Generative Adverserial Netwok for Image To Image Translation

Animesh Sharma
animesh.sharma@rutgers.edu
Rutgers University
Department of Computer Science

Harish Udhaya Kumar
harish.udhayakumar@rutgers.edu
Rutgers University
Department of Computer Science

Jaini Patel
jp1891@scarletmail.rutgers.edu
Rutgers University
Department of Computer Science

Karan Pardasani
karan.pardasani@rutgers.edu
Rutgers University
Department of Computer Science

***Abstract*—In this project, we are solving the image transformation problem, where an input image is transformed into an output image. We are training the Generative Adverserial Network to train an unlabelled data set that is capable of generating the photo-realistic images. The machine is taught to learn the characteristics of pizza from the dataset and then it generates similar designs using the characteristics that the model learns from the dataset. This report also shows qualitative results that indicates the performance of the Generative Adverserial Network on the synthetic and real pizza dataset.**

I. Introduction

Generative Modelling is an unsupervised learning task that discovers and learns the regularities or patterns in input data in such a way that the model can be used to generate new images that could have been drawn from the original dataset. For example, GANs can create images that look like flowers, even though the flower might not exist in real life.

A. Generator and Discriminator

There are two sub modules that are present in GAN namely, the Generator, which learns to produce the target output and the discriminator which learns to distinguish data from real and fake images. The training procedure involves training the generator fooling the discriminator, and the discriminator tries to not get fooled by the generator.
Given a set of data points X, formally, generators are defined as models that capture the probability of a point being in a dataset. The generative module models the distribution of data and generates different data samples that are present in the dataset. Whereas the Discriminator is the classifier that distinguishes whether the data from the generator and classifies whether the data is different from real or not.

B. Training

When we start training the GAN network, initially, the generator produces fake data and the discriminator learns to tell that the generated data is fake.
As training increases, the generator learns how to generate the output that fools the discriminator. In the end, if the generator is able to learn the distribution in the dataset, then the discriminator gets bad at telling the difference and the accuracy of the discriminator decreases.

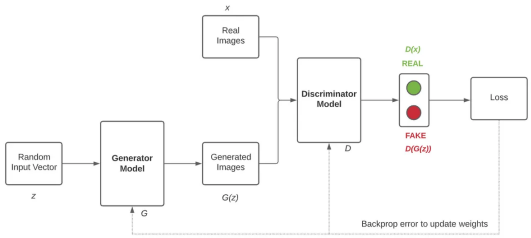

Fig. 1. Loss graph of GAN trained on Real Pizza dataset

II. Model Architecture

To learn the data distribution of the real and synthetic pizza dataset, we have trained a GAN model from scratch. The following section describes the architecture we have used for the training of the model.

A. Notations

The notation of the convolutional blocks used for the description of our architecture is as follows:

- **c7s1-k**
  This block will have 7x7 Convolution-InstanceNorm-ReLU layer with k filters and stride 1
- **dk**
  This block will have 3x3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2
- **Rk**
  Rk will have a residual block with 3x3 convolution layers with the same number of filters on both layers
- **uk**
  uk will have 3x3 fractional-strided-ConvolutionInstanceNorm-ReLU Layer with k filters and stride $\frac{1}{2}$

Fig. 2. Image of Real Pizza Dataset

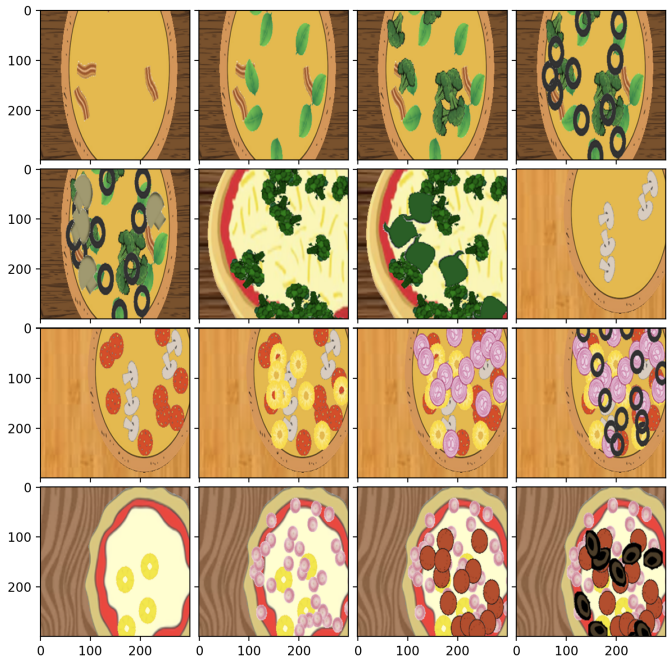

Fig. 3. Image of Synthetic Pizza Dataset

B. *Generator Architecture*

Using the above notation, the architecture for our GAN model can be described as follows:
c7s1-64, d128, d256, R256, R256, R256, R256, R256, R256, R256, R256, R256, u128, u64, c7s1-3

C. *Discriminator Architecture*

The architecture for our discriminator is adopted from [3] and is as follows:
C64-C128-C256-C512
In the above architecture, Ck denotes a 4x4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. Also, we apply convolution to the output of the last layer to produce a 1 dimensional output. In addition to this, we do not use InstanceNorm for the first C64 layer. Also for the leaky ReLU function, the value of the slope parameter is 0.2.

