Pokemon Incubator

Yice Jiang, Hongyi Jin, Jianing Sun, Lai Wei, Sen Hung Wong

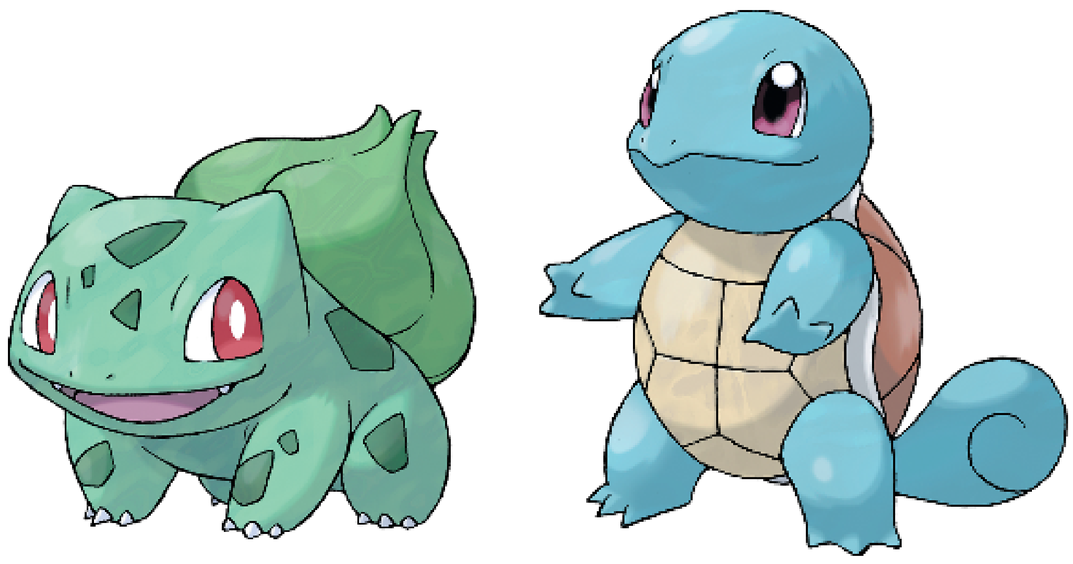
Computing Science department, SFU

Introduction

In this project we are going to create an incubator to generate new Pokemons. To do this we are going to use a deep neural network model called Generative Adversarial Network (GAN). This is a special type of generative model that can, given some set of inputs, generate totally novel outputs. GANs learn features/traits from the input data and try to produce outputs that will have similar features. We are going to generate monsters you could use in your video games, paintings or animations, which could save you hours of work. There gonna be lot of fun of doing it!

Data Set

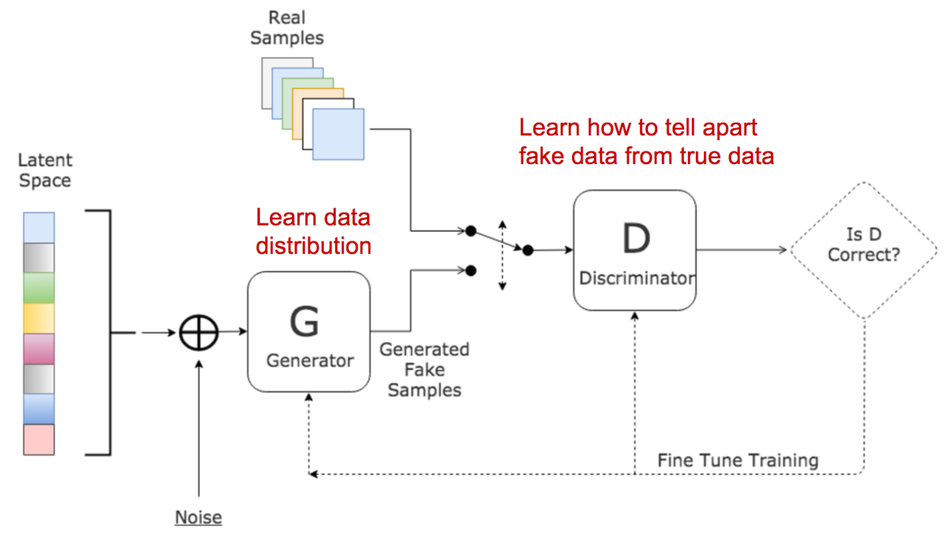
- The dataset was borrowed from https://www.kaggle.com/kvpratama/pokemon-images-dataset, which contains 721 different Pokemon images.
  - ▷ Here are two examples from our dataset.



- We also used the MNIST handwritten digits dataset at the start of experiments.

Baseline Method

- Generative Adversarial Network (GAN):
  - ▷ On one hand, we want to make sure the discriminator D's decisions over real data are accurate by maximizing E(x)[logD(x)]. Meanwhile, given a fake sample G(z),zpz(z), the discriminator is expected to output a probability, D(G(z)), close to zero by maximizing E(z)[log(1-D(G(z)))].
  - ▷ On the other hand, the generator is trained to increase the chances of D producing a high probability for a fake example, thus to minimize E(z)[log(1-D(G(z)))].



When combining both aspects together, D and G are playing a minimax game in which we should optimize the following loss function:

$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]$$

Experiments

Probelms in DCGAN:

- Hard to achieve Nash equilibrium.
- Low dimensional supports.
- Vanishing gradient probelm.
- Mode collapse.

