As per UGC guidelines an electronic bar code is provided to seure your paper

International Journal for Modern Trends in Science and Technology, 7(01): 114-120, 2021
Copyright © 2021 International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST070127
Available online at: http://www.ijmtst.com/vol7issue01.html

Human Face Generation using Deep Convolution Generative Adversarial Network

Chaudhary Sarimurrab[1] | Ankita Kesari[2] | Naman[3] | Sudha Narang[4]

[1]B.Tech Scholar, Department of CSE, Maharaja Agrasen Institute of Technology, Delhi, India
[2]B.Tech Scholar, Department of CSE, Maharaja Agrasen Institute of Technology, Delhi, India
[3]B.Tech Scholar, Department of CSE, Maharaja Agrasen Institute of Technology, Delhi, India
[4]Assistant Professor, Department of CSE, Maharaja Agrasen Institute of Technology, Delhi,India

To Cite this Article
Chaudhary Sarimurrab, Ankita Kesari, Naman and Sudha Narang, "Human Face Generation using Deep Convolution Generative Adversarial Network", *International Journal for Modern Trends in Science and Technology*, Vol. 07, Issue 01, January 2021, pp.- 114-120.

Article Info
Received on 18-December-2020, Revised on 02-January-2021, Accepted on 08-January-2021, Published on 17-January-2021.

ABSTRACT

The Generative Models have gained considerable attention in the field of unsupervised learning via a new and practical framework called Generative Adversarial Networks (GAN) due to its outstanding data generation capability. Many models of GAN have proposed, and several practical applications emerged in various domains of computer vision and machine learning. Despite GAN's excellent success, there are still obstacles to stable training. In this model, we aim to generate human faces through un-labelled data via the help of Deep Convolutional Generative Adversarial Networks. The applications for generating faces are vast in the field of image processing, entertainment, and other such industries. Our resulting model is successfully able to generate human faces from the given un-labelled data and random noise.

KEYWORDS: *Generative Models, Face Generations, Deep Convolutional Generative Adversarial Network*

I. Introduction

Generative Networks have collected a high quantity of attention within the field of learning that's unattended. GaN is that the most complex method in generative learning. This can be achieved through Generative Adversarial Networks. On paper, GAN takes a supervised learning approach to try and do unattended learning by generating pretend or artificial trying information. Generative Adversarial Networks gives a welcome to learn deep convolution while not extremely bounded information. They come through this deed associated opposing methods between 2 pairs of networks who fight with each other. To tackle, there are many some generative models that are planned. one amongst them is DC-GAN or Deep Convolution Gan. Deep Convolutional Gan may be a known form of GANs that uses convolutional layers that are absolutely connected. DCGAN is that the form of GAN model that's getting used here. The first goal of our project is to get faces with latent vectors or random noise. The essence of GAN is summated as coaching of 2 networks at the same time known as the generator network denoted by G and also the individual network indicated by D. D is truly a individual that follow its own methodology to classify original pictures as real. In qualification, G could be a generator and creates pictures and attempts to deceive the individual by generating somewhat real information. These 2 networks section themselves, and eventually, G produces realistic information, and D gets higher to predict the pretend ones The representations which will be learned by GANs additionally be utilized in a scope of uses, along with picture amalgamation, phonetics picture piece of writing, vogue transfer, image super-resolution, and classification. much, GAN has introduced several applications like hand-written font generation, image mixing, image in-painting, face aging, text synthesis, human create synthesis, script applications, image manipulation applications, visual salience prediction, object detection, 3D image synthesis, medical application, facial makeup transfer, facial landmark d1 detection, image super-resolution, texture synthesis, sketch synthesis, image-to-image translation, face frontal read generation, language and speech synthesis, music generation, video applications in laptop vision and graphics communities.

II. LITERATURE SURVEY

Generative models (GM) are a quickly propelling examination region of image vision problems. Generative models are the traditional models for unaided realizing where given preparing information ~ p-data(x) from an obscure information producing appropriation creates new examples information ~ p-model(x) from a similar dissemination. The ultimate objective of any GM is to draw comparable information tests (p-model(x) from the inclined genuine information conveyance p-data(x).

