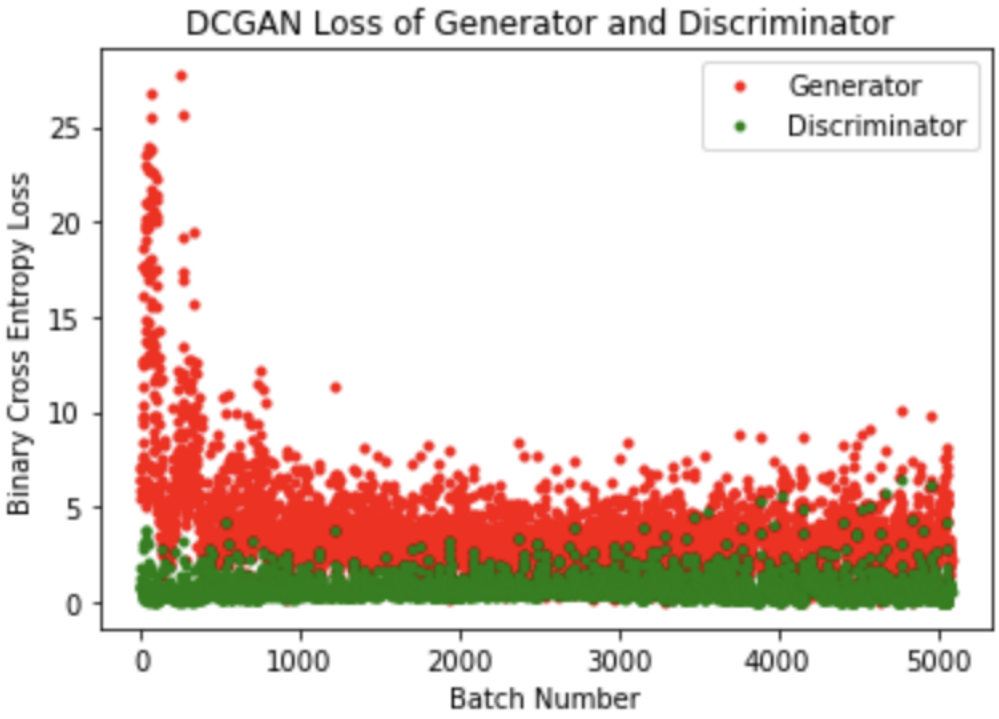
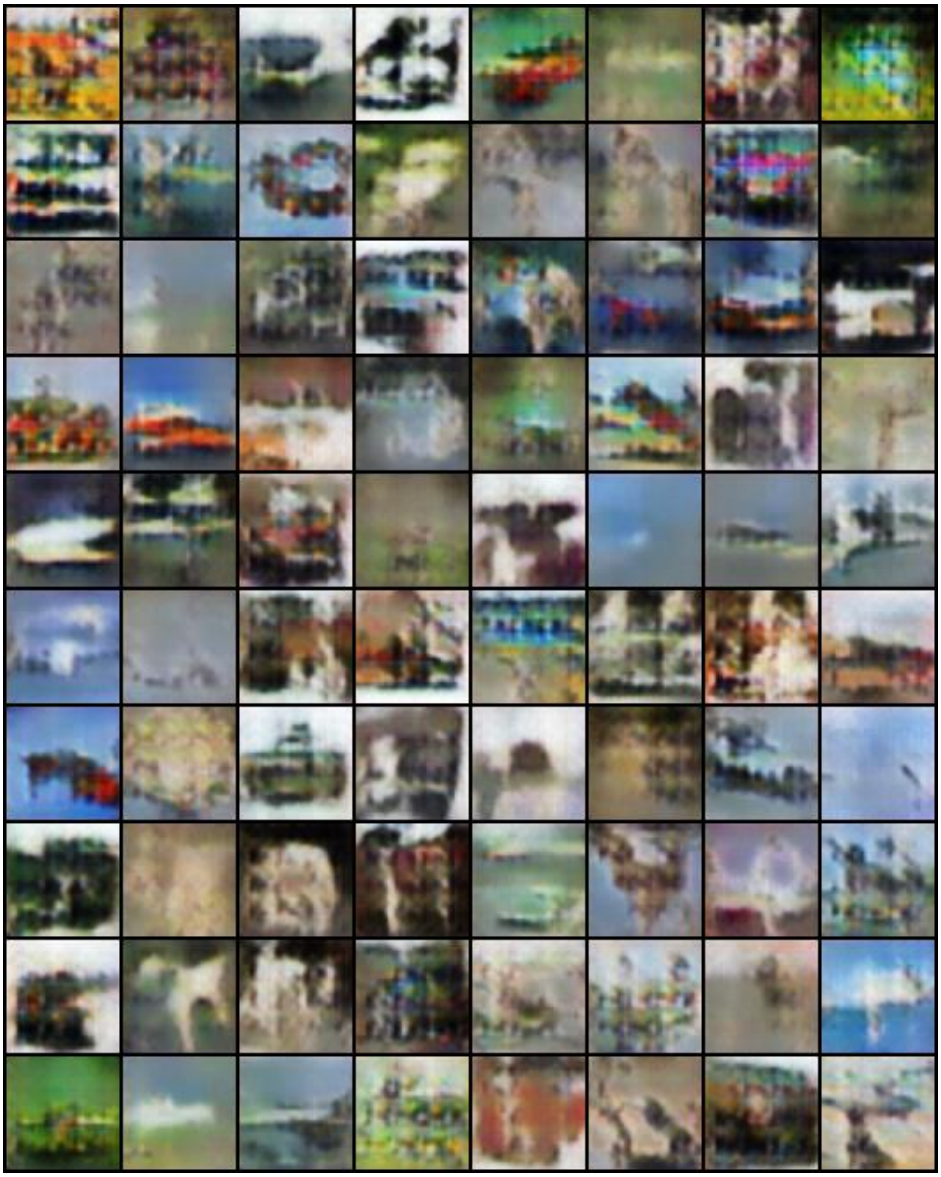
Home Work 4

