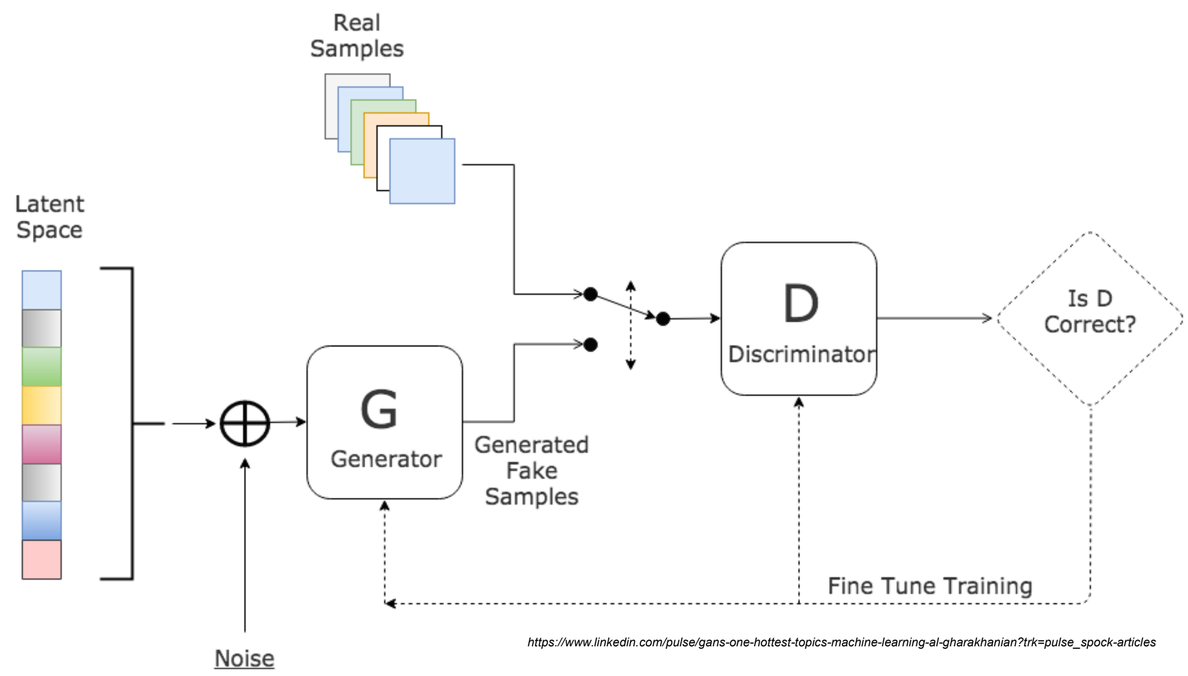
The core idea of InfoGAN

The important thing about InfoGAN is that it imposes certain restriction on the manner in which generator uses latent codes. This restriction causes the individual dimensions of latent codes to represent semantic features of the data. Instead of using a single noise vector z as in GAN, InfoGAN decomposes noise vector in the following two parts :

- z : incompressible noise as was earlier used in GAN
- c : these are latent codes that will represent interpretable information of data.

In order to discover these latent codes in an unsupervised manner InfoGAN proposes to maximise mutual information between these latent codes and the samples generated by G. This mutual information is used as a regularizer along with the original value function of GAN.

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$



https://www.linkedin.com/pulse/gans-one-hottest-topics-machine-learning-al-gharakhanian?trk=pulse_spock-articles

Mutual Information I

It is the measure of mutual dependence between two random variables X & Y.

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

The intution behind using I as a regularizer is that by maximising I we are reducing the uncertainty of X (i.e c) when Y (i,e G(z,c)) is observed.

Optimization of Mutual Information

Since it is difficult to maximize mutual information I directly, so we obtain a lower bound of I and then optimize it.

$$\begin{aligned} I(c; G(z,c)) &= H(c) - H(c|G(z,c)) \\ &= \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log P(c'|x)]] + H(c) \\ &= \mathbb{E}_{x\sim G(z,c)}[\underbrace{D_{\mathrm{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\geq 0} + \mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\geq \mathbb{E}_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \end{aligned}$$

Training of InfoGAN

There are very few differences in the training and performance of InfoGAN and GAN. A minimax game is set between two players. One of them is the generator(G). It takes in some latent codes (c,z) and tries to create fake samples that looks as if they have been sampled from the actual data distribution. The other player is discriminator(D). It takes either real or fake samples and examines if the samples are real or fake. It also outputs modified c that can be used for the reconstruction of c. Both these networks are sequentially trained and learning(gradient) from each round is used by both the players to improve their game.

