<u>Extremely Simplified Explanation of GAN :</u>

<u>By –</u> Ayush Agarwal, Electronics engineering , IIT BHU Varanasi

<u>Motivation :</u>

This paper intends to explain the concept of Generative Adversarial Networks (GAN's) in a simplified way to the audience .

<u>Background :</u>

GAN were introduced recently in 2014 in a research paper by Ian Goodfellow , who is now called as the father of GAN or GANfather . This concept was more developed in a research paper in 2016 . Since then , GAN s have been studied and have been made use of , and their newer versions have also come up .

<u>Explanation of GAN :</u>

Our aim is to add creativity to artificial intelligence , to be able to draw pictures using neural networks . Since Convolutional Neural Networks are known to extract features out of an image , we can expect that applying this operation in the reverse order should generate an image . This process is called deconvolution . Now we need to adjust the weights of this deconvolution network so that it outputs a proper image which makes sense to us . How will we do this ?

We do this using GAN . In GAN , we have 2 neural networks , generator and discriminator . The Generator is responsible for generating the images throught deconvolution , and the Discriminator is used to detect whether the image is real or fake .

Now suppose we want our Generator to draw human faces (just an example object , there are some problems associated with Vanilla GAN that dosent allow drawing shapes with that much precision , and SAGAN and cSAGAN are used or other modifications are used , let's not delve into it .) So we want to make human faces .

First we get ourselves a dataset of human faces , and we train our discriminator to identify whether the image is a human face or not . Now our discriminator knows to identify what human faces look like .

Next , we run our generator (yes we haven't trained it yet) , it produces some noise .

We then take these noisy images from the generator , feed them into the discriminator , and train it by telling it that these images are not human faces . Now our discriminator can clearly differentiate between human faces and noise from the generator .

Now we attach the generator and the discriminator together as a neural network . And we set the input to a seed input (most people take it as a vector of 1s , usually 5 , 10 , 100 in length) , and we set the output to 1 (1 meaning yes human faces are detected) , and discriminator is frozen layer (meaning we are not training it , it remains untrained and unchanged) . Now we start the training , through propagation and back propagation , the weights of the generator network get updated in such a way that it gets closer to outputting a real human face . (Since the generator should produce an image that should qualify as human face and not noise)

We usually say that generator is trying to fool the discriminator . We say that because the discriminator is trained to detect real images (human faces from the dataset) and fake images (from the generator) , while the generator is constantly training to produce images to fool the discriminator and make it believe that it has given it a real human face image . Hence in this “fooling” process , the generator is learning to draw a human face .

Now after this step , after the generator is trained (beware , not fully trained yet) , we separate the generator discriminator pair and make the generator output some images . Now , the discriminator is again trained by showing it the real dataset labelled as 1 (real human face) and the output produced by generator as 0 (not a real image , noise from generator , not human face , fake) . So now Discriminator is strengthened .

Now the generator and discriminator are again joined together , again seed is given as input , 1 is given as output , and again training is done with discriminator frozen , and again weights of generator are updated through backpropagation and training is done till the time it can fool the updated discriminator . So now once training is done , the generator is now even better at making realistic human faces .

This cycle is repeated a number of times to train the discriminator and the generator . Finally , to test whether our GAN is working good or not , we look at the output of the generator . And this output will resemble the images we wanted to draw in the first place , that is human face .

This is a very simple explanation for GAN invented by me which even someone like me with minimal experience in the field of ML can understand . The main focus of this paper has been DCGAN , which stands for Deep Convolution Generative Adversarial Networks , and basically , these are GANs which use Convolution Neural Networks and Deep learning techniques to make the GAN .

Multi Class GAN :

Now coming to the multiple image drawing network . Say we want a generator network which can make multiple images based on the input seed given to it . Say now we trained our discriminator on a dataset of multiple classes (like cifar10) . Now our discriminator is a classifier , say we have 3 classes so the output of this classifier can be 000(None), 100 (class 1), 010 (class 2), 001 (class 3) . Now , when we are training the generator , we train it in such a way that when input seed 100 is given , we also give it output as 100 and train it . This way it will produce an image of first class . Similarly training for other seed inputs and outputs in this combined network (with discriminator frozen) , we can make the Generator draw images of different classes for different inputs 100 010 001 given to it . Hence now our GAN is able to draw multiple images from different seeds . But this will require denser network .

EndNote :

Now that the explanation part is done , one of the things about GAN is that it is it is computationally very expensive (requires GPUs to train) and also that it’s training time is very long (needs about 50000 epochs to train properly , which can even end up taking days altogether) . Hence its modifications are made and used . It is also the reason why I couldn’t run the GAN on my Kaggle notebook , simply because I couldn’t keep running the notebook till days .

Another problem that comes is that while it can easily learn textures of surface , it tends to have some difficulty learning the overall shape of the body dur to the local nature of the CNN.

References :

Mnist gan code :

DCGAN/dcgan.py at master · developershutt/DCGAN (github.com)

And its youtube tutorial :

DCGAN || GAN || Generative Adversarial Networks || Developers Hutt - YouTube

Dcgan for cifar10 dataset :

4thgen/DCGAN-CIFAR10: A implementation of DCGAN (Deep Convolutional Generative Adversarial Networks) for CIFAR10 image (github.com)

Another eg for gan with cifar 10 well explained :

GANs — Conditional GANs with CIFAR10 (Part 9) | by fernanda rodríguez | Medium

Tensorflow tutorial for dcgan : (didn't understand much)

Deep Convolutional Generative Adversarial Network | TensorFlow Core

Cifar10 original dataset from tensorflow :

cifar10 | TensorFlow Datasets

and from the original site :

CIFAR-10 and CIFAR-100 datasets (toronto.edu)

The original 2014 research paper on gan : (I guess its original)

1411.1784.pdf (arxiv.org)

And the 2016 original research paper on gan :

[1701.00160] NIPS 2016 Tutorial: Generative Adversarial Networks (arxiv.org)

Tensorflow code for deconvolution layer :

Conv2DTranspose layer (keras.io)

proof that dcgan takes huge time to train

DCGAN, cGAN and SAGAN & the CIFAR-10 dataset | by Shruti Bendale | Analytics Vidhya | Medium

Sagan

Not just another GAN paper — SAGAN | by Divyansh Jha | Towards Data Science