# PokéGAN : Using Generative Adversarial Networks to Create Fake Pokémon

Samuel Maley
*Department of Computer and Information Science and Engineering*
*University of Florida*
Gainesville, FL
samuel.e.maley@gmail.com

***Abstract*—Previous endeavors employing generative adversarial networks (GANs) for generating new images of Pokémon have encountered challenges leading to limited success. This research endeavors to analyze the inadequacies of prior approaches and proposes a new implementation utilizing an innovative dataset. Initial outcomes reveal promising generated images, yet further investigation is warranted to fully exploit the potential of the dataset.**

***Keywords*—GAN, Convoluted Neural Network (CNN), Pokémon**

## I. Introduction

The advent of generative adversarial networks (GANs) has sparked significant interest in the realm of image generation, offering the potential to create realistic and novel visual content. Among the diverse applications of GANs, the generation of Pokémon images has garnered attention due to the immense popularity and rich diversity of the Pokémon franchise. However, past attempts to harness GANs for this purpose have encountered obstacles, resulting in limited success. In response to these challenges, this paper presents an investigation into the shortcomings of previous research endeavors and proposes a novel implementation approach leveraging a unique dataset. The primary aim is to explore avenues for improving the quality and diversity of generated Pokémon images. Initial results from our implementation exhibit promising outcomes, but further examination is required to fully exploit the capabilities of the dataset and address existing limitations in the domain of Pokémon image generation.

## II. Background

### A. Generative Adversarial Networks

Generative Adversarial Networks (GANs) represent a cutting-edge advancement in the field of artificial intelligence, particularly in the domain of generative modeling. Developed by Ian Goodfellow and his colleagues in 2014, GANs have rapidly gained prominence for their ability to generate highly realistic synthetic data, including images, audio, and text. The fundamental concept behind GANs involves a framework wherein two neural networks, the generator and the discriminator, compete in game-based learning. The generator aims to produce data samples that are indistinguishable from genuine data, while the discriminator tries to differentiate between real and fake samples. Through this adversarial training process, GANs effectively learn to capture the intricate statistical patterns inherent in the training data distribution. As a result, GANs have found wide-ranging applications across various domains, including image synthesis, data augmentation, image-to-image translation, and even in the realm of art generation.

### B. Previous Implementations

Generating fake Pokémon using a GAN has been done before. There are numerous, previous attempts that have tried to capture the essence of Pokémon. This task has been unsuccessful for two main reasons: a limited dataset and a difference in appearance of each Pokémon.

There are only 1,025 official Pokémon. For GANs, which require hundreds of thousands of images, this number is incredibily small. Previous implementations of GANs for this task used data augmentation to increase their dataset by a mutliple of 3, 4, or 5. Even with this extended dataset, the total number of training images is still only a few thousand. The implementation used in this study takes advantage of the sprites used in the populat fan game Pokémon Infinite Fusion. There are over 100,000 hand-drawn Pokémon that are used in this game. By utilizing this larger dataset and further augementing to increase its size, we hope to better capture the essence of a Pokémon sprite in our GAN.

Another difficulty apparent in this task is representing the vast differences that appear between each Pokémon. As shown in Fig. 1, there is a notable difference in the appearance of each sprite, even in the new dataset. This variance is essential to the composition of the a Pokémon, and as such, this obstacle is unavoidable.

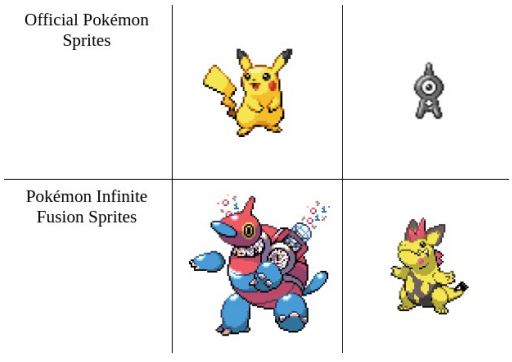


Fig. 1. Comparison of official and Infinite Fusion sprites

Identify applicable funding agency here. If none, delete this text box.

## III. Implementation

As mentioned prior, a GAN consists of two neural networks, a generator and a discriminator, which engage in a competitive learning process to improve their outputs. The parameters of each model are updated after one epoch, in which both models are trained and tested on several images. After each epoch, an output from the generator was saved. A majority of the logic used for this implementation was inspired by a previous project by Justin Kleiber, whose work can be found here.

