Qualitative and Quantitative Evaluation of GANs on Pizza Dataset

Adithya Shrivastava
as3652
Rutgers University
Department of Computer Science
as3652@scarletmail.rutgers.edu

Nishant Dhargalkar
nsd78
Rutgers University
Department of Computer Science
nsd78@scarletmail.rutgers.edu

Janish Parikh
jrp328
Rutgers University
Department of Computer Science
janish.parikh@rutgers.edu

Jiawei Tang
jt1039
Rutgers University
Department of Computer Science
jiawei.tang@rutgers.edu

Abstract

Generative Adversarial Networks (GANs) are deep-learning based generative models. They can be used for a wide variety of purposes like style transfer, photo blending, image-to-image translation, etc. In this task, we aim to use GAN training real pizza and synthetic pizza datasets so that we can generate our own synthetic pizza.

1. Introduction

Generative adversarial networks (GANs) are a generative model with implicit density estimation, part of unsupervised learning and are using two neural networks. Thus, we understand the terms "generative" and "networks" in "generative adversarial networks". This report displays our work for step 1.

2. Pre-processing

2.1. Datasets

The datasets included in this task are downloaded from https://drive.google.com/drive/folders/1jaag8cyb72bFHSA-V8ckieTL9uAOsK7r?usp=sharing [4]. We train models on two datasets: SyntheticPizza 2 and RealPizza datasets 2.1. The SyntheticPizza images and RealPizza images are resized to 256 × 256 pixels and normalized to $[-1, 1]$.

Figure 1. Real Pizza Images

Figure 2. Synthetic Pizza Images

3. Implementations

3.1. Generator architectures

We adopt the architecture from [3]. Use 9 residual blocks for 256 × 256 training images. Let $c7s1-k$ denote a 7 × 7 Convolution-InstanceNorm-ReLU layer with k filters and stride 1. dk denotes a 3 × 3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. Reflection padding was used to reduce artifacts. Rk denotes a residual block that contains two 3 × 3 convolutional layers with the same number of filters on both layer. uk denotes a 3 × 3 fractional-strided-ConvolutionInstanceNorm-ReLU layer with k filters and stride $\frac{1}{2}$ The network with 9 residual blocks consists of: $c7s1-64$, $d128$, $d256$, $R256$, $R256$, $R256$, $R256$, $R256$, $R256$, $R256$, $R256$, $u128$, $u64$, $c7s1-3$.

3.2. Discriminator architectures

We use 70 × 70 PatchGAN[2]. Let Ck denote a 4 × 4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. After the last layer, we apply a convolution to produce a 1-dimensional output. We do not use InstanceNorm for the first $C64$ layer. We use leaky ReLUs with a slope of 0.2. The discriminator architecture is:
$C64$ - $C128$ - $C256$ - $C512$

3.3. Training

Repeat step 3.3.1 and 3.3.2. The training of a GAN can be understood better from the below image.
Here we are generating a random noise z.

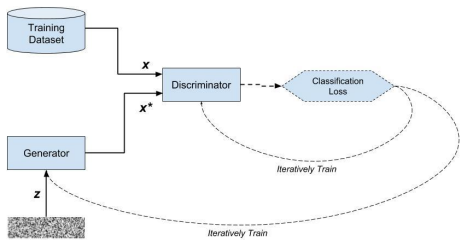

Figure 3. Generic GAN Training

It acts as the inout to our Generator network. This noise gets transformed into fake images with the help of the Generator.
Our dataset is either the real pizza images or the synthetic pizza images.
Discriminator and Generator are both Neural Networks that are trained together.
In GAN, a Discriminator competes with a Generator same as two players of zero-sum games.

3.3.1 Training Discriminator

Get real images from datasets and input them to the discriminator. Label the output as 1. Generate random noise z and input it to the generator to get $G(z)$, and then input $G(z)$ to the discriminator. Label the output as 0.

3.3.2 Training Generator

Generate random noise z and input it to the generator to get $G(z)$, and then input $G(z)$ to the discriminator. Label the output as 1.

