Generation of Handwritten Digits using GAN

Hany Raza

Abstract

- In machine learning projects we require a huge amount of training and testing dataset to train the model efficiently.
- In this case, we can make use of the GAN model to generate more training samples.
- In this project, we are training a GAN model that will generate handwritten digits from random noise. After the training is complete, the generator learns to make data that the discriminator cannot distinguish as real or fake.
- This data then can be used as a sample to train machine learning models.

Objective

- To generate more similar data using existing data
- Create more testing and training data for other machine learning projects
- To understand the basics of test data generation using Generative Adversarial networks.
- To create numbers from noise using GANs with MNIST dataset

Generative Adversarial Networks

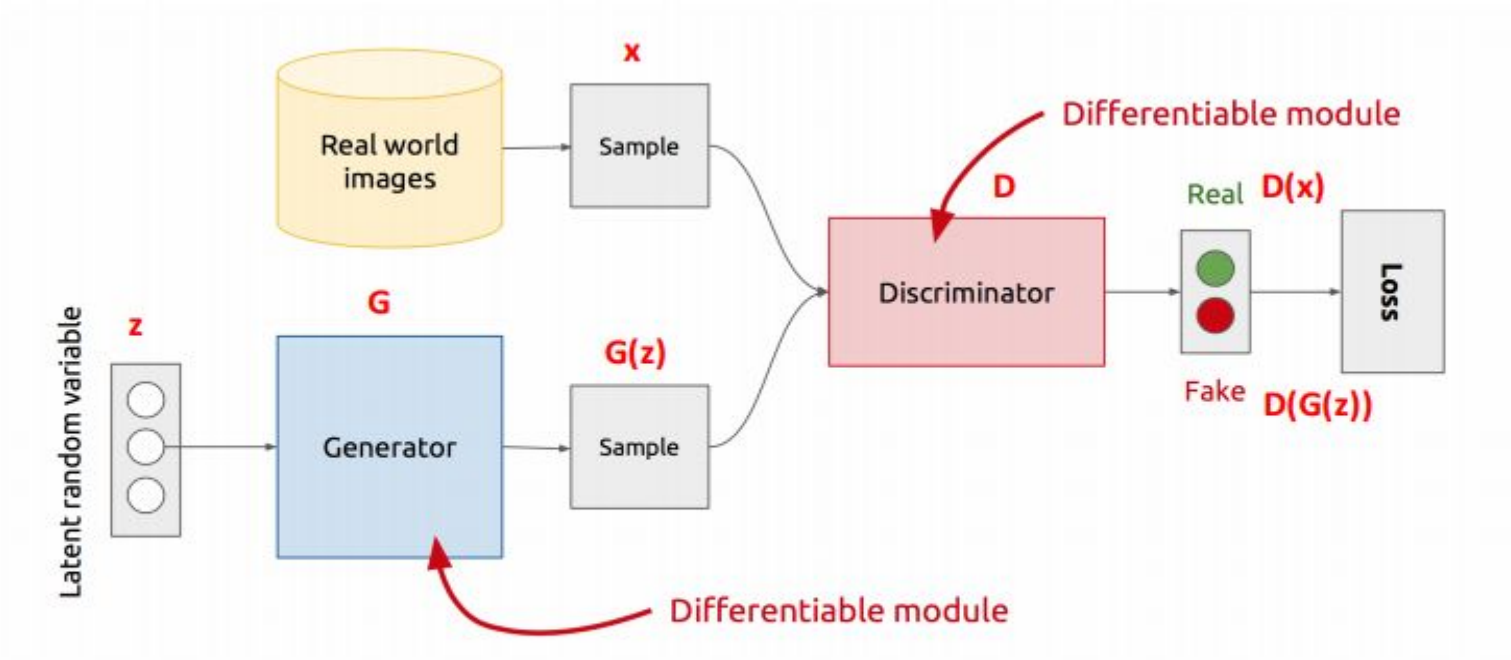
Generative adversarial networks (GANs) are algorithmic architectures that use two neural networks, pitting one against the other (thus the “adversarial”) in order to generate new, synthetic instances of data that can pass for real data.