### A. Data Processing

The training set of ~120,000 images collected from Pokémon Infinite Fusion was doubled through use of data augmentation. Pytorch transformations were appplied to the dataset to create a mirrored version, which doubled the amount of training images. For faster training, this implementation condenses the training images to 64x64. Furthermore, the training data was normalized on a scale of -1 to 1 so the models could have an easier time updating the weights.

### B. Discriminator

The discriminator network acts as a critic, receiving both real data samples from the dataset and fake samples produced by the generator. Its objective is to distinguish between real and fake data. Over time, the discriminator becomes adept at discerning genuine data from generated data.

The discriminator in this implementation utilizes five convolutional neural network (CNN) layers which take in an input of 64x64 sized images. Each output from a CNN layer (except the final layer) is normalized and activated. Batch normalization is used to adjust and scale the output from each layer. After batch nomalization, the output is passed into leaky rectified rinear unit (LeakyReLU) activation function. A slope coefficent of 0.2 was chosen to prior to training,.

### C. Generator

The generator takes random noise as input and attempts to create data samples that resemble the real data. It learns to transform the noise into meaningful data representations through a series of layers, gradually refining its output to mimic the distribution of the training data. It is worth noting that the generator has much more difficult job than the discriminator.

The architechture of the generator is similar to that of the discriminator and uses five CNN layers, batch normalization, and LeakyReLU.

We transpose the CNN layers to transform a random input vector into a complete image. While the discriminator's task is to distill an image into a singular value, denoting its authenticity, the generator operates inversely. It receives a random vector and transmutes it into a coherent image. Through these layers, the vector is progressively expanded until it ideally resembles a realistic Pokémon.

## IV. Results

Our implementation ran for 106 epochs of generator and discriminator competition. The output after the final epoch can be found below, in Fig. 2. The generator seems to be able to capture the general shape and color of a Pokémon, but it struggles to create distinguishing features such as arms, legs, faces, or any other apendages.

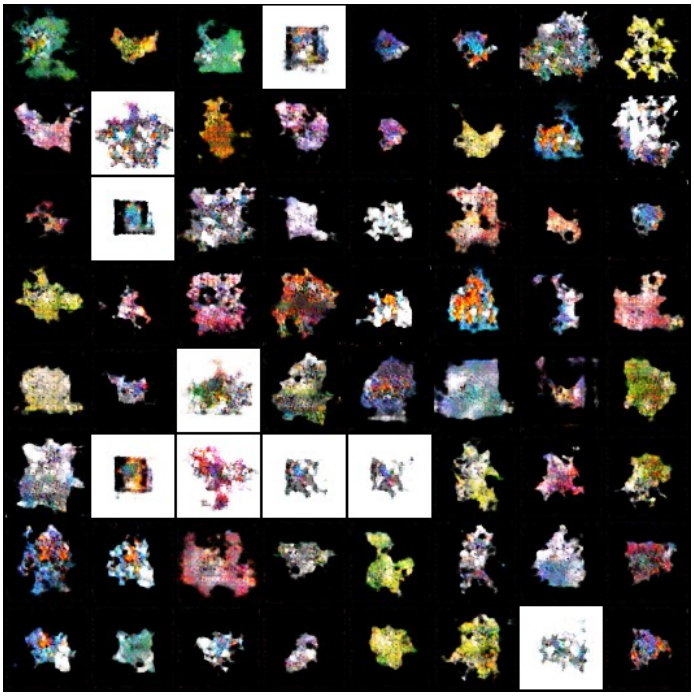

Fig. 2. Output from generator after 108 epochs (64 images)

### A. Limitations and Difficulties

Training a GAN requires a certain amount of computational resource. This project was conducted on a 2015 MacBook Air, which lacks a GPU that is commonly used in CNN training. The first major difficulty in this project was figuring out a way to train the GAN. Options such as Google Colab provide free cloud computing resources but only at a limited rate. As such, HiPerGator, the University of Florida's supercomputer, was used to train the model. 12 gigabytes of vram on a Nvidia A100 GPU was used in training, however, training a GAN can be notoriously frustrating and many difficulties found in the training stage limited the number of epochs to 106. Additionally, because much time was dedicated to finding resources to train the model, the time dedicated to training and debugging was limited. In future implementations, we would love to see this dataset used for many more epochs to see if the task of creating a functional Pokémon GAN is actually possible.