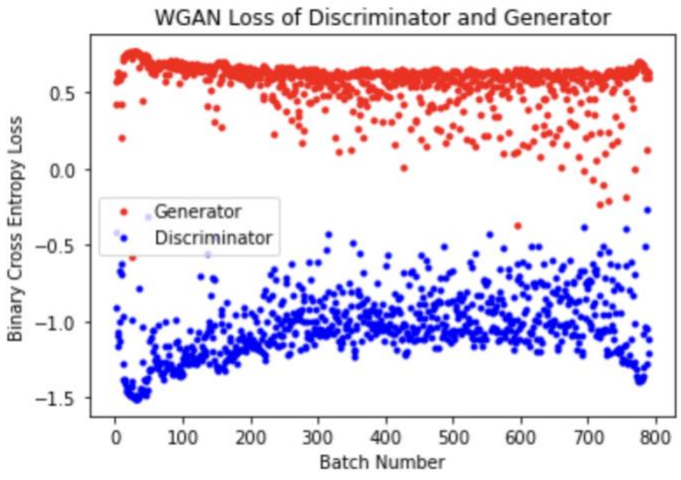
The DCGAN, WGAN, and ACGAN networks are implemented in the files in my repository.The function loops through a data loader and processes each batch of data once each epoch. Using the binary cross-entropy loss function, the discriminator is trained on both genuine and fake data. The generator is then trained using the discriminator's output in an effort to produce phony data that appears realistic. The loop also publishes the outputs of the discriminator for both actual and false data, together with the values of the discriminator and generator losses.
The code stores a collection of photos produced by the generator into a file at the conclusion of each epoch.





The Wasserstein GAN (WGAN) is trained by the code to produce realistic images. Using PyTorch, it creates a discriminator and a generator neural network and trains them on an image dataset.

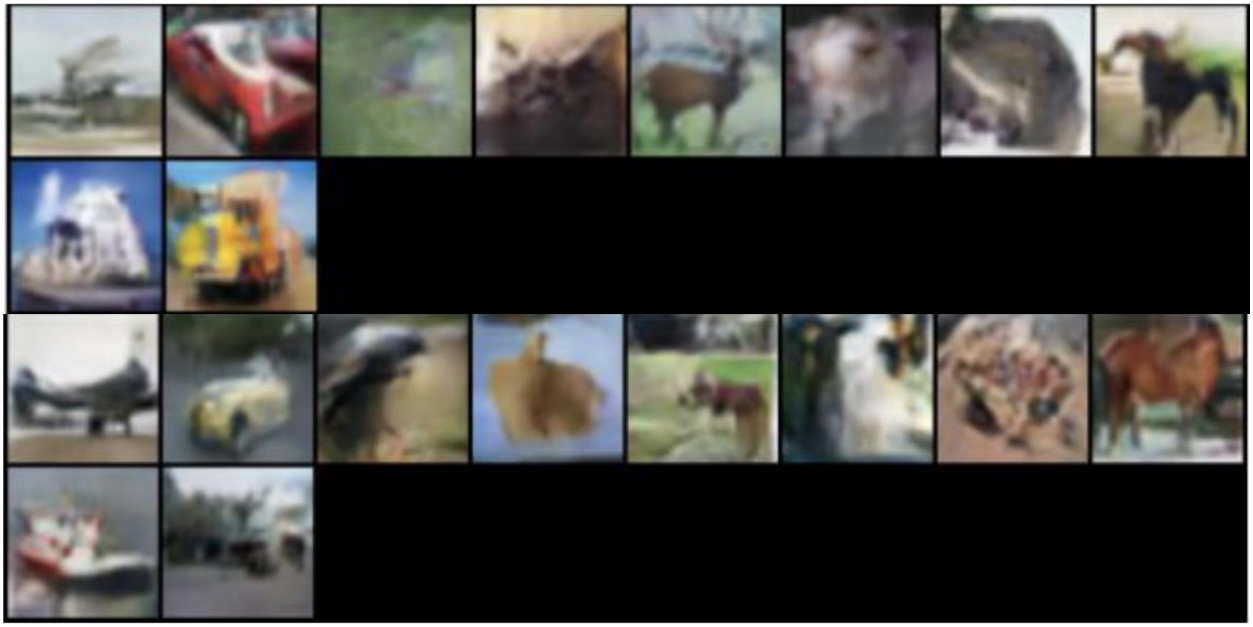
The discriminator is trained to discern between real and manufactured images, while the generator is trained to produce images that the discriminator cannot tell apart from actual ones. It use the Wasserstein distance as the loss function, trains the discriminator more than the generator (critic_iter), and clamps the weights of the discriminator network to increase training stability. Hyperparameter definition, network initialization, optimizer configuration, and model training over a predetermined number of epochs are all included in the code. The console is printed with the training's progress.

