McMaster University | ENGINEERING McMaster-Mohawk Bachelor of Technology Partnership

SMRTTECH 4AI3 – Artificial Intelligence

Final Project: Generative Adversarial Network (GAN)

Instructor: Dr. Siqi Zhao

Submitted By: Jeffrey Chan (400325242)

Jay Soni (400272862)

Paul Pyrcz (400175106)

Section: C01

Date of lab: Sat December 7, 2024

Github Link: https://github.com/chanj2124/4AI3FinalProject

Introduction:

This code implements a GAN that learns to generate realistic cat images from the CIFAR-10 dataset by training a generator and discriminator in an adversarial manner. The generator learns to create cat-like images, and the discriminator learns to distinguish between real and generated images. The models are trained for 100 epochs, with periodic saving of generated images to monitor progress.

Process:

The first step to preparing the data is loading the CIFAR-10 dataset. Afterwards the dataset must be filtered for cat images only by using label 3 which corresponds to that category. The images are then normalized to the range [-1, 1] for better performance in neural networks. Next the generator model is built to take an input of a 32x32x3 image. For the convolution layer, two Conv2D layers using ReLU activation are needed to extract features, followed by a Dropout for regularization. Afterwards two dense layers are used. The first layer processes the features while the second outputs a 32x32x3 image hopefully resembling a cat.

The discriminator model similarly takes a 32x32x3 image input and follows the same convolutional layers. However, the dense layer processing the features and make use of the sigmoid activation in order to indicate if the image is real. The discriminator then outputs either a 1 or 0 in line with the result. The discriminator model is compiled using the Adam optimizer and binary cross-entropy loss due to its binary nature.

The GAN model takes both the generator and discriminator as inputs. In order to properly implement the GAN, the discriminator weights must be frozen in place. Allowing the generator to run in a sequential pattern with the discriminator. Outputting the generated AI image.

Next, there are two functions real samples and fake samples. The first takes cat images from the CIFARs-10 dataset and assigns them a label 1 as they are real. The second takes images from the dataset and passes them through the generator so that it produces a fake image which is then labeled with label 0.

The GAN is then run through a training loop. During each epoch, the discriminator is trained using both real and fake images. Meanwhile, the generator is trained to create images that can deceive the discriminator into classifying them as real. Additionally, after every 10 epochs, the results are saved locally for evaluation.

Experimental Evaluation:

During the process of building the network, we learnt multiple things along the way and also have some code error that we encountered when coding.

First of all, the project allows us to learn that GAN image building started from a noise image. The noise image is just a pixelated image is generated randomly; GAN model refines every pixel iteratively and removing the noise within step by step based on the dataset, until the detail result emerged.

Apart from what we learnt, one of the challenges faced was a reshaping issue in the discriminator. This happened when trying to move between layers with different shapes, like from convolutional layers to dense layers. For example, the output from the convolutional layers needed to be reshaped to match the input size expected by the dense layers, such as going back to a 32x32x3 format. Fixing these shape mismatches was an important part of getting the model to work.

One of the Limitations encountered was the number of epochs that could be implemented. In order to train the model with various different parameters a pitiful 100 epochs was decided upon as it took only a small amount of compute time. In order to lift this limitation, it is possible to enable hardware acceleration by using GPU hardware. However, time constraints did not permit this method to be explored.

In order to implement it the CUDA and CuDNN libraries must be added. CUDA adds what is needed to enable hardware acceleration while CuDNN is a deep learning specific library that works in Parallel with CUDA. After ensuring the proper Nvidia drivers are installed, tensorflow-gpu must be installed as well in order to enable machine-learning. Then to ensure tensorflow-gpu can access the GPU both cuda_home and path must be properly labeled.

Additionally, it was found that allowing the discriminator to learn at a higher rate than the generator helped reduce artifacts in the generated images. However, if the learning rates are too different, the model may collapse and produce outputs with minimal variance.

Results:

Images generated from several different rounds of training.