Algorithm :

1) Sample (z,c) from any prior distribution that suits us - categorical, uniform, gaussian etc.
2) Generate some fake samples x_fake = G(z,c) using generator network.
3) Run Discriminator network on real samples real_d = D(x_real).
4) Run Discriminator network on fake samples fake_d = D(x_fake).
5) Calculate posterior q(c|x_fake) using encoder network.
6) Calculate discriminator loss and generator loss using 3,4,5 and minimize them sequentially.

$$D_{loss} = -\frac{1}{m}\sum_{i=1}^{m}[logD(x_{real}) + log(1 - D(x_{fake}))] - \lambda I(c, x_{fake})$$

$$G_{loss} = -\frac{1}{m}\sum_{i=1}^{m}[log(D(x_{fake}))] - \lambda I(c, x_{fake})$$

Training challenges : InfoGAN and GAN are notoriously hard to train.

Non convergence

We might not converge to the Nash equilibrium at all. Gradient descent techniques that are designed to find a low value of a cost function, rather than to find the Nash equilibrium of a game may fail to converge.
It is difficult to maintain power balance between the two networks.

Some work arounds

- Train discriminator more in comparison to generator.
- Pre-train discriminator together with real and fake data.
- Use labels to train discriminator

Mode collapse

Generator fails to output diverse samples. It collapse into very narrow distribution that only covers a single mode in data distribution. The implication of mode collapse is that generator can only generate very similar samples (e.g. a single digit in MNIST).

Some work arounds

- Minibatch discrimination
- Feature matching

Programming challenges

Debugging these networks was very hard.
Processing power was not enough for training different datasets.

Some tips

- Write test cases for checking each module don't wait for integration.
- Use tensorboard for visualizing graphs, scalars and generated images as soon in the development as possible.
- Use checkpoints to save training results and resume from previous best checkpoint in case you lack processing power.
- Convert your data into binary format and use it for training instead to directly reading images. This saves a lot on training time.

Training Environment

- Memory : 8 GB
- Processor : (Intel® Core™ i5-6200U CPU @ 2.30GHz) × 4
- Os : Ubuntu 16.04
- Framework : Tensorflow

Hyper parameters for MNIST

- D learning rate : 2e-4
- G learning rate : 1e-3
- Regularization parameter : 1
- 1 ten-dimensional categorical code (c1), 2 continuous latent codes (c2, c3) and 62 noise variables (z).
- Leaky rate : 0.1
- Batch size : 128
- Number of epochs : 50

These results were generated by using vanilla multi layer neural network for discriminator D, generator G and recognition networks Q.

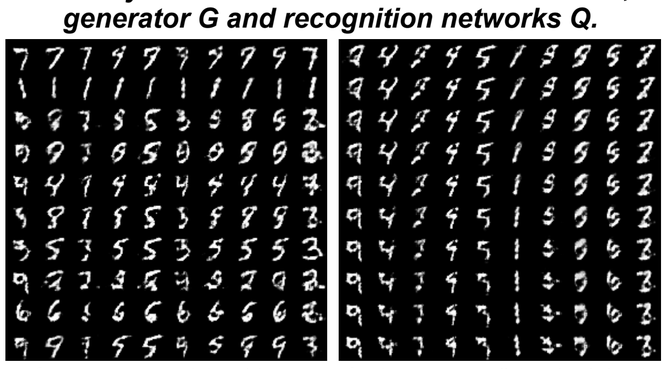


fig 1.a varying c1 (digit type)

fig 1.b varying c2 (digit rotation)

Network architecture for training MNIST

Table 1: The discriminator and generator CNNs used for MNIST dataset.

discriminator D / recognition network Q	**generator G**
Input 28×28 Gray image	Input $\in \mathbb{R}^{74}$
4×4 conv. 64 lRELU. stride 2	FC. 1024 RELU. batchnorm
4×4 conv. 128 lRELU. stride 2. batchnorm	FC. $7 \times 7 \times 128$ RELU. batchnorm
FC. 1024 lRELU. batchnorm	4×4 upconv. 64 RELU. stride 2. batchnorm
FC. output layer for D, FC.128-batchnorm-lRELU-FC.output for Q	4×4 upconv. 1 channel

These results were generated by using convolutional neural network for discriminator D, generator G and recognition networks Q.