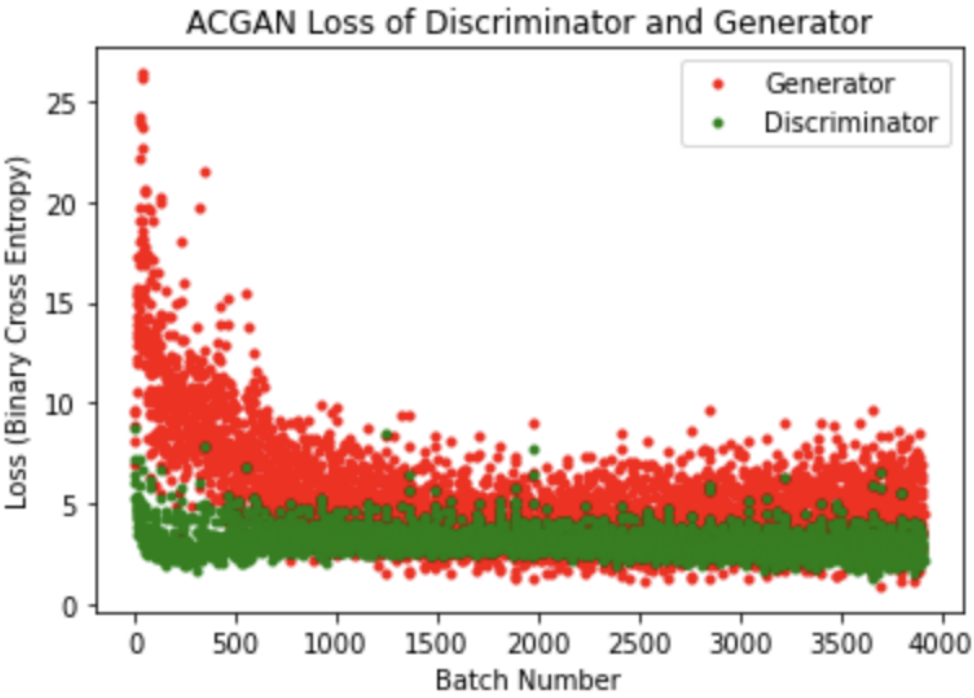


Torch, Numpy, and Matplotlib are imported first as necessary libraries. Torchvision's transformations are used to download and transform the CIFAR-10 dataset.

The photos are resized and normalized using the Compose() method. Several convolutional layers, a final sigmoid layer for binary classification, and a log softmax layer for label classification make up the discriminator network, which is a subclass of convolutional networks.

The code then selects the GPU (if available) or the CPU as the computing device. The weights_init() function is used to initialize the weights of the discriminator and generator networks. The data and image labels for each batch are fed onto the machine for processing. We produce labels for both the genuine and fake data, with the fake labels chosen at random from the possible labels. The discriminator is subsequently trained using only the bogus data, not the real data. After that, the generator is trained using the discriminator's output while being fed random noise as input and randomly produced labels for each batch. Various error terms and performance data are calculated throughout the training and printed to the terminal. The trained discriminator and generator networks are the code's final output, and they can be utilized to create new images.




ACGAN Loss of Discriminator and Generator
Generator
Discriminator
Loss (Binary Cross Entropy)
Batch Number
0
5
10
15
20
25
0
500
1000
1500
2000
2500
3000
3500
4000