Image generation: GAN vs DCGAN

Victor M. M. Alvarez [1]

Abstract

Recently, the search for creative artificial intelligence has turned to generative adversarial network (GAN), which is currently one of the most popular and successful application of deep learning. Motivated by the ability of GANs to sampling from a latent space of images to create entirely new images, here we evaluate and compare the performance of a GAN, where both generator and discriminator are multilayer perceptrons with a deep convolutional generative adversarial network (DCGAN) on the MNIST dataset.

1. Introduction

Sampling from a latent space of images to produce entirely new images is currently one of the most prominent and successful application of creative artificial intelligence. In this context, generative adversarial networks (or GANs for short) (Goodfellow et al., 2014), first introduced in 2014, have exploded in popularity as an alternative to variational autoencoders (VAEs) for learning latent spaces of images. They have been used in real-life applications for text/image/video generation, drug discovery and text-to-image synthesis.

GANs are a kind of generative model that allows us to generate a whole image in parallel, in contrast with recurrent networks where the model generates the image one pixel at a time. Along with several other kinds of generative models, GANs use a differentiable function represented by a neural network as a generator G network. The generator network takes random noise as input, then runs that noise through a differentiable function to transform the noise and reshape it to have recognizable structure. The output of the generator is a realistic image. The choice of the random input noise determines which image will come out of the generator network. Running the generator with many different input noise values produces many different realistic output images.

[1]UFPE. Correspondence to: Victor M. M. Alvarez <vmartinezalvarez88@gmail.com>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

The goal is for these images to be as fair samples from the distribution over real data. Of course, the generator net doesn't start out producing realistic images. It has to be trained. The training process for a generative model is very different from the training process for a supervised learning model. For a supervised learning model, we show the model an image of an object and we tell it, this is the label. For a generative model, there is no output to associate with each image. We just show the model a lot of images and ask it to make more images that come from the same probability distribution.

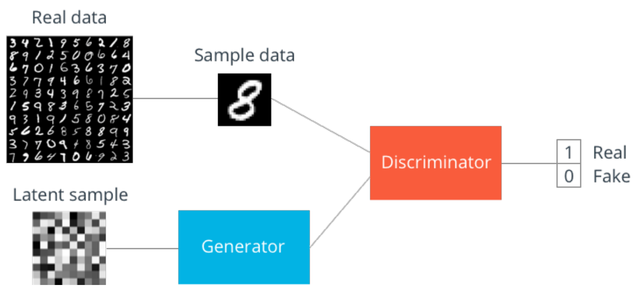

Figure 1. Scheme representing the general structure of a GAN, using MNIST images as data. The latent sample is a random vector the generator uses to construct its fake images. As the generator learns through training, it figures out how to map these random vectors to recognizable images that can fool the discriminator. The output of the discriminator is a sigmoid function, where 0 indicates a fake image and 1 indicates a real image.

But how we actually get the model to do that? Most generative models are trained by adjusting the parameters to maximize the probability that the generator net will generate the training data set. Unfortunately for a lot of interesting models, it can be very difficult to compute this probability. Most generative models get around that with some kind of approximation. GANs use an approximation where a second network, called the discriminator D, learns to guide the generator. The discriminator is just a regular neural net classifier. During the training process, the discriminator is shown real images from the training data half the time and fake images from the generator the other half of the time. The discriminator is trained to output the probability that the input is real. So it tries to assign a probability near 1 to real images, and a probability near zero to fake images (see

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter.

```
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
```

Fig. 1). Meanwhile, the generator tries to do the opposite. It is trained to try to output images that the discriminator will assign probability near one of being real. Over time, the generator is forced to produce more realistic output in order to fool the discriminator. The generator takes random noise values z and maps them to output values x. Wherever the generator maps more values of z, the probability distribution over x, represented by the model, becomes denser. The discriminator outputs high values wherever the density of real data is greater than the density of generated data. The generator changes the samples it produces to move uphill along the function learned by the discriminator (see Algorithm 1). In other words, the generator moves its samples into areas where the model distribution is not yet dense enough. Eventually, the generator's distribution matches the real distribution, and the discriminator has to output a probability of one half everywhere because every point is equally likely to be generated by the real data set as to be generated by the model. The two densities are equal.

2. Model architecture & Experiment

Motivated by the ability of GANs to sampling from a latent space of images to create entirely new images, here we evaluate and compare the performance of a GAN, where both generator G and discriminator D are multilayer perceptrons (MLP) with a deep convolutional generative adversarial network (DCGAN) (Radford et al., 2015). The experiments are performed on the MNIST dataset (Lecun et al., 1998), consisting of about 60.000 black and white images of handwritten digits, each with size 28×28 pixels. This dataset will be preprocessed according to some useful tricks proven to be useful for training GANs. The detailed descriptions about the model architectures and selected hyperparameters can be found in the colab notebook accompanying this project.

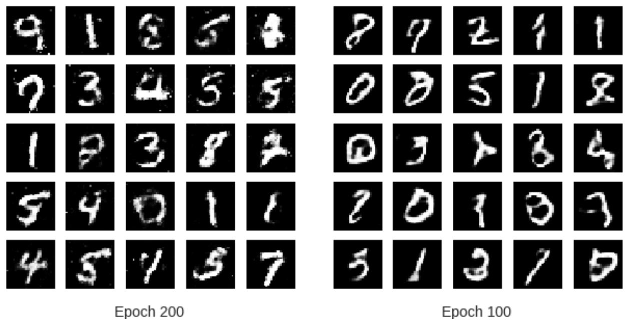

Figure 2. Generated images using a GAN (left) after 200 epoch in comparison with the generated images using a DCGAN (right) after 100 epoch.

3. Conclusions

In this short project, we have implemented, evaluated, and compared the performance of a GAN, where both generator and discriminator are multilayer perceptrons with a DCGAN. The images generated using the DCGAN model architecture were significantly better (less noisy) than the ones generates using the multilayer perceptron GAN (see Fig. 2). This can be understood as follows: convolutional neural nets, in general, find areas of correlation within an image, that is, they look for spatial correlations. This means a DCGAN would likely be more fitting for image/video data, whereas the general idea of a GAN can be applied to wider domains, as the model specifics are left open to be addresses by individual model architectures. Therefore, of the two model architectures studied here, the DCGAN model is the most appropriate tool to generate images with high resolution and with less noise.

References

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *arXiv preprint arXiv:1406.2661*, 2014.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.