Min-Max Optimization in GANs

Team Project 8

Project Report: Min-Max Optimization in GANs

Introduction

Generative Adversarial Networks (GANs) have emerged as a powerful class of generative models, capable of producing highly realistic data samples. At the core of GANs lies a min-max optimization problem involving two competing neural networks — the generator and the discriminator. However, standard optimization techniques like gradient descent/ascent often struggle with convergence and stability in this adversarial setup.

Objective

The objective of this project is to explore and implement a more robust and convergent optimization strategy for the min-max problem in GANs, based on the recent work by Keswani et al. (2023). The project aims to evaluate this algorithm's effectiveness compared to traditional methods in training GANs.

Methodology

1. **Understanding GAN Formulation**:
   The project starts with the standard min-max formulation of GANs:

   $$min\ G\ max\ D\ E_{x \sim p\_data}[log\ D(x)]\ +\ E_{z \sim p\_z}[log(1\ -\ D(G(z)))]$$

2. **Baseline Optimizer**:
   Traditional gradient ascent-descent (GDA) is implemented to serve as a baseline for performance and convergence analysis.

3. **Advanced Optimizer Implementation**:
   A dimension-independent, convergent optimizer from Keswani et al. is implemented, which:

   - Incorporates momentum for stability.
   - Demonstrates convergence guarantees.

    - Avoids dependency on problem dimension.

4. **Evaluation Metrics**:
    - **Convergence Speed**
    - **Training Stability**
    - **Quality of Generated Images** using FID (Fréchet Inception Distance) and Inception Score.

Results & Analysis

- The proposed optimizer showed **improved convergence behavior** in comparison to standard GDA, especially in high-dimensional settings.
- **Training stability** improved significantly, with reduced oscillations and more consistent learning curves.
- **Image quality** metrics like FID indicated enhanced generation quality under the new optimizer.

Challenges faced

- Initially we were working with a batch size of 128 therefore it was taking a lot of time to generate results. We then reduced it to a batch size of 64.
- Initially our results for the algorithm were not promising. We fixed some lines of code in fid , create_generator , create_discriminator , training_gd_mx functions in our_algorithm.ipynb .

Other points to note

- The MNIST dataset is quite lightweight. A batch size of 64 should run comfortably on a google COLAB GPU , CPU .
- We have updated the results that we had uploaded in the slides. We have replaced them with the new results.You can see the images generated by both our algorithm and GDA in the slides.
- The research paper is available in the docs folder of our submission. You can refer to the algorithm there.

Summary Table :

Iteration	Algorithm (FID / IS)	GDA (FID / IS)	Observation
400	219.22 / 3.96	248.06 / 3.23	**Algorithm** better in FID and IS (both poor overall).
800	97.59 / 4.49	198.92 / 3.80	**Algorithm** outperforms **GDA** in both FID and IS.
1200	77.74 / 5.04	137.10 / 3.66	**Algorithm** significantly better in both FID and IS .
1600	72.13 / 5.52	119.62 / 4.76	**Algorithm** still leads in both FID and IS , gap remains large.
2000	63.27 / 5.17	111.76 / 5.15	**Algorithm** maintains lead in both FID and IS, though gap in IS narrows slightly.
2400	56.79/5.31	105.95/4.89	**Algorithm** continues to outperform **GDA** in both FID and IS.
2800	59.62/5.60	89.84/5.11	**Algorithm** still leads in FID and IS, gap in FID slightly increases.