3.4. Methodology

We used script files to train the GAN and persist the weights. Due to the rigorous disconnects and failures faced in using Jupyter Hub, we decide to move to ilabs.We executed python scripts in the ssh environment of ilabs and persisted the weights of generators and discriminators. However,we later realized that we have to submit completely executed colab notebooks.Hence, due to the shortage of time and carelessness from our end we could run lesser number of epochs to train our GAN in the Colab Notebook. We have shared both the colab notebooks and the python scripts used by us for the training of the GAN. I hope you understand.

4. Qualitatively Evaluation

4.1. Real Pizza Dataset

We ran 100 epochs in the initial setup and here are the images generated by the GAN at different epochs:

Figure 4. Output of GAN trained on Real Pizza dataset - Epoch 67

Figure 5. Output of GAN trained on Real Pizza dataset - Epoch 74

In the initial epochs the Generator learns the borders and structure of the pizza. In further epochs we can observe that the generator tries to learn the color of the pizza top (the sauces) and then toppings on the pizza.We can see that the generated pizza images are blurry in the toppings region.

Figure 6. Output of GAN trained on Real Pizza dataset - Epoch 87

Figure 7. Output of GAN trained on Real Pizza dataset - Epoch 89

4.2. Synthetic Pizza Dataset

We ran 50 epochs in the initial setup and here are the images generated by the GAN at different epochs:
In the initial epochs the Generator trained on the Syntehtic

Figure 8. Output of GAN trained on Synthetic Pizza dataset - Epoch 30

Pizza Dataset learns the borders and structure of the pizza. In further epochs we can observe that the generator tries to learn the sauce and cheese on the pizza.We can see that the generated synthetic pizza images are blurry as the number of epochs it is trained for is less.