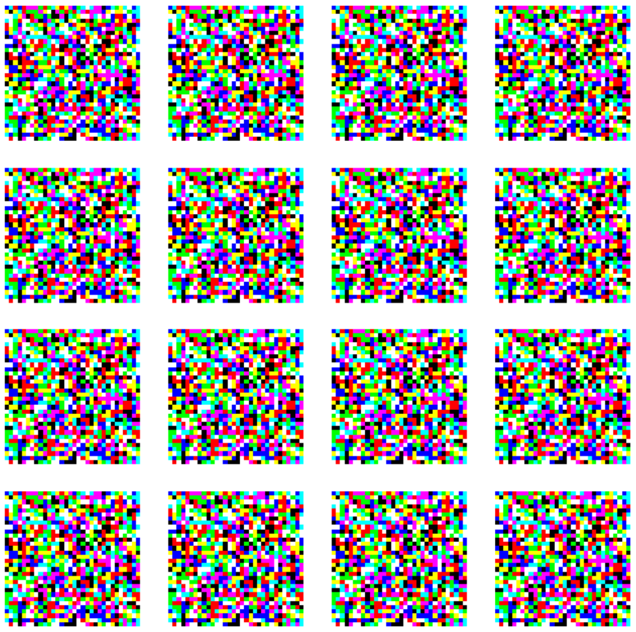

Figure 1: An example of model collapse

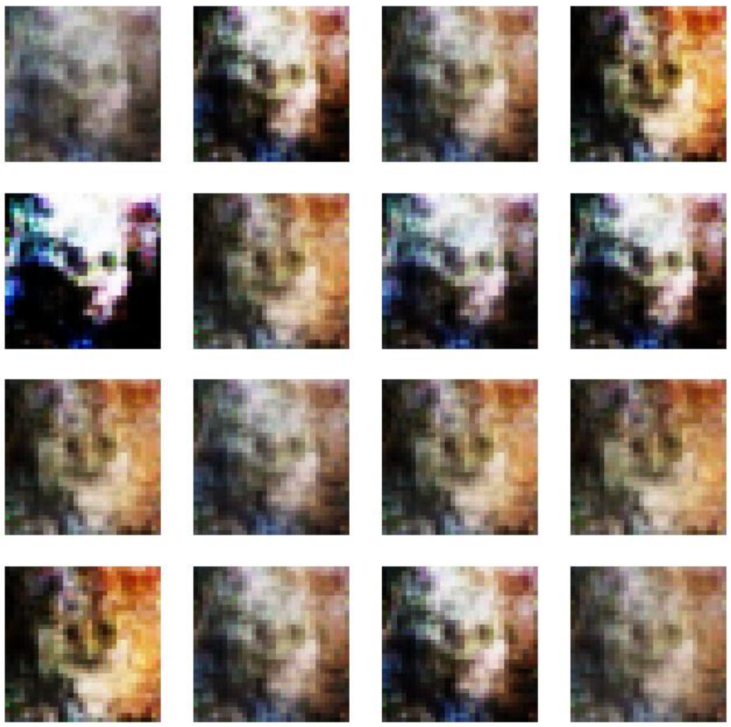

Figure 2: after sets of generate cat image from after 100 epoch

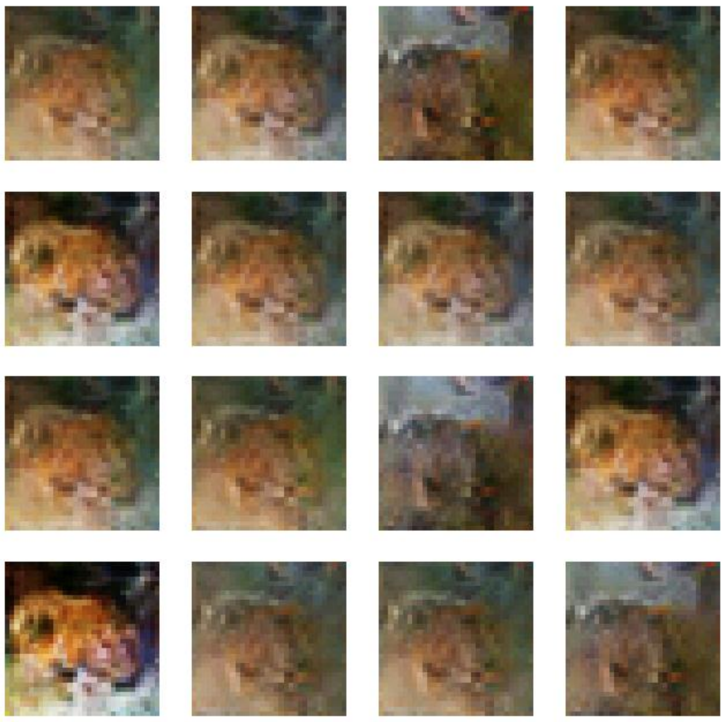

Figure 3: Another attempt on generating cat image from the dataset