The idea behind Generative Adversarial Nets is that you have two networks, a generator G and a discriminator D, competing against each other.

Generator — The generator makes “fake” data to pass to the discriminator i.e. the only job of the generator is to spawn ‘fake’ images that look like the training images.

Discriminator — The discriminator also sees real training data and predicts if the data it’s received is real or fake i.e. only job of the discriminator is to look at an image and output whether or not it is a real training image or a fake image from the generator.

GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Working

- The generator is trained to fool the discriminator, it wants to output data that looks as close as possible to real, training data.

- The discriminator is a classifier that is trained to figure out which data is real and which is fake.

- During training, the generator is constantly trying to outsmart the discriminator by generating better and better fakes, while the discriminator is working to become a better detective and correctly classify the real and fake images.

Working

- The equilibrium of this game is when the generator is generating perfect fakes that look as if they came directly from the training data, and the discriminator is left to always guess at 50% confidence that the generator output is real or fake.

- What ends up happening is that the generator learns to make data that is indistinguishable from real data to the discriminator.

Applications

1) Fashion, art and advertising

GANs can be used to create photos of imaginary fashion models, with no need to hire a model, photographer, makeup artist, or pay for a studio and transportation.

2) Science

GANs can improve astronomical images and simulate gravitational lensing for dark matter research. They were used in 2019 to successfully model the distribution of dark matter in a particular direction in space and to predict the gravitational lensing that will occur.

GANs have been proposed as a fast and accurate way of modeling high energy jet formation and modeling showers through calorimeters of high-energy physics experiments. GANs have also been trained to accurately approximate bottlenecks in computationally expensive simulations of particle physics experiments. Applications in the context of present and proposed CERN experiments have demonstrated the potential of these methods for accelerating simulation and/or improving simulation fidelity.

3) Video games

In 2018, GANs reached the video game modding community, as a method of up-scaling low-resolution 2D textures in old video games by recreating them in 4k or higher resolutions via image training, and then down-sampling them to fit the game's native resolution (with results resembling the supersampling method of anti-aliasing). With proper training, GANs provide a clearer and sharper 2D texture image magnitudes higher in quality than the original, while fully retaining the original's level of details, colors, etc. Known examples of extensive GAN usage include Final Fantasy VIII, Final Fantasy IX, Resident Evil REmake HD Remaster, and Max Payne.

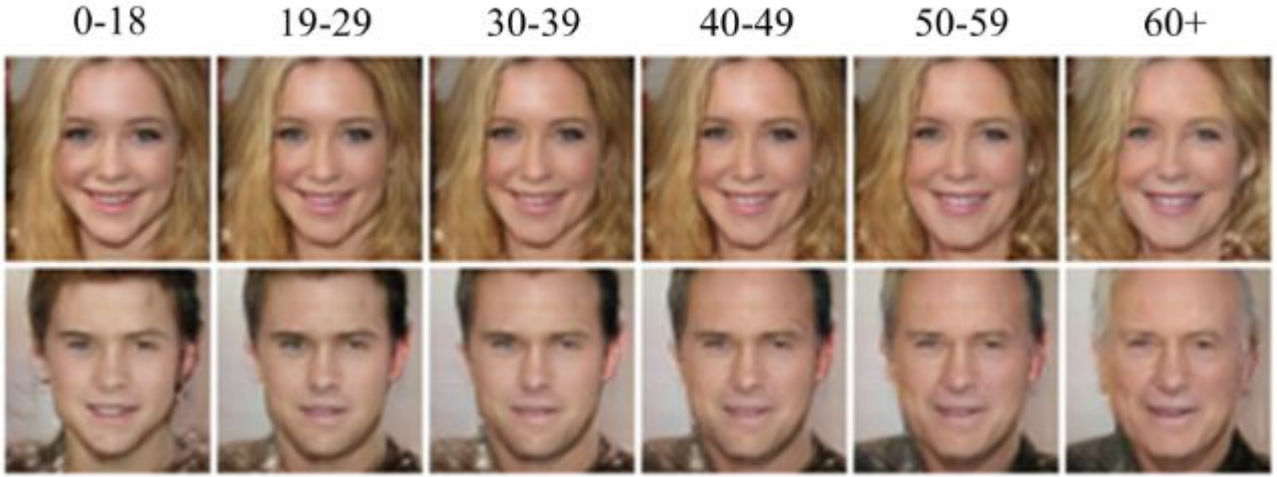
Maxpayne

4) Concerns about malicious applications:

