Weekly Report（12/30/16）

Progress this week:

1) **Read Paper**

Generative Adversarial Nets

lan J.Goodfellow, good name for study.

This paper propose a new framework for estimating generative model. This adversarial process, in which simultaneously train two models: a generative model **G** that try to learn from the data distribution, and a discriminative model **D** that estimates the probability that a sample came from the training data rather than **G** model. In the process of train the data, **G** and **D** model are defined by multilayer perceptrons, the entire system only use the backpropagation and dropout algorithms. At all, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. more important, there is no approximate inference or Markov chains are necessary.

Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

This paper introduce a generative parametric model capable of producing high quality samples of natural images. They uses a cascade of convolutional networks within a Laplacian pyramid framework to generate images in a coarse-to-fine fashion.In a quantitative assessment by human evaluators, they results is better than the samples drawn from a GAN baseline model.

They apply the approach to three datasets: (i) CIFAR10 – 32X32 pixel color images of 10 different classes, 100k training samples with tight crops of objects; (ii) STL – 96X96 pixel color images of 10 different classes, 100k training samples; and (iii) LSUN - 10M images of 10 different natural scene types, downsampled to 64X64 pixels. For each dataset, they explored a variety of architectures for (**G**; **D**). For all models, the noise vector z_k is drawn from a uniform [-1,1] distribution.

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

This paper introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. What's more, they show that the generators have interesting vector arithmetic properties allowing for easy manipulation of many semantic qualities of generated samples.

Architecture guidelines for stable Deep Convolutional GANs.

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Learning to Protect Communications with Adversarial Neural Cryptography

This paper proposed a system may consist of neural networks named Alice and Bob, and they aim to limit what a third neural network named Eve learns from eavesdropping on the communication between Alice and Bob.

2) **Some Thoughts**

I was impressed by the interesting of the generate samples model. It is enjoyable to read the most cutting-edge papers.

Plan next week:

1) Finish the "See the Structure of Network"
2) Begin the "Distance Metric Learning"

This week's results:

By modifying some of the probabilities, we can get the picture of from the rules to the random in the scale-free, small world.

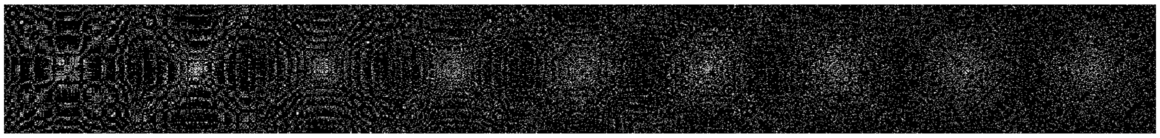

Figure 1: Small World network with the P change

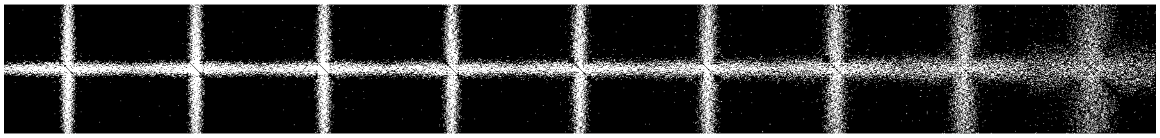

Figure 2: Scale Free network with the P change

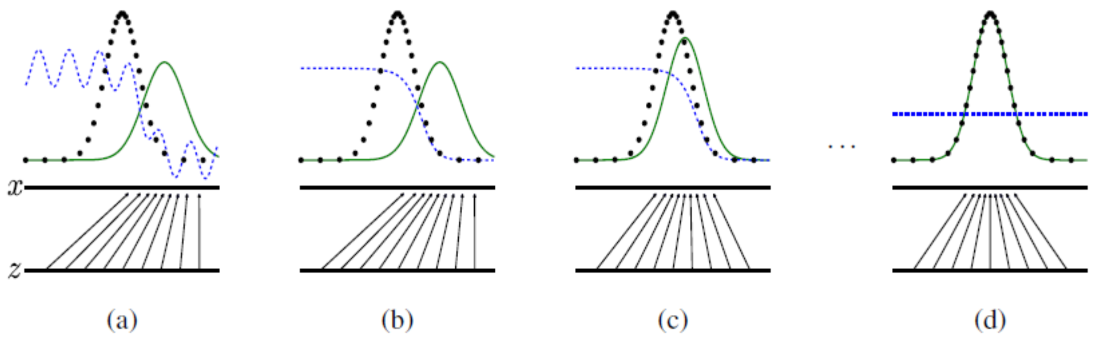

Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) p_x from those of the generative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which z is sampled, in this case uniformly. The horizontal line above is part of the domain of x. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g(x)}$. (c) After an update to G, gradient of D has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.

