Quantitative Evaluation of GAN

Team 9
Vamshi Saggurthi - vs734
Vignesh Rajesh Singh -vrs60
Qingyang Yu - qy116
Chaitanya Vallabhaneni - cv346

Abstract

The first task is to build a GAN model from image to image without any label so that we can capture the features of the images concisely by the data and explore the inner working theory of mapping function. The GAN model has to be trained on two different set of real and synthetic pizza datasets. To qualitatively evaluate the generated images we are using FID (Frechet inception distance) and Inception score metrics on the generated images and the images in the training dataset.

1. Introduction

GANs are a clever way of training a generative model by framing the generative problem as a supervised learning problem. It consists of two sub-models:-

1. The generator model takes a fixed-length random vector(usually random gaussian distribution) as input and generates an image.The goal of generator is to eventually generate images that are considerably closer to the real images, thereby fooling the discriminator.

2. The discriminator model takes a sample image as input (real or generated) and predicts a binary class label of real or fake. We use real images form training dataset and fake images from generator to train the discriminator. The discriminator is only used for training.

The two models are trained together in a zero-sum game, both try to beat/fool each other adversarial until the discriminator model until the discriminator model is fooled about a certain percent of times. We can consider this to be 50% (half) of generated examples, although this is not always the case.

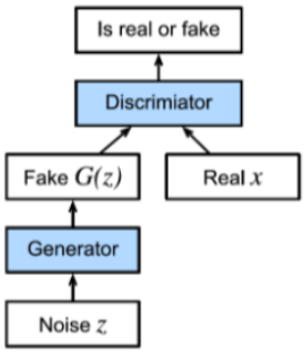

Figure 1. Basic Structure of GAN

2. Model Architectures

The Paper specifies Generator and Discriminator Architectures as below

2.1. Generator

- The generator model is based on the architecture created by J. Johnson, A. Alahi and L. Fei-Fei.
- We use 9 residual blocks for 256 ×256 training images.The network with 9 residual blocks consists of:-

  c7s164,d128,d256,R256,R256,R256,R256,R256,
  R256,R256,R256,R256,u128,u64,c7s1

2.2. Discriminator

- For discriminator, we used the PatchGAN model created by P. Isola, J.-Y. Zhou, and A. A. Efros.
- A regular GAN discriminator reduces an input image of dimensions 256*256 to a single Scalar output ,which can be classified as 0(generated) or 1(real), but the PatchGAN artitechture used here maps the image to a N*N patch from 256*256.

- For discriminator networks, we use 70 ×70 PatchGAN.We use leaky ReLUs with a slope of 0.2.
- The discriminator architecture is: C64 C128 C256 C512.

3. Dataset

We have 2 different data sets for Real and Synthetic. The training/test size of each of them is.

- Real Pizza - 6500+ train images/ 836 test images
- Synthetic Pizza- 13,321 train images/ 1,538 test images

3.1. Data Preprocessing and loading

Transforms :-

1. We resize the image to required 256*256 since some images may not suit our exact dimensions.
2. Center crop the image.
3. Load it to a Tensor
4. Normalize the values (0.5,0.5,0.5)

We use the dataloader of pytorch and batch size of 24 for both Real images and Synthetic images for training(we parallelized processing on 4 GPUS).

4. Implementation/Workflow of program

4.1. Task 1

1. Initialize all the variables and paths, includes setting up the loss function, batch size, epochs, learning rate,etc.
2. Importing of train images, pre-processing them and loading them using Pytorch DataLoader.
3. Discriminator and Generator Artitechtures.
4. Discriminator and Generator Training Functions that calculate losses, scores and propogate them back.
5. Functions to display and save generated images.
6. Training loop which calls Discriminator and Generator Training on the every batch of input for 50/150 epochs(real/synthetic). This loop also saves the model of every epoch which will be used to generate scores in Task 2.

4.2. Task 2

1. Initialize all the variables and paths, includes setting up the loss function, batch size, epochs, learning rate,etc.
2. Importing of test images, pre-processing them and loading them using Pytorch DataLoader.
3. Discriminator and Generator Architecture.
4. Discriminator and Generator Training Functions that calculate losses, scores and propagate them back.
5. Training loop which calls loads the models that are saved in the Task 1 and calculates FID and Evaluation scores on them. These scores are collected using Tensorboard scalars.

5. Loss functions and Scores

1. Generator Loss
   - This loss is the measure of how good is a generated image in fooling the discriminator.
   - We use Binary Cross Entropy(BCE) loss function to calculate the difference between expected discriminator output(all 1) and actual discriminator output.
   -

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Figure 2. BCE Loss

2. Discriminator Loss
   - This loss is the measure of how good is a discriminator in classifying an image as real or generated. It consists of two values, one for images from dataset and other for generated images.
   - We use Binary Cross Entropy(BCE) loss function to calculate both the losses against the expected output.
3. Real Score
   - This a measure of how good is a discriminator in classifying an image from dataset as real.
   - We use the Sigmoid function on the discriminator output and calculate mean. As the model gets better this value should be moving closer to one.
4. Fake Score

- This a measure of how good is a discriminator in classifying a generated image as fake.
- We use the Sigmoid function on the discriminator output and calculate mean. As the model gets better this value should be moving closer to zero.