Improved Wasserstein GAN (WGAN):

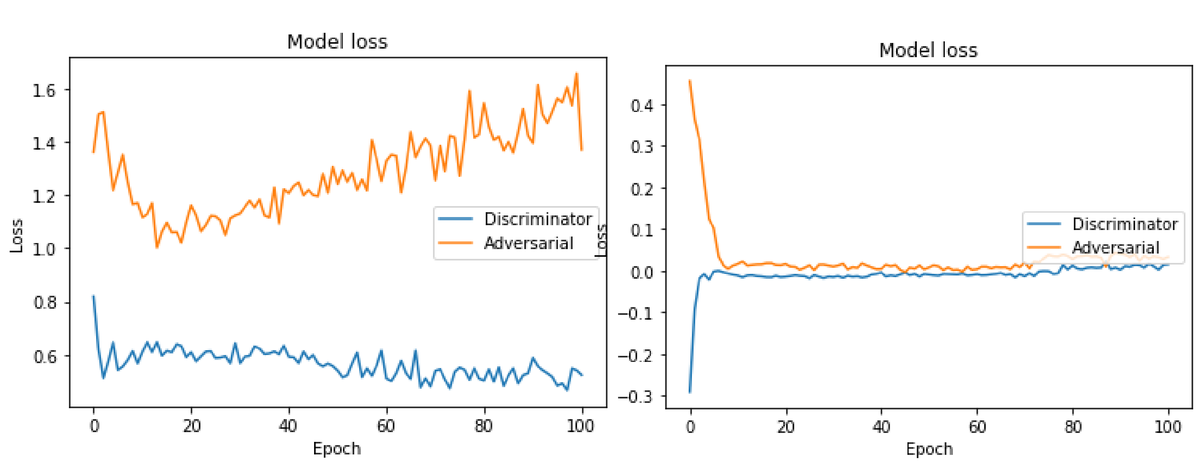
- Use Wasserstein distance as GAN loss function:

Discriminator	
GAN	$\max_D E_{x\sim p_X}[\log D(x)] + E_{z\sim p_Z}[\log(1-D(G(z))]$
WGAN	$\max_D E_{x\sim p_X}[D(x)] - E_{z\sim p_Z}[D(G(z))]$
Generator	
GAN	$\max_G E_{z\sim p_Z}[\log D(G(z))]$
WGAN	$\max_G E_{z\sim p_Z}[D(G(z))]$

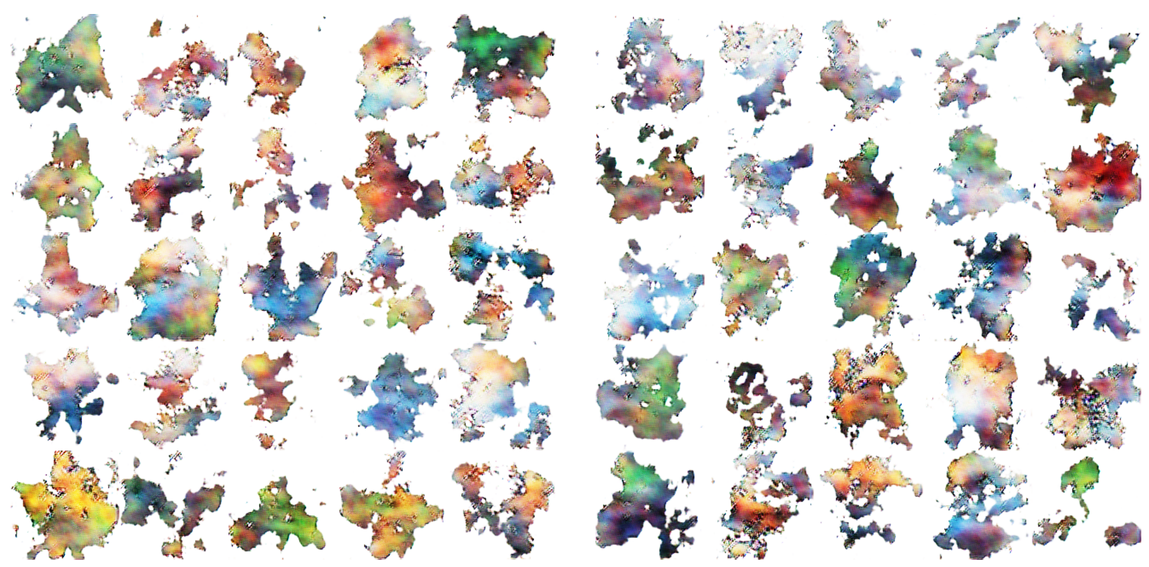
- Applies weight clipping to restrict the maximum weight value in f, i.e. the weights of the discriminator must be within a certain range controlled by the hyperparameters c:

$$w \leftarrow w + \alpha \cdot \mathrm{RMSProp}(w, g_w)$$
$$w \leftarrow \mathrm{clip}(w, -c, c)$$

GAN vs. WGAN on MNIST dataset:



Example



The two images above show the results of our model on the train epoch of 2200. They are not perfect yet, but if we increase the number of iterations as well as the number of pictures in the dataset, our Pokemons would be more vivid.

Reference

- 1. Goodfellow, Ian, et al. "Generative adversarial nets." NIPS, 2014.
- 2. Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN." arXiv preprint arXiv:1701.07875 (2017).
- 3. Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville. Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028 (2017).