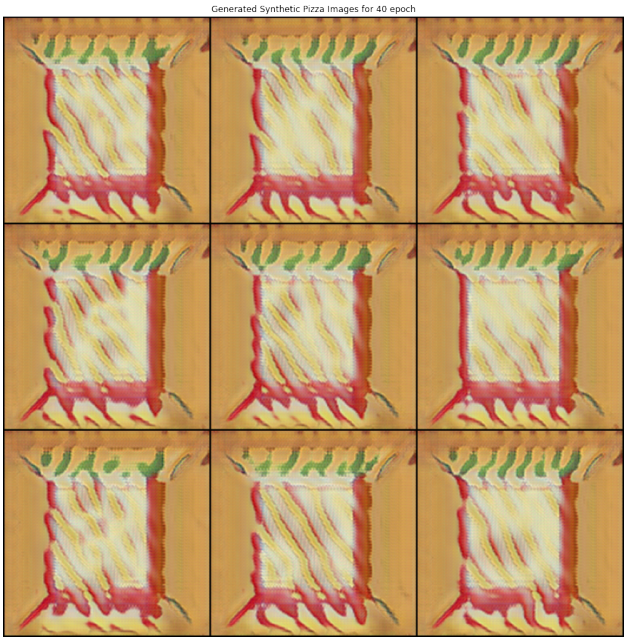

Figure 9. Output of GAN trained on Synthetic Pizza dataset - Epoch 40

4.3. Metrics

Real Pizza Dataset

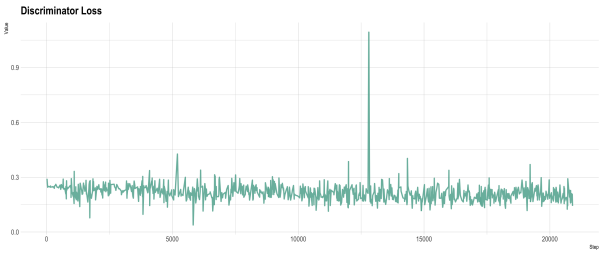

Figure 10. Discriminator Loss

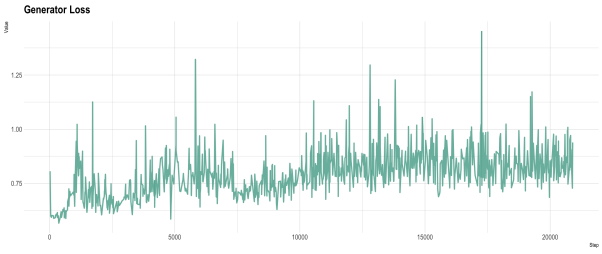

Figure 11. Generator Loss

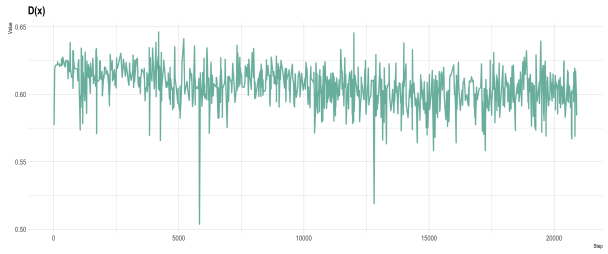

Figure 12. $D(x)$

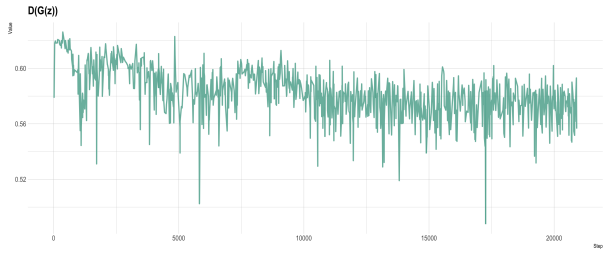

Figure 13. $D(G(z))$

Synthetic Pizza Dataset

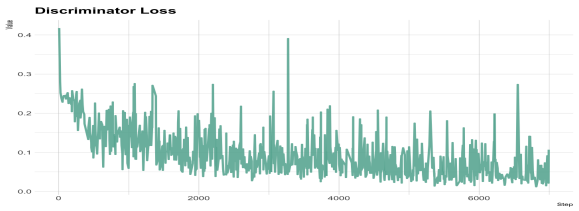

Figure 14. Discriminator Loss

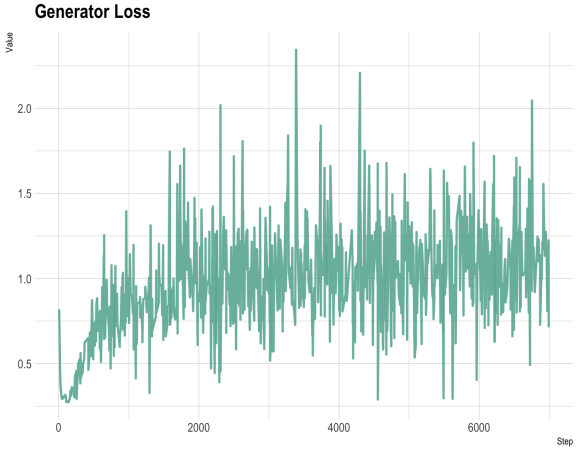

Figure 15. Generator Loss

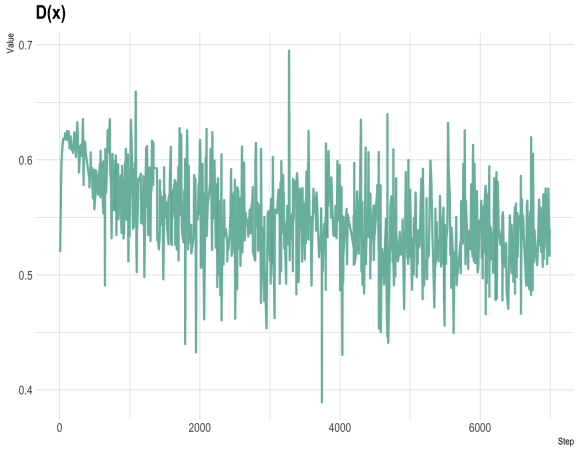

Figure 16. $D(x)$

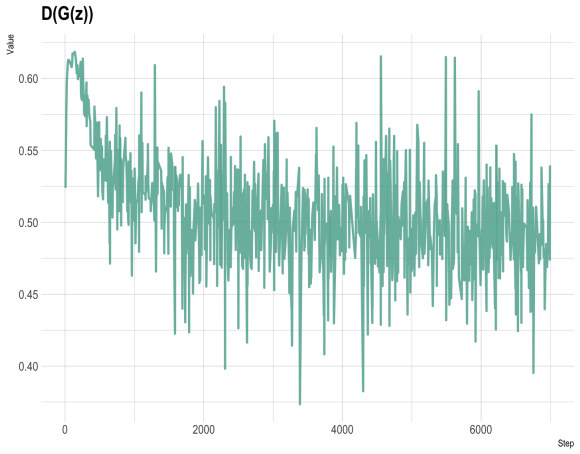

Figure 17. $D(G(z))$

For both real and synthetic pizza datasets:
We can observe that both the metrics D_x and $D(G(z))$ tend to 0.5. This means that after training for more epochs our discriminator is no longer able to discriminate between the real and fake images and hence our goal to train a generator that can fool the discriminator is realized.
Generator Loss increases as expected and the Discriminator Loss decreases as expected too.

5. Quantitative Evaluation

5.1. Introduction

The two most common GAN evaluation measures are Inception Score (IS) [5] and Fréchet Inception Distance (FID) [1]. In this section, we use FID and IS to evaluate