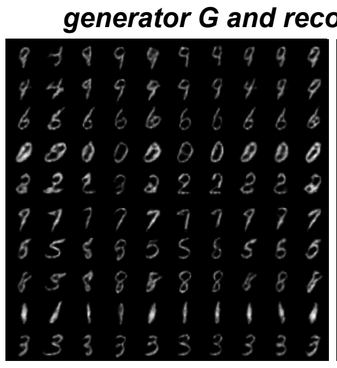


fig 2.a varying c1 (digit type)

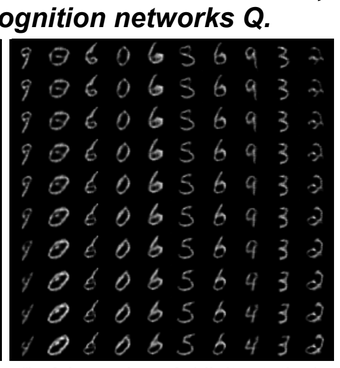


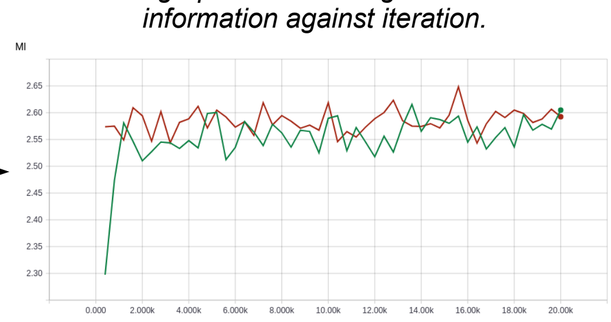
fig 2.b varying c2 (digit rotation)

Comparison of InfoGAN results using NN and CNN for MNIST dataset

	Vanilla Neural Network	***Convolution neural network***
Time per epoch	~2.5 mins	~12 min
Fig 1.a, 2.a varying c1 (digit type) Digits in every row correspond to different random samples of fixed c1 and varying c2, c3 and noise. c1 varies from top to bottom in each column while c2, c3 and noise are fixed.	• Digits are noisy. • Network hasn't learned enough to interpret latent codes properly. Hence not able to distinguish between similar digits like 3,5,8	• Digits comparatively less noisy but more blurry. • Network is able to interpret latent codes quite properly. Hence able to distinguish between similar digits like 3,5,8
Fig 1.b, 2.b varying c2 (digit rotation) Digits in every row correspond to different random samples of fixed c2 and varying c1, c3 and noise. c2 varies from top to bottom in each column while c1, c3 and noise are fixed.	• Digits are noisy and blurry. • Results showing mode collapse.	• Digits are comparatively less noisy but more blurry. • It is also showing mode collapse.

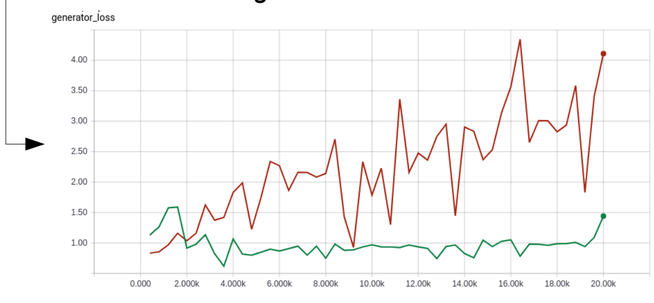
Some not so successful experiments with celebA and omniglot dataset

This graph shows change in mutual information against iteration.



This graph shows change in discriminator loss against iteration.

This graph shows change in generator loss against iteration.



References

1) https://arxiv.org/pdf/1606.03657.pdf
2) https://arxiv.org/pdf/1606.03498.pdf
3) http://slazebni.cs.illinois.edu/spring17/lec11_gan.pdf