IMAGE GENERATION USING DCGAN

Abhishek Aryan
2K18/SE/007
Department of Software Engineering
abhishekaryan_2k18se007@dtu.ac.in

Vignesh Kashyap
2K18/SE/132
Department of Software Engineering
vigneshkashyap_2k18se132@dtu.ac.in

Abstract

Generative Adversarial Networks are one of the most involved and fast progressing areas in Deep learning. All models require good quality Data to be trained and tested against. Data is the most prized possession in 21st century. In this paper, we will explore on the synthetic generation of fruit images using Deep Convolutional Generative Adversarial Networks and outline some interesting quirks we observed.

1 Introduction

Data is being the most important resource of 21st century. If we were able to replicate similar data to generate synthetic data it would be the ideal solution. Requirement for extensive amount of data in order to improve and further provide new features with the same. With the application of Machine Learning in many fields, the requirement for data grows.

This is where Generative Adversarial Networks (Goodfellow, et al., 2014) comes to use, which is a very attractive method to achieve synthetic data generation.

Generative Adversarial Networks have proved to be quite efficient in generating data that mimic the data in the training set. GANs are a compelling alternative to maximum likelihood methods.

GANs have a reputation for being difficult to train, resulting in generators that create illogical outputs. There has been very little published research towards understanding and visualizing what GANs learn, as well as the intermediate representations of multi-layer GANs. Hence, we in this paper intend to explore and describe the process of GANs with respect to image generation.

Convolutional Networks are very good in identifying spatial correlation within the image. GAN architecture can be used for any type of data. But since we are primarily using GAN for Image Generation, we preferred the use of Deep Convolutional Generative Adversarial Networks.

Our aim in this paper is to implement DC GAN. We have used the Fruits 360 Dataset for training. Using the DCGAN we can generate synthetic images of fruits which would be similar to the training samples. The obtained images of the different fruits are then analyzed and we classify the performance measures of the different Fruit Images.

The following are the steps involved in this paper: Related Work, Methodology, Experimental Results, Conclusion

2 Related Work

In general computer vision research, as well as in the context of images, unsupervised representation learning is a well-studied problem. In the context of images, hierarchical clustering of image patches can be used to learn powerful image representations (Coates & Ng, 2012). Another prevalent method is to train auto-encoders that encode an image (convolutionally, stacked (Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010) separating what and where components of the code (Zhao, Mathieu, Goroshin, & LeCun, 2015), and ladder structures (Rasmus, Valpola, Honkala, Berglund, & Raiko, 2015)

We intend to use Deep Convolution Generative Adversarial Networks, an area where the research has not been as extensive. Image Generative Models can be classified into two: parametric and non-parametric.

Painting (Hays & Efros, 2007), Texture synthesis (Efros & Leung, 1999), have been achieved using non parametric models, wherein they patch different images from the training set.

The usage of parametric models to generate images is a well explored field (Portilla & Simoncelli, 2000). After that the research into generation of natural images came into picture recently. Another approach for the same was with the usage of Variational Sampler (Kingma & Welling, 2014), but the images obtained were quite blurred. Images generated from Generative Adversarial Networks were quite disturbed and noisy to understand clearly (Goodfellow, et al., 2014). Iterative Forward Diffusion Process is another method to generate images was discussed in (Sohl-Dickstein, Weiss, Maheswaranathan, & Ganguli, 2015). Due to the chaining of Multiple Models in Laplacian pyramid Extension (Denton, Chintala, Szlam, & Fergus, 2015) approach to GAN, the generated images looked unclear event though the resolution of images improved significantly.

DCGAN have been implemented (Dosovitskiy, Springenberg, Tatarchenko, & Brox, 2014) generate images of Chairs, Tables and Cars and (Radford, Metz, & Chintala, 2016) to generate faces and implemented on CIFAR 10 dataset.

In this research we have utilized Differentiable Augmentation for Data-Efficient GAN Training (Zhao, Liu, Lin, Zhu, & Han, 2020)

3 Methodology

3.1 Dataset

We have used the Fruits 360 dataset, which contains around 131 classes of fruits and vegetables. Each image is of the size 100 x 100 pixels.

Dataset was developed by recording a 20 second short video of the fruits which were placed on a slow motor shaft. There are a total of 90483 images. Training set contains 67692 images while the test set contains 22688 images. Different variations of the fruit were placed in different classes.

They also employed the use of a novel algorithm that extracts the food from the background as the variations in lighting condition produced unsatisfactory results.