6. Evaluation Metrics

There is no objective way to evaluate the quality of the generated images, generally the generated images are saved and evaluation is done by humans to evaluate the quality of image and select the final generative model. We use FID scores and inception sores which are 2 of the metrics that can help overcome this task.

1. Inception Score:-

   - The Inception Score is an objective metric for evaluating the quality of generated image.
   - The generated images are then divided into n sets and sent to InceptionV3 model which returns the probability of each image belonging to each class.
   - We then find the KL divergence on this classification between a set of images to find how different they are.
   - In the end we return the mean and standard deviation of these scores. The higher the mean scores, the better are the generated images.
   - These scores depend on the classification by InceptionV3 model which has 1000 classes, so if we try to find IS on generated images that do not belong to these 1000 classes it will return lower values even when quality of image generated is very high.

2. FID (Frechet inception distance):-

   - The FID is a metric that calculates the distance between feature vectors calculated for real and generated images.
   - Unlike IS which evaluates only the distribution of generated images, the FID compares the distribution of generated images with the distribution of real images that were used to train the generator.
   - In FID lower scores indicate the two groups of images are more similar, or have more similar statistics, with a perfect score being 0.0 indicating that the two groups of images are identical.
   - Although FID is more closer to human judgement compared to inception score, there are cases where FID is inconsistent with human judgment.

7. Sample Images:-

1. Real Pizza Images:-

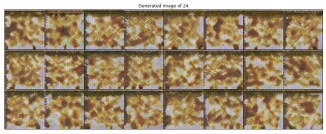

Figure 3. Real Images at Epoch 24

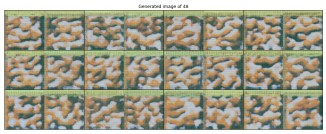

Figure 4. Real Images at Epoch 48

2. Synthetic Pizza Images:-

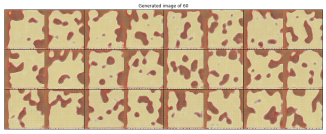

Figure 5. Synthetic Image at Epoch 60

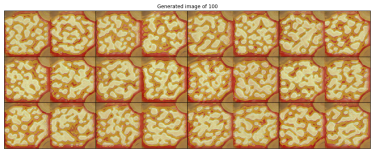

Figure 6. Synthetic Images at Epoch 100

8. Results:-

1. Real Pizza Images:-

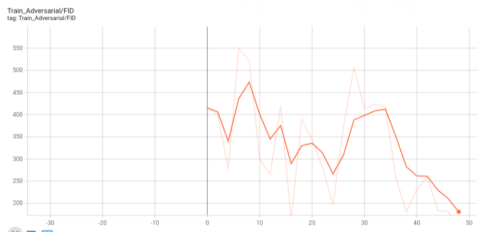

Figure 7. Real Images FID

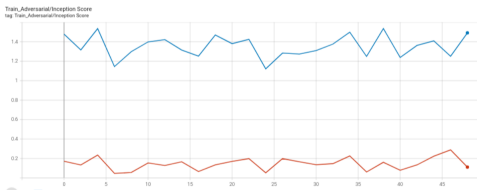

Figure 8. Real Images Inception Score

2. Synthetic Pizza Images:-

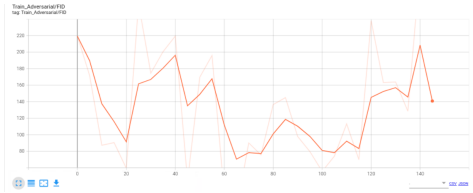

Figure 9. Synthetic Images FID

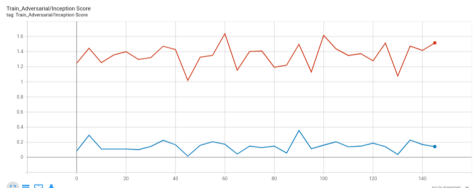

Figure 10. Synthetic Images Inception Score

9. Conclusion:-

1. Real Images:-
   - For **real images** we trained for 50 epochs and we can see that after some minor fluctuations till 30 epochs, the FID scores drop sharply to around '146' at 48th epoch, thus we can conclude that we would get better results if we trained for more epochs.
   - The mean inception score for our set oscillates but we find it to be maximum at 48th epoch which is inline with the FID scores.
2. Synthetic Images:-
   - We trained for 150 epochs on **synthetic images** data-set. We can see that after initial fluctuation we reach the minima of FID score around epoch 60 after which model performs significantly worse and reaches the level that it began from.
   - Similarly the mean inception score although has fluctuations but the maximum is around the point of 60th epoch which agrees with our FID scores.
   - This constant increase in the FID value after 70th epoch could be a result of over-fitting of our model to the train data and thus it performs poor on test set.