	Deep directed graphical models	Deep undirected graphical models	Generative autoencoders	Adversarial models
Training	Inference needed during training.	Inference needed during training. MCMC needed to approximate partition function gradient.	Enforced tradeoff between mixing and power of reconstruction generation	Synchronizing the discriminator with the generator. Helvetica.
Inference	Learned approximate inference	Variational inference	MCMC-based inference	Learned approximate inference
Sampling	No difficulties	Requires Markov chain	Requires Markov chain	No difficulties
Evaluating $p(x)$	Intractable, may be approximated with AIS	Intractable, may be approximated with AIS	Not explicitly represented, may be approximated with Parzen density estimation	Not explicitly represented, may be approximated with Parzen density estimation
Model design	Nearly all models incur extreme difficulty	Careful design needed to ensure multiple properties	Any differentiable function is theoretically permitted	Any differentiable function is theoretically permitted

Table 2: Challenges in generative modeling: a summary of the difficulties encountered by different approaches to deep generative modeling for each of the major operations involving a model.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
        • Sample minibatch of $m$ examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
              $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$$
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient:
          $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```

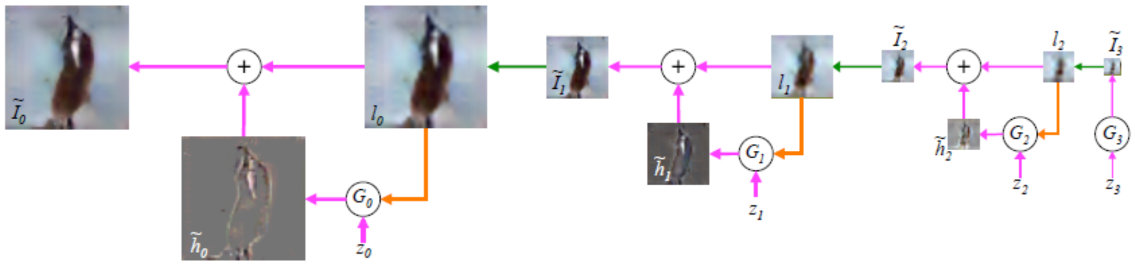

Figure 3: The sampling procedure for LAPGAN model.

Starting with a 64x64 input image I from our training set (top left):

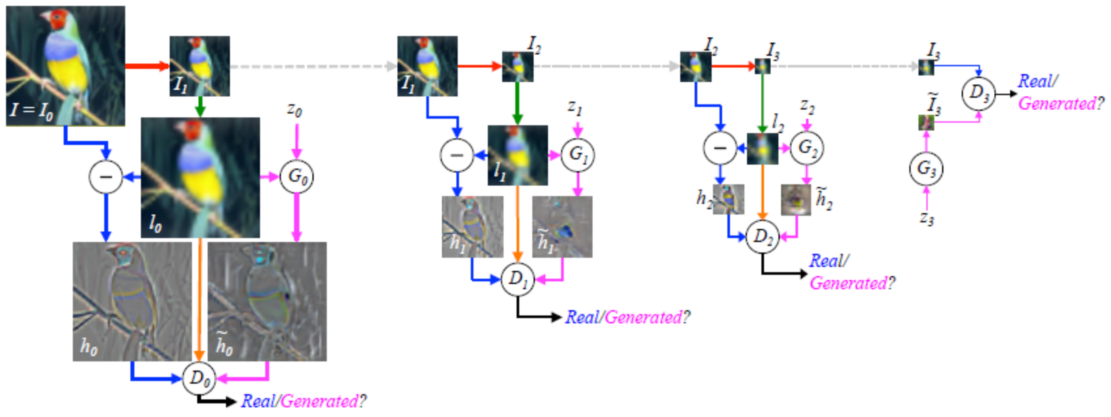

Figure 4: The training procedure for LAPGAN model.