From the dataset we have extracted the following fruits images:

- Apples (6,405 images)

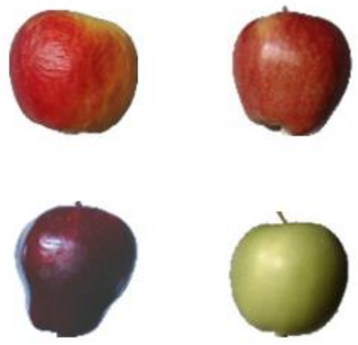

Figure 1: Apples

- Banana (1,430 images)

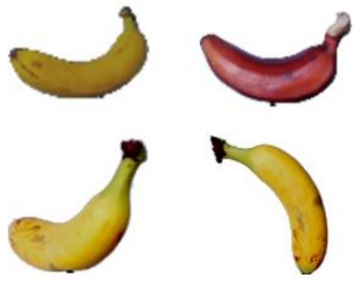

Figure 2: Bananas

- Grapes (3,419 images)

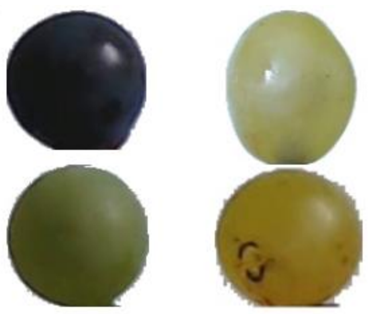

Figure 3: Grapes

- Tomato (5,103 images

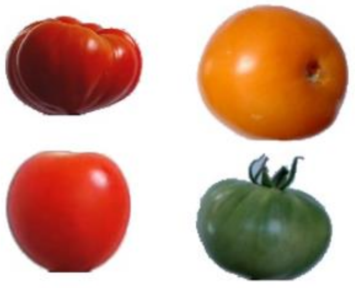

Figure 4: Tomatoes

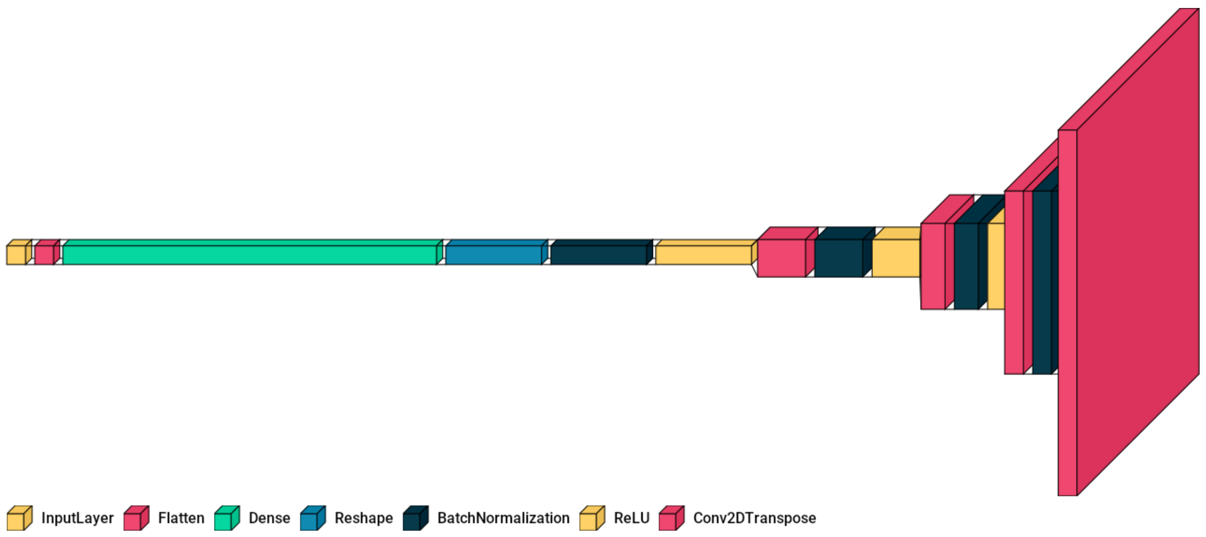

Figure 5: Architecture of the Generator

3.2 Architecture of the GAN

The proposed architecture of GAN (Goodfellow, et al., 2014) consists of two Networks which are trained simultaneously. At the end we end up freezing/ removing the “Discriminator” Network and just saving/ using the “Generator” part of the model. Due to this nature, GANs have been largely unstable to train, and often result in outputs that are random. Hence, we choose to use an architecture known as DC-GAN (Radford, Metz, & Chintala, 2016).
(Radford, Metz, & Chintala, 2016) proposed to use a set of constraints on the architectural topology of contemporary Convolutional GANs and hence become more stable for training. Hence their name, Deep Convolutional GANs (DCGANs).