Concerns have been raised about the potential use of GAN-based human image synthesis for sinister purposes, e.g., to produce fake, possibly incriminating, photographs and videos. GANs can be used to generate unique, realistic profile photos of people who do not exist, in order to automate creation of fake social media profiles.

5) Miscellaneous applications:

- GAN can be used to detect glaucomatous images helping the early diagnosis which is essential to avoid partial or total loss of vision.
- GANs that produce photorealistic images can be used to visualize interior design, industrial design, shoes, bags, and clothing items or items for computer games' scenes.
- GANs have been used to visualize the effect that climate change will have on specific houses.
- GANs can reconstruct 3D models of objects from images, and model patterns of motion in video.
- GANs can be used to age face photographs to show how an individual's appearance might change with age


0-18
19-29
30-39
40-49
50-59
60+

Creation of images from noise using MNIST dataset

A GAN is comprised of two adversarial networks:

1) Discriminator
2) Generator

Discriminator

The discriminator network is going to be a pretty typical linear classifier. To make this network a universal function approximator, we'll need at least one hidden layer, and these hidden layers should have one key attribute:

All hidden layers have a Leaky ReLu activation function

Generator

The generator network will be almost exactly the same as the discriminator network, except that we’re applying a tanh activation function to our output layer.

To avoid overfitting there is a dropout layer

Building the network

Now we're instantiating the discriminator and generator from the classes defined above.Output obatined is as follows

```
Discriminator(
  (fc1): Linear(in_features=784, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=256, bias=True)
  (fc4): Linear(in_features=256, out_features=1, bias=True)
)
Generator(
  (fc1): Linear(in_features=100, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=1024, bias=True)
  (fc4): Linear(in_features=1024, out_features=784, bias=True)
)
```

Training

Training will involve alternating between training the discriminator and the generator. We'll use our functions real_loss and fake_loss to help us calculate the discriminator losses in all of the following cases.

Discriminator training:

- Compute the discriminator loss on real, training images
- Generate fake images
- Compute the discriminator loss on fake, generated images
- Add up the real and fake loss
- Perform backpropagation + an optimization step to update the discriminator's weights

Generator training:

- Generate fake images
- Compute the discriminator loss on fake images, using flipped labels!
- Perform backpropagation + an optimization step to update the generator’s weights

Output

Result

- Results after 200 epochs is that the discriminator loss is higher than generator. But the images generated are recognizable. In the beginning the images generated were pure noise.
- But as the training continues the images generated are better.
- The images generated are now recognizable as 'handwritten digits'. Here we have generated 100 images, out of which 95 are recognizable.
- The efficiency of the model can be increased by increasing the number of epochs or playing with different parameters like batch size, dropout probability, learning rate and discriminator optimizer.

References

- **Generative Adversarial Networks**

  https://arxiv.org/abs/1406.2661

- **Generative Adversarial Networks for Classification**

  https://ieeexplore.ieee.org/document/8457952

  https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-an-mnist-handwritten-digits-from-scratch-in-keras/

Literature Survey

Paper1 -Generative Adversarial Networks for Classification

- This paper highlights on the problem of classification.Generative Adversarial Networks(GANs) have been shown to reduce training requirements for detection problems.
- GANs compete generative and discriminative classifiers to improve detection performance.
- his paper expands the use of GANs from detection (k=2) to classification (k>2)problems.
- Several GAN network structures and training set sizes were compared to the baseline discriminative network and Bayes' classifiers. The results show no significant performance differences among any of the network configurations or training set size trials.

Paper 2 - Generative Adversarial Networks

- The paper proposes a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G.
- The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game.
- In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 1/2 everywhere.

- In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation.
- There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples.
- Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

Thank You