Generative Adversarial Networks (GAN), a robust network used for unsupervised machine learning to build a min-max game between two-player, i.e., setting up both the player (networks) with their different objectives. One player is known as the generator network(G), and the other is known as the discriminator network (D). The first player (G) attempts to trick the second player (D) by creating exceptionally common-looking true pictures from arbitrary idle vector z, and the second player (D) improves in-recognizing genuine and produced information. Both the organizations attempt to streamline themselves in the most ideal manner to achieve the individual goals on the grounds that both have their goal capacities, i.e., D needs is to expand its cost worth, and G needs to limit its cost esteem is given as given below:

Another class of convolutional neural organizations (CNN) [16] called Deep Convolutional GAN (DCGAN). DCGAN was the principal structure that rehearsed de-convolutional neural organizations (de-CNN) foundational layout that altogether balances out GAN preparing. These systems comprise of two organizations; one organization fills in as a CNN called the generator, and the other organization functions as a de-CNN called discriminator. A recently proposed class of design limitations remembered for CNN engineering is:

— Remove all degrees of pooling layers with step convolutions.

— Both G and D should utilize Batch Normalization (BN) [192].

— Use ReLU and Leaky-ReLU in the generator and the discriminator organizations, separately.. We will discuss pdftotext, tesseract and tesseract4.

III. METHODOLOGY

The project started by breaking down into a series of sub-parts from data loading using data loader to defining and training the adversarial networks.

In this project, we have used the Celeba dataset in which there are approx 30,000 celebrities face images. We used Pytorch, PyTorch is an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing,

In the first step, we loaded the data into the train loader. To load the data we first defined our transformation for all images. In transformation, we defined image size, which we kept 32*32. So that all images have the same size to work on. The second transformation definition is a center crop so that we have exact feature values. Transforms are common image transformations. We chained together these transforms using Pytorch Compose. Then we have applied these transform to the data set using the image folder function which takes two arguments, one is data and the other is transform definition.

At last, during the train loader, we divided the data set into batches of 20. So that we can train faster and also can look through the training process.

Now next step we took is defining the architecture of both the model discriminator and generator.

We are using the DC GAN technique which is based on CNN architecture. CNN's are used for image classification and recognition because of their high accuracy. It was proposed by computer scientist Yann LeCun in the late 90s when he was inspired by the human visual perception of recognizing things. The CNN follows a hierarchical model that works on building a network, like a funnel, and finally gives out a fully-connected layer where all the neurons are connected to each other and the output is processed.In the discriminator model, we have defined our first convolution layer in which input to layer is an image of 3 channels (RGB channel). And after that 32 is the dimension of the convolution layer. In other words, we are applying 32 kernels to an input image or convolving to an input image. The filter size or convolution kernel size is 4*4 which going to divide the x-axis and y-axis of the image by a factor of 4. Then we have used a stride of 2*2. Which will move with 2 steps during the convolution process. And also we set bias = False otherwise will deviate from zero.

Conv2d(3, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)

The next layer is also a convolution layer in which input channels are 32 and output channels are 64. And also filter size or convolution kernel size is 4*4 which going to divide the x-axis and y-axis of the image by a factor of 4. Then we have used a stride of 2*2. Which will move with 2 steps during the convolution process same as before.

Conv2d(32, 64, (4, 4) – size of filter, stride=(2, 2), padding=(1, 1), bias set to Fasle)

Then we used the batch normalization, Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. It does this scaling the output of the layer, specifically by standardizing the activations of each input variable per mini-batch, such as the activations of a node from the previous layer. Recall that standardization refers to rescaling data to have a mean of zero and a standard deviation of one, e.g. a standard Gaussian.

The next layer is also a convolution layer in which input channels are 64 and output channels are 128. And also filter size or convolution kernel size is 4*4 which going to divide the x-axis and y-axis of the image by a factor of 4. Then we have used a stride of 2*2. Which will move with 2 steps during the convolution process same as before. Then again we used the batch normalization layer . BatchNorm impacts network training in a fundamental way: it makes the landscape of the corresponding optimization problem be significantly more smooth. This ensures, in particular, that the gradients are more predictive and thus allow for use of larger range of learning rates and faster network convergence.

Then after we defined the one fully connected layer which takes the input size of 2048 and gives the output of 1 feature. Then we applied sigmoid function to output of full connected layer. The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice. The function is differentiable. And we also used dropout with 0.5 probability.