The first is all convolutional net (Dosovitskiy, Springenberg, Tatarchenko, & Brox, 2014) which uses strided convolutions instead of deterministic spatial pooling functions like maxpooling to enable the network to learn its own spatial downsampling.

We also try to eliminate fully connected layers on top of convolutional features. This was previously utilized in state of art Image classification networks (Mordvintsev, Olah, & Tyka, 2015). The GAN's first layer, which takes a uniform noise distribution Z as input, may be considered entirely connected since it is simply a matrix multiplication, but the result is reshaped into a four-dimensional tensor and used as the start of the convolution stack. For the discriminator, the last convolution layer is flattened and then fed into a single sigmoid output.

We then use Batch Normalization (Ioffe & Szegedy, 2015) which stabilizes learning by normalizing the input to each unit to have zero mean and unit variance. This technique proved to be very helpful in curbing problems that arise due to poor initialization and helps the gradient in the model to slowly flow. As discussed in their paper, applying directly to all layers though, causes sample oscillation and made the model unstable. It was avoided by not applying batch normalization to the output and input layers of generator and discriminator models, respectively.

As discussed in their paper, the usage of ReLU activation (Nair & Hinton, 2010) is used for all except the output layer that used Tanh instead. Bounded activations allow the model to learn more quickly and expand to the whole domain’s color space. For discriminator training, leaky ReLU performed better instead of the maxout activation discussed by (Goodfellow, et al., 2014).

A major part of the architecture was as per the DCGAN guidelines provided by (Radford, Metz, & Chintala, 2016) with some extra layers of convolutions added and the input and output layers tweaked according to the dataset we had in use. We also tried to reduce the operational memory and expedite the process. During the model architecture process, we were able to remove ~3 convolutions without much harm to the generator,

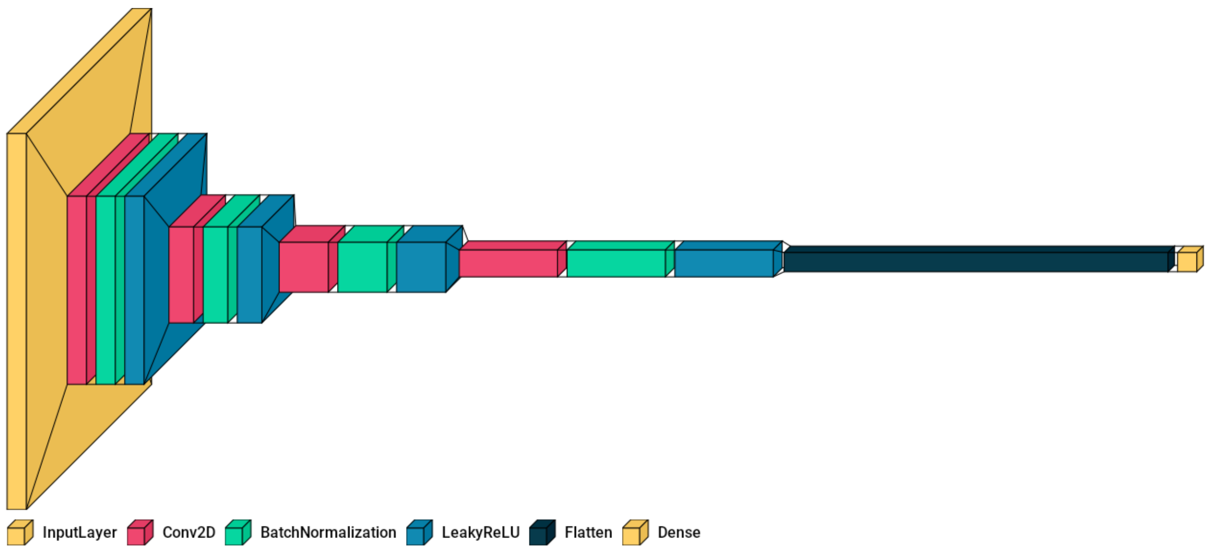

Figure 6: Architecture of the Discriminator

but as it was just generated over ~50 epochs, we are quite uncertain about its long-term effects.

