Machine Learning Engineer Nanodegree

Capstone Proposal

Chuan Hong
December 1st, 2018

Proposal

Domain Background

Generative Adversarial Networks (GANs), are a framework first proposed by Ian J. Goodfellow[1] in 2014. Different from discriminative models which take an input and are trained to produce a labeled output, GANs aim to generate new data by learning from the training data via an adversarial process.

A typical GAN usually includes two separate networks being trained simultaneously: a generative model, called the generator or G, captures the data distribution and produces "fake" samples, and a discriminative model, called the discriminator or D, estimates the probability that a sample came from the training data rather than the generator. The training procedure for the generator is to maximize the probability of the discriminator making a mistake. The framework of two competing models in GANs is commonly viewed as a minimax two-player game. In a minimax two-player game, a Nash equilibrium[2] happens when one player will not change its action regardless of what the opponent may do. The GAN model converges when the discriminator and the generator reach a Nash equilibrium. Such a minimax game can be mathmatically formulated as:

$$\min_G \max_D \ \mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p(z)}\left[\log(1 - D(G(z)))\right]$$

To optimize the minimax game, the typical algorithms is to update the two networks alternatively as follows[3]:

1. update the generator (G) to minimize the probability of the discriminator making the correct choice.
2. update the discriminator (D) to maximize the probability of it making the correct choice.

The process of training a GAN, therefore, is to find a Nash equilibrium of a non-convex game with continuous, high-dimensional parameters.

Problem Statement

Unlike the discriminative models which have a well-defined loss function to minimize, finding such a Nash equilibrium could be extremely difficult. When using gradient descent techniques to train GANs, the algorithm above may oscillate, destabilize, and fail to converge for many games.[4] Other issues that GAN models may suffer include Model collapse, disminished gradient, etc.[5]

While a theoretical understanding is needed to improve the fundamental stability of GANs[6], there are a variety of approaches that might help with GAN training issues[7]. In this project we will experiment several different network architecures, cost functions, and other implementation tips on image data and compare their results.

Datasets and Inputs

GANs are notorious for being hyperparameter sensitive and usually require many epochs training on modern GPUs. Due to the limitation of hardware resources, we will first be working on some small images datasets, e.g., MNIST and Fashion-MNIST, on my desktop (Intel i5, NVidia GTX 860) for experimenting and benchmarking purposes and then move towards larger ones, e.g., CelebA using AWS GPU enabled EC2 (p2.xlarge) if the procedures are feasible.

Below is a summary of the datasets being considered for the project. We might also explore some other datasets if necessary.

1. MNIST[8]
    - The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples.
    - The digits have been size-normalized and centered in a 28x28 grayscale image.
2. Fashion-MNIST[9]
    - Fashion-MNIST consists of a training set of 60,000 examples and a test set of 10,000 examples.
    - Each example is a 28x28 grayscale image, associated with a label from 10 classes
3. CelebA[10]
    - 10,177 number of identities
    - 202,599 number of face images, and
    - 5 landmark locations, 40 binary attributes annotations per image.

Solution Statement

Based on a comprehensive literature review[11],[12], we plan to explore the following (but not limited to) approaches and compare with the original GAN:

- Network design:
  - DCGAN
  - Conditional GAN

- Loss function:
  - LSGAN(least squares loss)
  - WGAN(Wasserstein distance)

- Implementation Tips:
  - Normalize inputs
  - Sampling with Gaussian distributions
  - Batch Normalization
  - Optimize G with ADAM
  - Add labels

Benchmark Model

We will use the original GAN (also called Vanilla GAN) proposed by Ian J. Goodfellow[^i.goodfellow] in 2014 as our benchmark model. The losses for the generator and the discriminator are:

$$\ell_G = -\mathbb{E}_{z\sim p(z)}\left[\log D(G(z))\right]$$

$$\ell_D = -\mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] - \mathbb{E}_{z\sim p(z)}\left[\log(1 - D(G(z)))\right]$$

Evaluation Metrics

GANs lack an objective function, which makes it difficult to compare performance of different models[13]. Although several measures have been introduced, there is still no consensus as to which measure should be used for fair model comparison[14].

Fréchet Inception Distance (FID score) is a measure of similarity between two datasets of images. Among most of the popular measures, FID seems more plausible than others and is most often used to evaluate the quality of samples of Generative Adversarial Networks.

In this porject, in addition to directly compare the generated images by humans, we will also use FID as our evaluation metric.

Project Design

We will be using `python 3.6` and `pytorch` thoroughout the entire project. The implementations of GANs will be modularized in python scripts and the results will be evaluated using jupyter notebooks.

- data loading

  Pytorch provides datasets wrapper for MNIST and Fashion-MNIST which can be loaded very easily via `torchvision.datasets`. For CelebA or any other datasets that we may use, we will write a wrapper (inherits from `torch.utils.data.Dataset`) for them.

- network design

  We will start off with the notebook GANs-PyTorch.ipynb as our template and implement the following networks: LSGAN, DCGAN, CGAN, WGAN, and WGAN-GP

- network training and evaluation

  We will train our network on a CUDA enabled desktop computer for testing and debugging purpose. If everything works, then we will move the work to AWS and generate enough results for the final report.

Reference

1. Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In Advances in neural information processing systems, pp. 2672-2680. 2014. ↩
2. Nash, John. "Non-cooperative games." Annals of mathematics (1951): 286-295. ↩
3. Stanford CS231n Assignment #3: Image Captioning with Vanilla RNNs, Image Captioning with LSTMs, Network Visualization, Style Transfer, Generative Adversarial Networks. http://cs231n.github.io/assignments2018/assignment3/ ↩
4. Metz, Luke, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. "Unrolled generative adversarial networks." arXiv preprint arXiv:1611.02163 (2016). ↩
5. Hui, Jonathan. "GAN — A comprehensive review into the gangsters of GANs (Part 1)" https://medium.com/@jonathan_hui/gan-a-comprehensive-review-into-the-gangsters-of-gans-part-1-95ff52455672 ↩
6. Arjovsky, Martin, and Léon Bottou. "Towards principled methods for training generative adversarial networks." arXiv preprint arXiv:1701.04862 (2017). ↩
7. Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In Advances in Neural Information Processing Systems, pp. 2234-2242. 2016. ↩

8. THE MNIST DATABASE of handwritten digits http://yann.lecun.com/exdb/mnist/ ↩

9. Fashion-MNIST https://github.com/zalandoresearch/fashion-mnist ↩

10. Large-scale CelebFaces Attributes (CelebA) Dataset http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html ↩

11. Hui, Jonathan. "GAN — A comprehensive review into the gangsters of GANs (Part 2)" https://medium.com/@jonathan_hui/gan-a-comprehensive-review-into-the-gangsters-of-gans-part-2-73233a670d19 ↩

12. Hui, Jonathan. "GAN — Ways to improve GAN performance" https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b ↩

13. Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv preprint arXiv:1511.06434 (2015). ↩

14. Borji, Ali. "Pros and Cons of GAN Evaluation Measures." arXiv preprint arXiv:1802.03446 (2018). ↩