our GAN results quantitatively. Code Implementations see FID_IS.ipynb

5.2. Inception Score (IS)

IS computes the KL divergence between the conditional class distribution and the marginal class distribution over the generated data.[5]. IS does not capture intra-class diversity, is insensitive to the prior distribution over labels (hence is biased towards ImageNet dataset and Inception model4), and is very sensitive to model parameters and implementations.

5.2.1 IS for generated real pizza

Figure.18 shows the inception score of generated real pizza from epoch 1 to epoch 85.

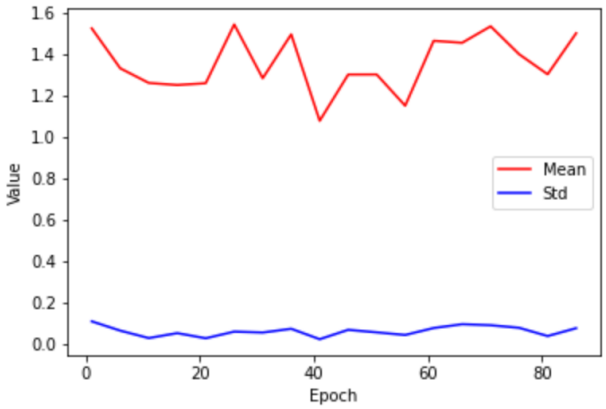

Figure 18. FID for real generated images

5.2.2 IS for generated synthetic pizza

Figure.19 shows the inception score of generated synthetic pizza from epoch 1 to epoch 85.

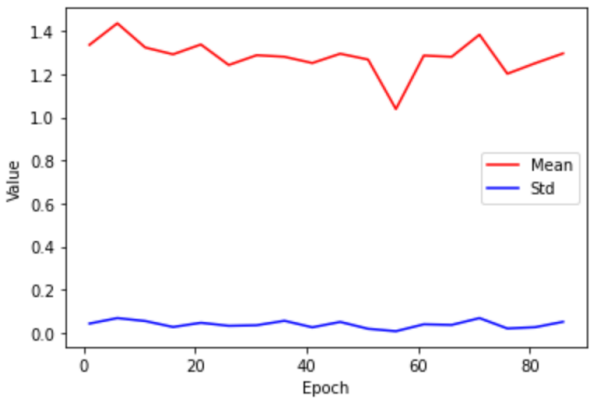

Figure 19. FID for synthetic generated images

5.3. Fréchet Inception Distance (FID)

Unlike the earlier inception score (IS)[5], which evaluates only the distribution of generated images, FID [1] calculates the Wasserstein-2 (a.k.a Fréchet) distance between multivariate Gaussians fitted to the embedding space of the Inception-v3 network of generated and real images[5]. And it can detect intra-class mode collapse.

5.3.1 FID for generated real pizza

Figure.20 shows the FID of generated real pizza from epoch 1 to epoch 85.

