Generative Adversarial Nets

Ian J. Goodfellow,Jean Pouget-Abadie,Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio.

Presented By: Nimrod Shneor, CS Dept.

Agenda

- Introduction
- Develop understanding of theory
- Experiments and results
- Discussion

Generator Nets

“Generator networks are essentially just parameterized computational procedure for generating samples where the architecture provides the family of possible distributions to sample from and the parameters select a distribution from within that family.” (Y Bengio, Deep Learning)

Naive Example - Generator for the normal distributions with mean μ and covariance Σ.

We feed samples ***z*** from the normal distribution with mean 0 and the Id covariance into a generator net which on input performs the following:

$$\mathbf{X = g(z) = \mu + Lz}$$

Where L is the Cholesky Decomposition of Σ. (A decomposition a positive semidefinite matrices Calculated by ‘Cholskey’s Algorithem’ - a variant of ‘Gaussian Ellimination’.).

What about more complicated distributions you ask? We simply use Feedforward Neural Networks of course!

Learning to Generate Chairs, Tables and Cars with Convolutional Networks

Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, Thomas Brox

Abstract—We train generative 'up-convolutional' neural networks which are able to generate images of objects given object style, viewpoint, and color. We train the networks on rendered 3D models of chairs, tables, and cars. Our experiments show that the networks do not merely learn all images by heart, but rather find a meaningful representation of 3D models allowing them to assess the similarity of different models, interpolate between given views to generate the missing ones, extrapolate views, and invent new objects not present in the training set by recombining training instances, or even two different object classes. Moreover, we show that such generative networks can be used to find correspondences between different objects from the dataset, outperforming existing approaches on this task.

Index Terms—Convolutional networks, generative models, image generation, up-convolutional networks

Generative Adversarial Nets - Training Rule

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

D(x) - Probability that input **x** cam from data generating distribution.

G(z) - Generated output from input **z**.

Algorithm

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Illustration

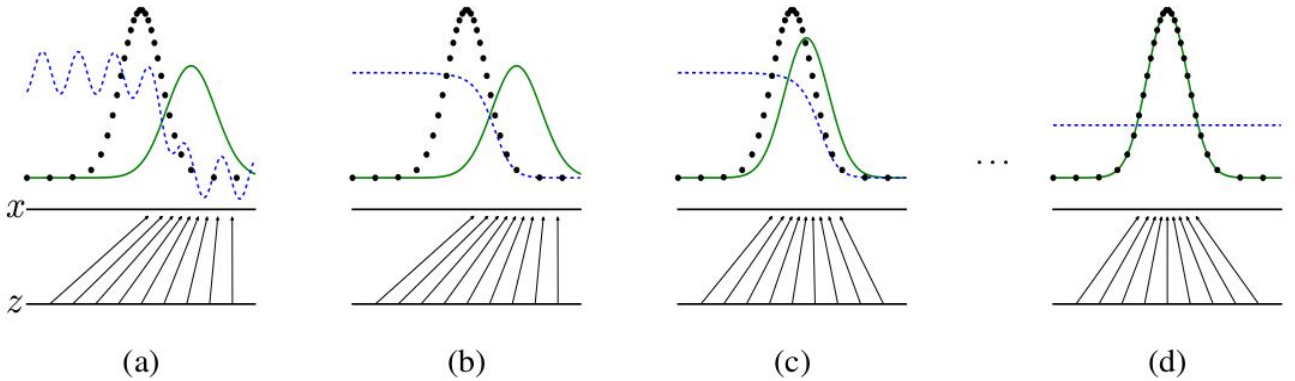


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ 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 $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ 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. $D(\boldsymbol{x}) = \frac{1}{2}$.