To augment the training data in an effective manner, we used ideas proposed by (Zhao, Liu, Lin, Zhu, & Han, 2020) To produce the significant diversity of data required by the GANs. This was especially useful in the context of some fruits like bananas which had only 1,430 images available for training. Many large scale and successful models must be provided a huge number of training data. To name a few, FFGQ dataset used by StyleGAN contains 70,000 selective high-resolution images of human faces. Collecting this scale of data is very unrealistic and requires months or even years of considerable human effort to be able to be procured. In various cases it might not be even possible to have that number of examples, like in the case of a landmark, a rare species or a localized object. Reducing the amount of available data causes the quality of the generator to decrease drastically. A study done by (Zhao, Liu, Lin, Zhu, & Han, 2020) showed that using a small subset of the CIFAR dataset (around 20%) caused the training accuracy of the discriminator to quickly saturate to nearly 100% but the validation accuracy then kept decreasing. This leads us to believe that in small datasets the discriminator simply memorizes all the images in the data. This severe overfitting adversely affects the training of the generated and causes it to only learn a few images due to the positive feedback loop.

Still even this technique cannot really help to get more diversity in the generated image sets as the information required for that simply does not exist in the dataset.

3.3 Evaluation Metrics

Due to the convergence of GAN training being very unpredictable and not being able to use a measurable metric like MLE. There is usually not a good metric for objectively evaluating the Performance of GANs. We use the techniques Inception Score discussed by (Salimans, et al., 2016) and Fréchet Inception Distance (Heusel, Ramsauer, Unterthiner, Nessler, & Hochreiter, 2018) to evaluate the performance of the produced generators. The first metric uses the KL divergence to combine two criteria, the marginal and the conditional probability of the images from the generated set. Which reflect the diversity and the quality of the images, respectively. A higher value is preferred as it represents a high KL divergence between the two compared distributions.

$$\mathrm{IS}(G) = \exp\left(E_{x\sim p_g} D_{KL}(p(y\,|x)\,||\,p(y))\right), \quad (1)$$

$$\mathrm{p(y|x)}, \quad (2)$$

$$\mathrm{p(y)} = \int_{x} \mathrm{p(y|x)p_g(x)}, \quad (3)$$

Where y is the label for an image x,

Here gives the Inception Score of a Generator.

Fréchet Inception Distance uses an Inception Network as well but extracts the features from an intermedia layer instead. Lower FID is better.

$$\mathrm{FID} = \left|\left|\mu_r - \mu_g\right|\right|^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{1/2}\right), \quad (4)$$

Where $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$ are the 2048-dimensional activations of the Inception-v3 pool3 layer for real and generated samples, respectively.

The combination of these metrics allows us a good peek into the real-world performance of our GAN. We discuss some shortcomings of these metrics in our Conclusion which we deduced due to some baffling results we encountered from our Network.

4 Experimental Results

We trained the model on 4 different fruit distributions. We tried to capture some different scales and shapes by picking these examples.

The embedded GIFs show the training progress of the generator model and is made by reusing the same seed to generate an image after every training epoch. The images are not temporally stable over the duration of the full training cycle and we can see some significant changes in that are only present in that epoch, due to the low number of epochs we also see some significant background noise and a noise texture overlay as well. We hope to rectify it by training it for much more epochs.

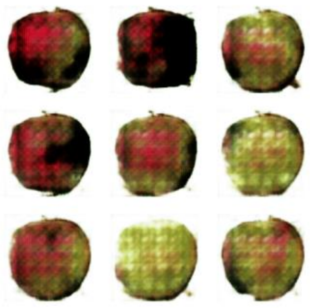

Figure 7: Performance of Apple GAN over the Training Cycle

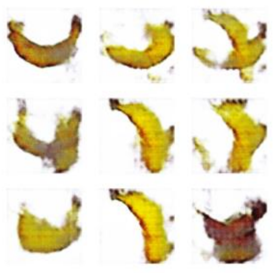

Figure 9: Performance of Banana GAN over the Training Cycle

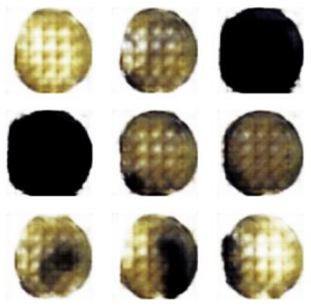

Figure 8: Performance of Grapes GAN over the Training Cycle

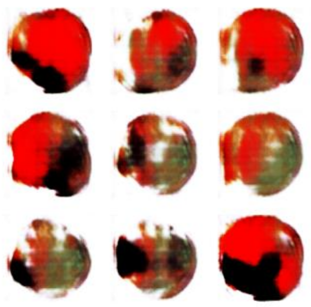

Figure 10: Performance of Tomato GAN over the Training Cycle

Fruit	Inception Score	Fréchet Inception Distance
Apple	(2.475, 0.112)	129
Banana	(2.556, 0.291)	270
Grape	(3.001, 0.078)	250
Tomato	(1.382, 0.036)	236