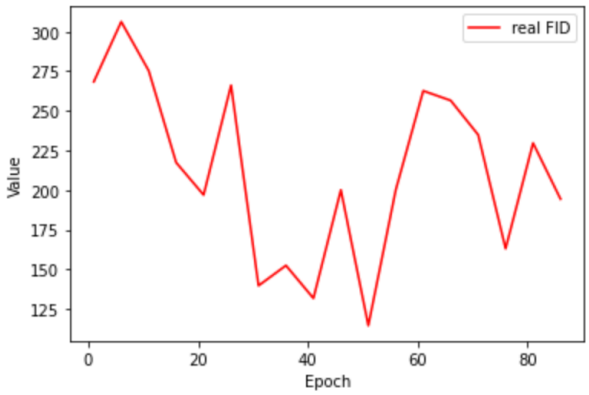

Figure 20. FID for real generated images

5.3.2 FID for generated synthetic pizza

Figure.21 shows the FID of generated synthetic pizza from epoch 1 to epoch 85.

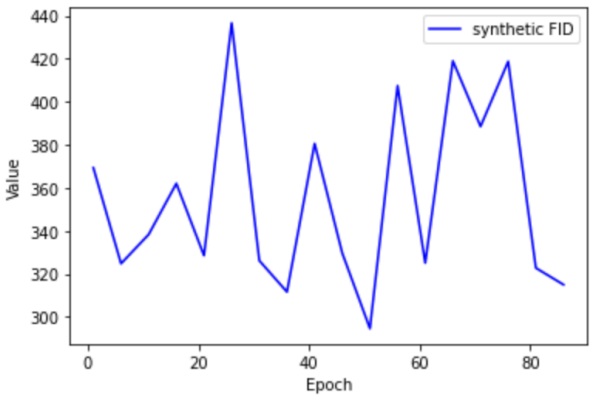

Figure 21. FID for synthetic generated images

6. Analysis

6.1. IS for generated real pizza

We can observe that the inception score [5] increases slightly with increase in the number of epochs. Increase in the IS indicates better performance of the model, which is inline with our generated images.

6.2. IS for generated synthetic pizza

Similar to the pizza images generated on the RealPizza Dataset, there is an increase in the IS [5] with increase in the number of epochs, showing improvement in the performance of the model. This again is consistent with our generated images.

6.3. FID for generated real pizza

For the pizza images generated on the RealPizza dataset, the FID score [1] has a value of 275 initially. With increase in the number of epochs, there is an overall decrease in the FID. This implies better performance of the model with increase in the number of epochs, which is consistent with our generated images. The FID score goes as low as 125. With increase in the number of epochs and decrease in batch size, we can further improve the performance of the model, thereby further decreasing the FID.

6.4. FID for generated synthetic pizza

For the pizza images generated on the SyntheticPizza dataset, the FID score [1] has an initial value of around 370. We can see that similar to the pizza images generated on the RealPizza dataset, there is an overall decrease in the FID score with increase in the number of epochs, indicating improvement in the performance of the model. This is inline with the quality of our generated images. The minimum value of the FID score obtained here is slightly below 300. This can be further reduced by increasing the number of epochs and decreasing the batch size.

7. Conclusion

In this project step we tackle the problem of Generating fake images using Generative Adversarial Networks. We trained two GANs, one for the real pizza images and one for the synthetic pizza images. We observed that the images generated by the real pizza GAN were very visually appealing and looked very real. We can achieve better performance by using a smaller batch size and training more epochs.
Further, we were able to generate good results with 85 epochs of training and if we were to train 100-150 epochs and augment more data either via blurring , rotating images ,adding jitters, we can achieve better results. Another future scope of work would be to fine tune the architecture to get better results.

References

[1] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017. 4, 5, 6

[2] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "image-to-image translation with conditional adversarial networks" in *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 1125–1134, 2011. 2

[3] J. Johnson, A. Alahi, and L. Fei-Fei. "perceptual losses for real-time style transfer and super-resolution" in *European conference on computer vision*. pages 694–711, 2016. 2

[4] D. P. Papadopoulos, Y. Tamaazousti, F. Ofli, I. Weber, and A. Torralba. How to make a pizza: Learning a compositional layer-based gan model, Jan 1970. 1

[5] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016. 4, 5, 6