III. Dataset Used

For this project, our group has used the pizza image dataset[2]. The dataset contains two types of images: Real Pizza images and Synthetic Pizza images. There are 8348 images in real pizza dataset and 16430 images of synthetic pizza dataset. The sample images of pizza dataset are shown in Figure 2 and Figure 3.

IV. Preprocessing

After analysing the images from the dataset, we found that the dimension of all the images were not uniform and some images had dimensions larger than 256x256 and some images had dimension smaller than 256x256. So, we did the folowing preprocessing steps before feeding the images to the training data:

- **Resize**: We resized the images to 256x256.
- **CenterCrop Function**: We used the CenterCrop Function to make sure that the cropped images retain the information present in the center of the image.
- **Normalisation**: We normalised the images to ”center” the data. We are doing this so that each feature of the image will have similar range and the gradients won’t blow up during the training.

V. Training

We implemented and did the training of the model in PyTorch. We separately trained the model from scratch for the real pizza dataset and the synthetic pizza dataset.
We used General Adverserial Loss for the training of Generator and Discriminator. The criterion used for calculating the loss was Mean Squared Error. To improve the training, we also used Adam Optimizer for update of weights. The parameter that we used for Adam Optimizer is 0.27.

The workflow of the training for GANs is that, first the real image was given as an input to the discriminator and then, we calculate the loss of the output of the discriminator against real label.
Then, we send an image generated by the generator and calculate the loss of the output against the fake label. Then we take average of these losses and backpropagate it.
In the same way, we generate an image using the Generator using Random noise as input and put the output of the generator in the Discriminator. Then we calculate the loss of

the output that we get from Discriminator against the real label. Then, we again backpropagate the loss to update the weights of the generator.

VI. Experiments and Evaluation

A. *Experiments*

For the generator architecture, we have used ReLU activation after the last layer (c7s1-3). The training on this model did not produce good results because the images were black and we believe it is happening because ReLU has a very large range. After this, we went through the [1] research paper, where the author uses tanh activation layer to keep the results between 0 and 255.
For Discriminator we implemented the architecture given in the paper. And the Discriminator along with the Generator with tanh in the output layer gave good results.
For the generator, the input is a noisy 3x256x256 matrix using which the generator model is trained to design different pizzas. For the discriminator, the input dimension is 256x256x3and the output dimension is 1x30x30. For training the model we take the label of the real image as 1x30x30 where all the values are 1 and for fake images we take the labels as 1x30x30 where all the values are 0.

B. *Evaluation*

In this section, we are conducting qualitative evaluation of the GAN. The following are the two images, that we got after training the GAN on real and synthetic pizza dataset. Figure 4 and Figure 5 describes the qualitative evaluation of the GAN output

Fig. 4. **Output of Generator after Training on Synthetic Pizza Dataset**. From the above image, we can see that the model has learned the shape of the pizza. The model has also identified certain ingredients like cheese, peppers, tomatoes. The main issue that we observe from the output is that the ingredients like peppers have only been kept at the boundary. Our conclusion is that if the model had been trained more, then the pizza designs would have been more realistic.

Fig. 5. **Output of Generator after Training on Real Pizza Dataset**.From the above image, we can see that the model has learned the shape of the pizza. The model has also identified certain ingredients like cheese, pepperoni, olives. The model is able to produce realistic images with different designs of the pizza.

The following are the graphs of loss function for real and synthetic images:

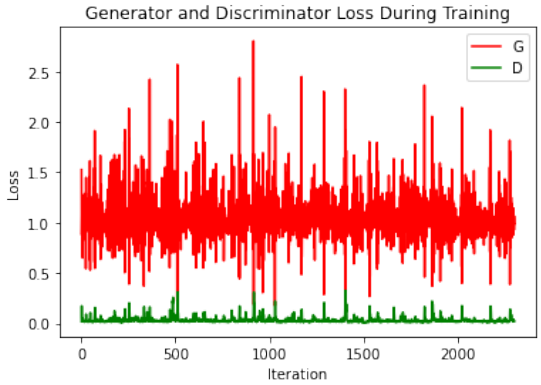

Fig. 6. **Change in Loss Function when trained on Synthetic Pizza images dataset**

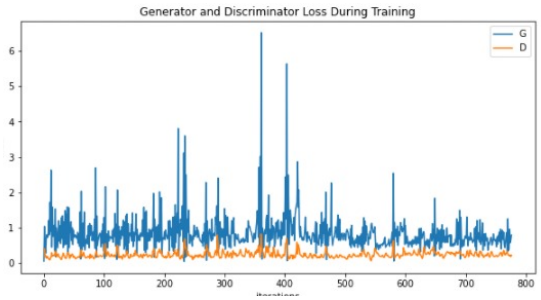

Fig. 7. **Change in Loss Function when trained on Real Pizza images dataset**

VII. References

1) Perceptual Losses for Real-Time Style Transfer and Super-Resolution Computer Vision-ECCV 2016, 2016, Volume 9906 ISBN : 978-3-319-46474-9 Justin Johnson, Alexandre Alahi, Li Fei-Fei

2) D. P. Papadopoulos, Y. Tamaazousti, F. Ofli, I. Weber, and A. Torralba, "How to make a pizza: Learning a compositional layer-based gan model," in proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8002-8011, 2019.
3) 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, pp. 1125-1134, 2017

VIII. Conclusion

In this report, we implemented a GAN architecture that generates new pizza images using the characteristics that are learned from real and synthetic pizza dataset. Though we have evaluated only in pizza dataset, we believe that this might be true for other types of food.