Image Generative Models by GANs and DCGANs

Jiayi Ding, Lingyun Shao and Yangfan Ren

Duke University, Department of Statistical Science

Abstract

In our project, our goal is to train image generative models with GANs to generate digit images that look like hand-written digit images from MNIST database. A Generative Adversarial Network (GAN) is a model consisting of two adversarial parts: a generator that generates images and a discriminator that tries to determine whether images are real or generated. The generator portion takes random noise and tries to generate a digit that indistinguishable from actual digits. The discriminator tries to distinguish between a fake digit made by the generator and a real digit. We generate fake digits images by GANs and DCGANs and compared their performance.

Methodology

The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency[1]. In this case, the counterfeiters are our generator and the police are our discriminator.

To learn the generator's distribution p_g over data x, we need to define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z;\theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g . We also need to define a second multilayer perceptron $D(x;\theta_d)$ that outputs a single scalar. $D(x)$ represents the probability that x came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We need to simultaneously train G to minimize $\log(1-D(G(z)))$. In other words, D and G play the following two-player minimax game with value function $V(G,D)$[1]:

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

DCGAN[2] is one of the popular and successful network design for GAN. It mainly composes of convolution layers without max pooling or fully connected layers. It uses convolutional stride and transposed convolution for the downsampling and the upsampling.

Algorithm

for number of training iterations **do**
  for k steps **do**
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)},\dots,\boldsymbol{z}^{(m)}\}$ 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

Results

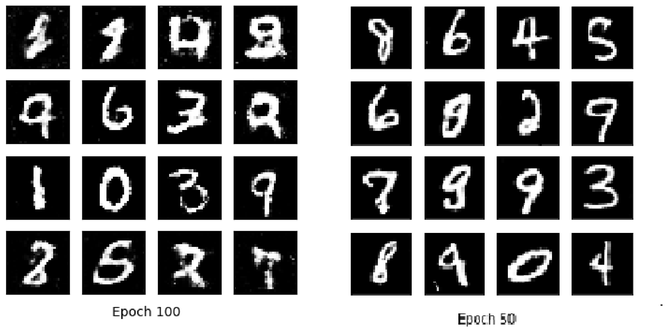

Figure 1:Generated Digits Images. Left: GANs, right: DCGANs

Above is our main result, where the left image is generated by the generator from GANs after 100 epochs and the right image is generated by the generator from DCGANS after 50 epochs. As expected, the result from DCGANs outperforms that from naive GANs (even with half the number of epochs) since we used convoluted neural networks as our discriminator and generator which are generally more powerful in image recognition problems.

As the figure shows, the generatorq from DCGANs can generate figures with clearer edges and also with more implicit meaning (from which we can tell what number it is). Generally, DCGANs would be more fitting for image data, whereas the general idea of GANs can be applied to wider domains, as the model specifics are left open to be addresses by individual model architectures.

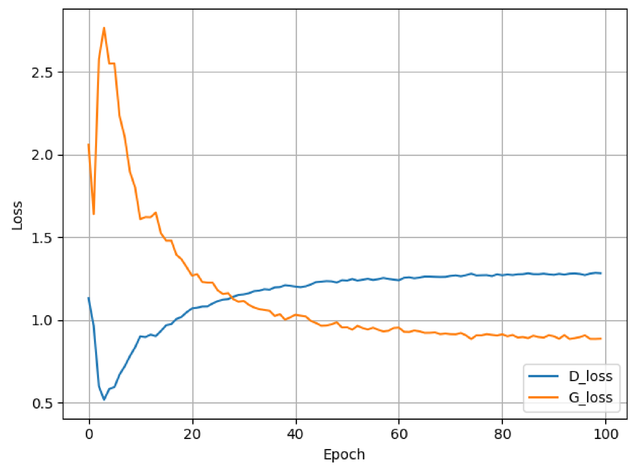

Figure 2:Loss by epoch in GANs

As the loss by epoch plot shows, the loss for discriminator and generator have converged to some stable state where the generator does a quite decent job of generating the fake data from a random noise which the discriminator can not distinguish the true data from.

Future Work

In our attempts to train image generators with GANs, only the handwritten digits images were used, which are essentially gray scale matrix data with only one color channel. But more often in real world scenarios, we would have to deal with colored images, which might require more sophisticated architecture of our generator and discriminator as well as tricks like one-sided label smoothing and reference batch normalization to get desirable results. So next, we are planning to apply GANs to colored cartoon figure images and try to train a decent generator that can create a brand new cartoon figure. Or maybe create a new superhero using the "imagination of computation", :)!

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[2] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.