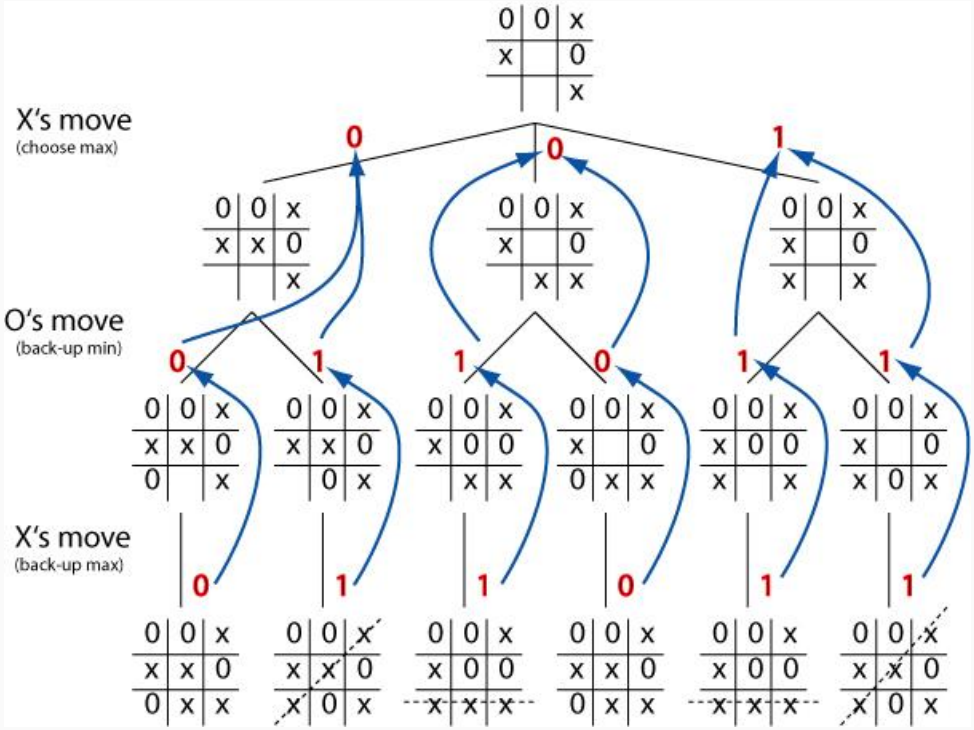
Minimax

“**Minimax** is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case *(maximum loss)* scenario.” (Wikipedia)

$$\overline{v_i} = \min_{a_{-i}} \max_{a_i} v_i(a_i, a_{-i})$$

Quick Example

The **minimax value** of a player is the smallest value that the other players can force the player to receive, without knowing his actions. Equivalently, it is the largest value the player can be sure to get when he *knows* the actions of the other players. (Wikipedia)



$$\overline{v_i} = \min_{a_{-i}} \max_{a_i} v_i(a_i, a_{-i})$$

Back to Generative Adversarial Networks

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

The two nets are playing ‘zero-sum game’ where we use the Minimax strategy to train both nets.

Theorem: Once the above expression reaches its minimum the Generator’s learned distribution is exactly the data generating distribution.

Generative Adversarial Networks - Experiments

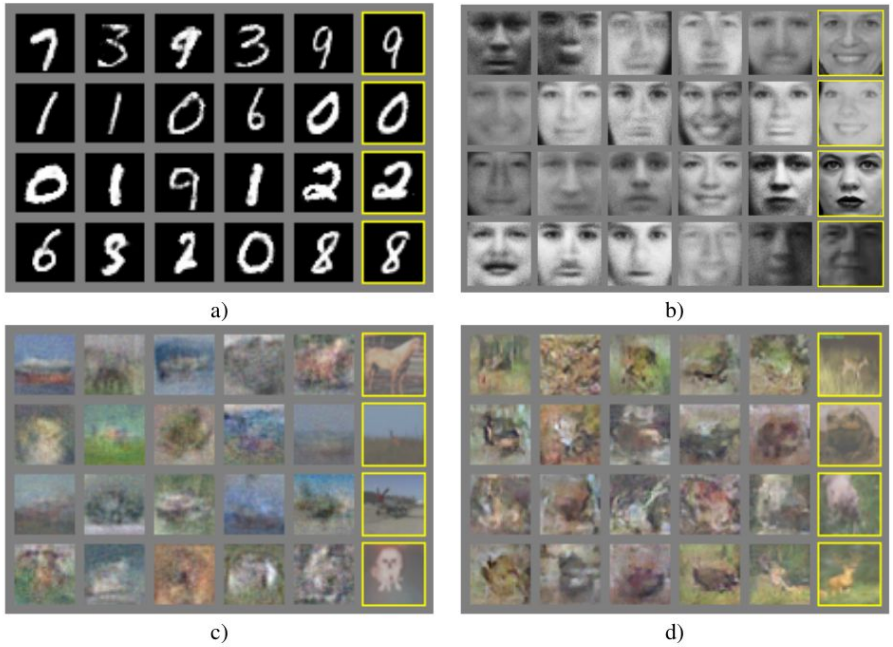


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

Discussion

- Pros:
  - Great potential for solving real problems.
  - Easy to understand and implement
- Cons:
  - Lack of theoretical background.

Thanks for listening!