DCGAN for Cartoon Dataset

Arshdeep Singh
IIT Delhi
cs5160625@iitd.ac.in

Aditya Jain
IIT Delhi
cs1160335@iitd.ac.in

1. Introduction

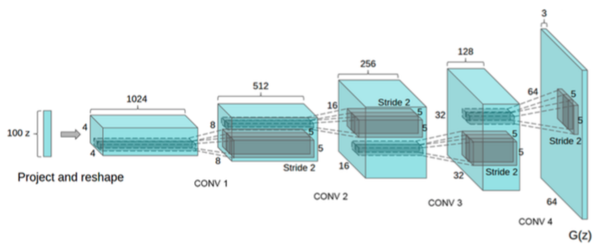
In this work, we train a generative adversarial network (GAN), to generate new Cartoons after showing it pictures of cartoons from a cartoons dataset. We use Cartoon set dataset from Google [1]

2. Related Work

GANs are a framework through which we can teach a Deep learning model to capture training data's distribution. They were initially introduces in a paper by Goodfellow in 2014. They are made of two networks generator and discriminator. Generator is trained in such a way to produce fake images that follow the training data distribution. And the job of the discriminator is to distinguish between real and fake images. During training, generator is constantly trying to fool the discriminator, while discriminator is trying to become better at detecting fake images. This game between generator and discriminator reaches equilibrium when the generator is generating perfect fakes that look like coming from the training data, and

3. Proposed Model

In this work we train a fully convolutional generative adversarial network known as DCGAN for the cartoon dataset . The generator architecture of DCGAN is as follows : The



network uses a all convolutional network with strided convolutions , this allows the network to learn to its own spatial downsampling. This technique is used in both generator and discriminator.

Batch normalization is used to stabilize learning by normalizing input to zero mean and unit variance. This helped in preventing the generator from mode collapsing which is a common failure in GANs. However applying batchnorm to all layers results in sample oscillation so batchnorm was not applied at generator output layer and discriminator input layer as told by the paper.
Next , ReLu is used as activation in generator except at the output layer which used tanh fucniton. Using bounded activation allowed the model to learn faster and cover the color space of the distribution. Also Leaky ReLu is used as activation in the discriminator with sigmoid at the end.

4. Experimental Details

The model was trained with a batch size of 128 . The image size used was 64 , and training images were resized to match this size. The size of the latent vector (generator input) used was 100. The model was trained for 50 epochs with learning rate of 0.0002 and beta1 (adam optimizer) as 0.5 . All weights were initialized from a zero-centered Normal distribution with standard deviation 0.02. In the Leaky ReLU, the slope of the leak was set to 0.2 in the model.

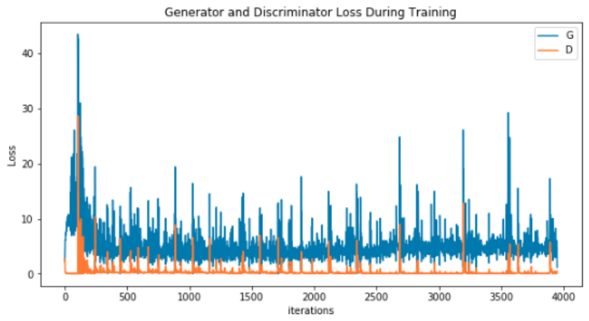
5. Training

With all the parts set up, we begin training GAN, which is somewhat tricky as incorrect hyper parameter settings can lead to mode collapse. We followed the training procedure from Goodfellow's paper, while following best practices from ganhacks [2]

1. Training the discriminator with the goal of maximizing the probability of correctly classifying a given input as real or fake. In terms of Goodfellow's paper we want to "update the discriminator by ascending it's stochastic gradient". We want to maximize $\log(D(x)) + \log(1 - D(G(z)))$. We follow the seperate mini batch advice from ganhacks, we calculate this in two steps. First, constructing a batch of eal samples from the training set, making a forward pass through D, and hence calculating the loss $\log(D(x))$ then calculate the gradients in a backward pass. Secondly, we

took a batch of fake samples with the current generator, forward pass this batch through D, calculate the loss $(\log(1-D(G(z))))$, and accumulate the gradients with a backward pass. Now, with the gradients accumulated from both the all-real and all-fake batches, we call a step of the Discriminators optimizer.

2. To train the Generator we want to minimize $\log(1-D(G(z)))$ to generate better fakes. This was shown by Goodfellow to not provide better sufficient gradients, especially early in the learning process. As a fix, we maximise $\log(D(G(x)))$. We do this by classifying the Generator ouput from Part1 with the discriminator computing G's loss using real labels as Ground truth. Computing G's gradients in a backward pass and finally updating G's parameters with an optimizer step.



6. Results

The following images shows the real images vs the fake images generated by our model in a 8 by 8 fashion: Some amazing results were found as follows which shows that the model has learned some hidden underlying representation of the data:

Figure 1. adding noises of two images produce combined attributes

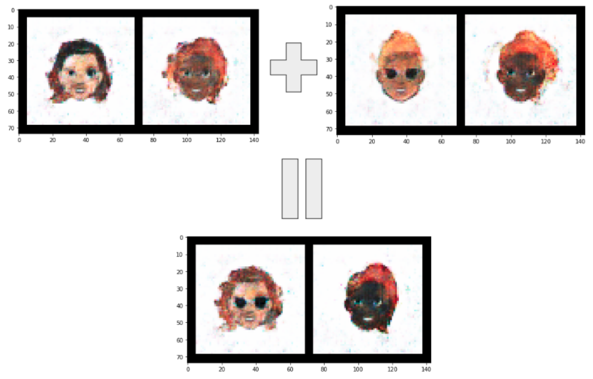


Figure 1 shows how adding the noise of two inputs produces the results which contains both of their attributes. As the left combined image containes the face of the left and glasses of the right cartoon while the right combined image increases the facial color attribute and maintains the hair style of the right subimage. Similarly it was found that these mathematical operations can be extended to more than one operation. Figure 2 shows this as the resulting image contains the hair style of the first image, glasses of the second image and complexion of the second image. These results are very intuitive and tells us that the model has learned the underlying distribution.

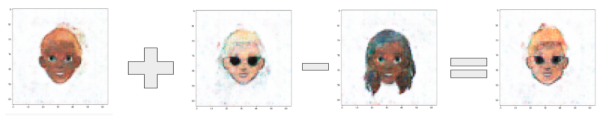


Figure 2. Combining different mathematical operations to produce their aggregated result

7. Conclusion

In this work we CNNs can be useful in the domain of unsupervised learning by their application in GANs. We implemented novel architecture model known as DCGAN for the google cartoon images dataset . Further more results obtained shows how the deep learning model learns the hidden underlying distribution of the data and how it produces human intuitive results when mathematical operations are done on the latent space using which they are created. We further hope to extend the model for more high resolution images in future.

References

[1] Cartoon set. 1

[2] M. A. M. M. Soumith Chintala, Emily Denton. How to train a gan? tips and tricks to make gans work. 1

Real Images

Fake Images