The results we see from the mentioned (Table 1:

Table 1: Fréchet Inception Distance and Inception Score of the 4 Fruit GANs

Fréchet Inception Distance and Inception Score of the 4 Fruit GANs) are very peculiar indeed. While human trails we did expressed that the Bananas were the most realistic looking and diverse, the Scores we calculated show the inverse, with bananas having the highest FID score and the Apples having the lowest. This also relates to the findings in (Barratt & Sharma, 2018). We feel that this is because of the sparse number of samples that we could use for training the inception network and hence deriving the scores from it. In a contemporary paper upwards of 10,000 samples are used for both the FID and IS scores.

5 Conclusion

While using the DiffAugment (Zhao, Liu, Lin, Zhu, & Han, 2020) policy did allow us to now have convergence failures, it also caused our model to initially mode collapse in the first 20 epochs, as it kept providing pics with similar composition. This can be really observed in cases like the model that was trained on grapes which as it only generates very similar looking grapes and has almost the same shape.

Even though we used TPUs for the training, some fruits and had various run to run variations in the training time which made the whole process hard to automate, we hope to get more powerful hardware in the future to continue optimization and integrate various other developments to rectify the shortcomings present in our method.

References

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN.

Barratt, S., & Sharma, R. (2018). A Note on the Inception Score.

Coates, A., & Ng, A. Y. (2012). Learning Feature Representations with K-Means. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.). Berlin, Heidelberg: Springer Berlin Heidelberg.

Denton, E., Chintala, S., Szlam, A., & Fergus, R. (2015). Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *arXiv: Computer Vision and Pattern Recognition*. Retrieved 5 28, 2021, from https://arxiv.org/abs/1506.05751

Dosovitskiy, A., Springenberg, J. T., Tatarchenko, M., & Brox, T. (2014). Learning to Generate Chairs, Tables and Cars with Convolutional Networks. *arXiv: Computer Vision and Pattern Recognition*. Retrieved 5 28, 2021, from https://arxiv.org/abs/1411.5928

Efros, A. A., & Leung, T. K. (1999, September). Texture Synthesis by Non-parametric Sampling. *IEEE International Conference on Computer Vision*, (pp. 1033-1038). Corfu, Greece.

Goodfellow, I. (2017). NIPS 2016 Tutorial: Generative Adversarial Networks.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative Adversarial Networks.

Hays, J., & Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics, 26*(3), 4. Retrieved 5 28, 2021, from https://dl.acm.org/ft_gateway.cfm?id=1276382&ftid=426808&dwn=1

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2018). GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium.

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). *Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations*. Retrieved 5 28, 2021, from http://web.eecs.umich.edu/~honglak/icml09-convolutionaldeepbeliefnetworks.pdf

Mordvintsev, A., Olah, C., & Tyka, M. (2015). *Inceptionism: Going Deeper into Neural Networks*. Retrieved 5 28, 2021, from https://ai.google/research/pubs/pub45507

Mureșan, H., & Oltean, M. (2018, 06). Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica, 10*, pp. 26-42.

Nair, V., & Hinton, G. E. (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines*. Retrieved 5 28, 2021, from http://cs.toronto.edu/~fritz/absps/reluicml.pdf

Portilla, J., & Simoncelli, E. P. (2000, oct). A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. *Int. J. Comput. Vision, 40*(1), pp. 49-70.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., & Raiko, T. (2015). *Semi-supervised learning with Ladder networks*. Retrieved 5 28, 2021, from http://papers.nips.cc/paper/5947-semi-supervised-learning-with-ladder-ne

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved Techniques for Training GANs.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010, dec). Stacked Denoising

Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res., 11*, pp. 3371-3408.

Welling, D. P. (2014). Auto-Encoding Variational Bayes. arXiv. doi:1312.6114

Zhao, J. J., Mathieu, M., Goroshin, R., & LeCun, Y. (2015). Stacked What-Where Auto-encoders. *arXiv: Machine Learning*. Retrieved 5 28, 2021, from https://arxiv.org/abs/1506.02351

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., & Han, S. (2020). Differentiable Augmentation for Data-Efficient GAN Training.