A 100 dimensional uniform distribution **Z** is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions then convert this high level representation into a 64X64 pixel image. Notably, no fully connected or pooling layers are used.

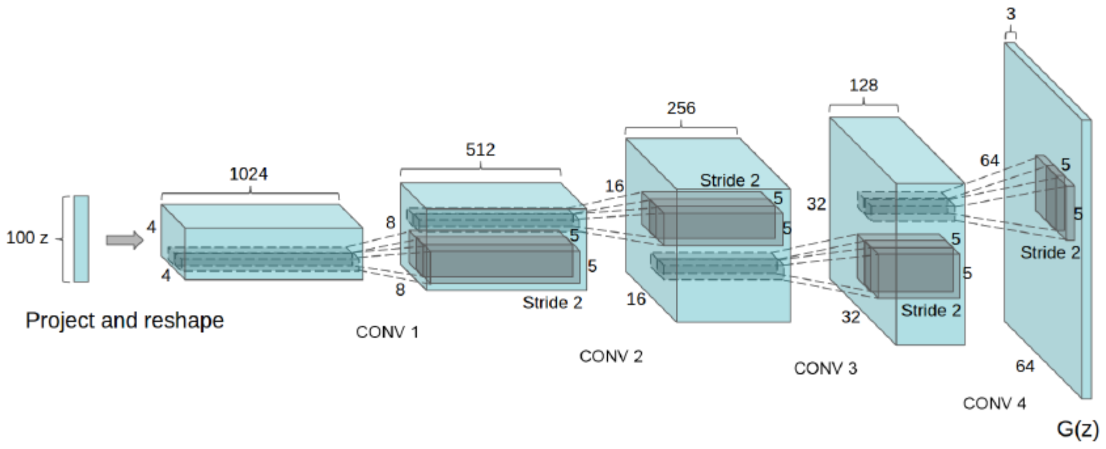

Figure 5: DCGAN generator used for LSUN scene modeling.

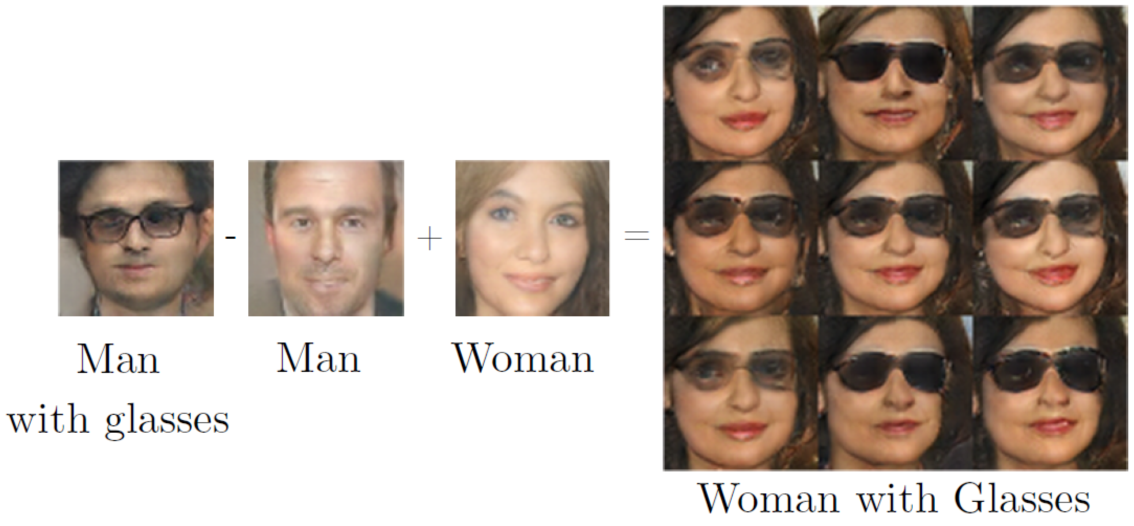

Figure 6: One demo of Vector Space Arithmetic.

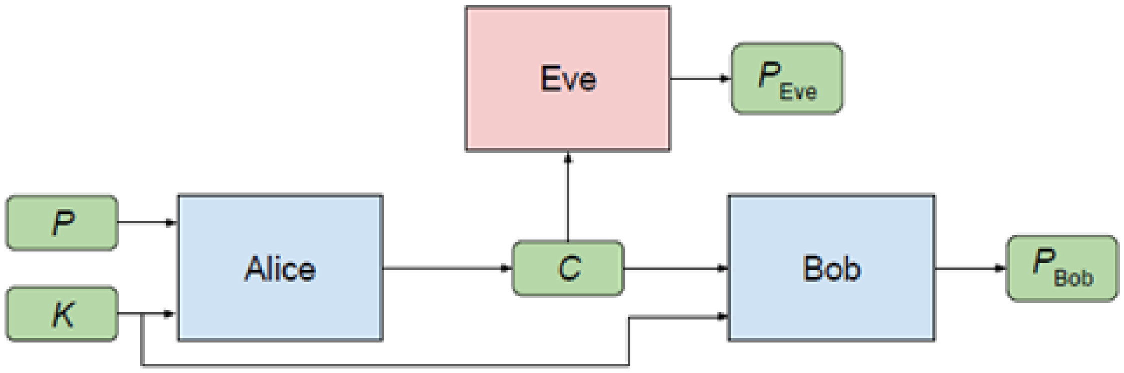

Figure 7: Alice, Bob, and Eve, with a symmetric cryptosystem.

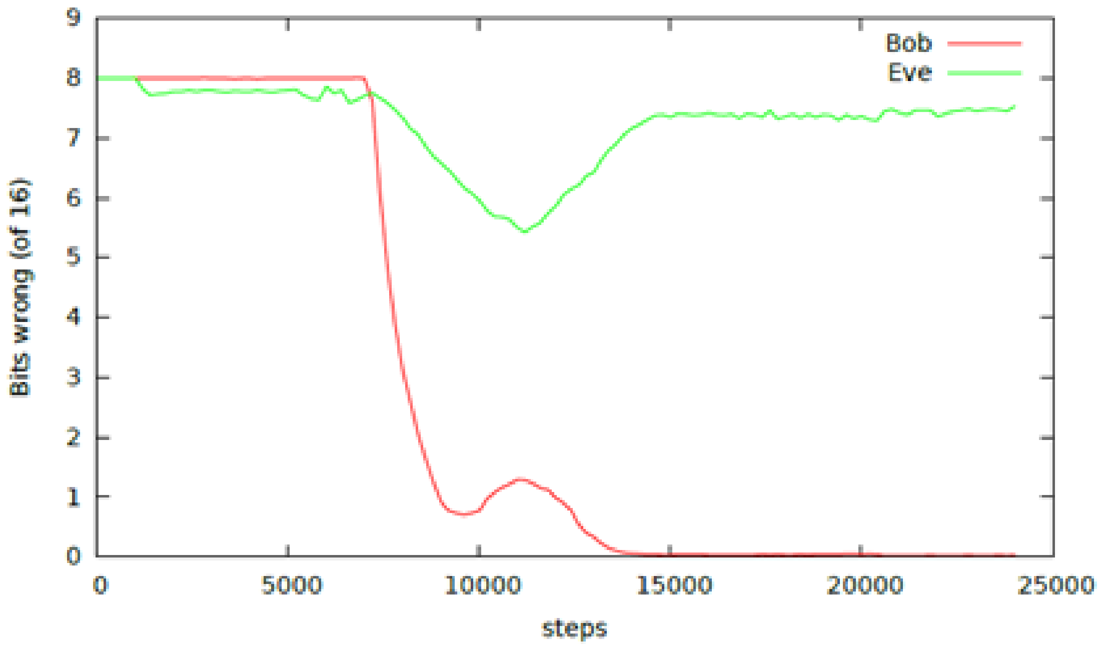

Figure 8: Evolution of Bob’s and Eve’s reconstruction errors during training. Lines represent the mean error across a minibatch size of 4096.