Dropout is where arbitrarily chose neurons are overlooked during preparation. They are "excited" haphazardly. This implies that their commitment to the actuation of downstream neurons is transiently eliminated on the forward pass and any weight refreshes are not applied to the neuron on the retrogressive pass.

As a neural organization learns, neuron loads subside into their setting inside the organization. Loads of neurons are tuned for explicit highlights giving some specialization. Neighboring neurons become to depend on this specialization, which whenever taken excessively far can bring about a delicate model excessively specific to the preparation information. This dependent on setting for a neuron during preparing is alluded to as intricate co-transformations. You can envision that if neurons are arbitrarily exited the organization during preparing, that other neuron should step in and handle the portrayal needed to make forecasts for the missing neurons. This is accepted to bring about various autonomous inner portrayals being found out by the organization. The impact is that the organization turns out to be less touchy to the particular loads of neurons. This thus brings about an organization that is prepared to do better speculation and is less inclined to overfit the preparation information.

The Complete Discriminator architecture is

Discriminator(

(cnv1): Sequential(

(0): Conv2d(3 – input size, 32 – number of filters, (4, 4) – the size of filters, (2, 2) – stride for compression, 1, 1) – of padding for corner pixel values, bias set to False)

)

(cnv2): Sequential(

(0): Conv2d(32 – input size, 64– number of filters, (4, 4) – the size of filters, (2, 2) – stride for compression, 1, 1) – of padding for corner pixel values, bias set to False)

(1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

)

(cnv3): Sequential(

(0): Conv2d(64, 128, kernel_size=(4, 4) , (2, 2) – stride for compression, 1, 1) – of padding for corner pixel values, bias set to Fasle)

(1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

)

(fcl): Linear(in_features=2048, out_features=1, bias=True)

(out): Sigmoid()

(dropout): Dropout(p=0.5, inplace=False)

)

Presently next we characterized the engineering of the generator. It is actually equivalent to a discriminator yet in a contrary manner. We have utilized Transposed Convolutions for that. Exactly when we're deciphering convolutions we change the solicitation for the estimations in this convolution action organization, which has interesting effects and prompts different practices to the conventional convolutions. At times we see this activity alluded to as a 'deconvolution' yet they are not the same. A deconvolution endeavors to invert the impacts of a convolution. Albeit rendered convolutions can be utilized for this, they are more adaptable. Other legitimate names for rendered convolutions we may find in the wild are 'partially stridden convolutions' and 'up convolutions'.We utilize normal convolutions to pack the info information into a theoretical spatial portrayal, and afterward utilize translated convolutions to decompress the theoretical portrayal into something of utilization. A convolutional auto-encoder is entrusted with reproducing its information picture, subsequent to going halfway outcomes through a 'bottleneck' of restricted size. Employments of auto-encoders incorporate pressure, commotion evacuation, colorization, and in-painting. Achievement relies upon having the option to learn dataset-explicit pressure in the convolution parts and dataset-explicit decompression in the rendered convolution pieces. The boundary we utilizing here is equivalent to Discriminator.

The final generator architecture is

Generator(

(fcl): Linear (100 – input size, 2048 – output size, bias set to true)

(dcnv1): Sequential(

(0): ConvTr2d(128 – input size, 64 – number of filters , (4, 4) – the size of filters, (2, 2)) – stride for compression, (1, 1) – of padding for corner pixel values, bias set to Fasle)

(1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

)

(dcnv2): Sequential(

(0): ConvTranspose2d(64– input size, 32– number of filters, (4, 4)) – the size of filters, (2, 2) – stride for compression, (1, 1) – of padding for corner pixel values, bias set to False)

(1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

)

(dcnv3): Sequential(

(0): ConvTranspose2d(32– input size, 3– number of filters, (4, 4) – the size of filters, (2, 2) – stride for compression, (1, 1) – of padding for corner pixel values, bias set to False)

)

(dropout): Dropout(p=0.5, inplace=False)

)

At that point the subsequent stage is characterizing the misfortune work without a doubt and phony misfortune. We have utilized BCEWithLogitsLoss. This misfortune joins a Sigmoid layer and the BCELoss in one single class. This adaptation is more mathematically stable than utilizing a plain Sigmoid followed by a BCELoss as, by consolidating the activities into one layer, we exploit the log-total exp stunt for mathematical solidness. We characterized counterfeit name as zero and genuine mark as 1 to recognize the picture and figure the misfortune during the feedforward and backpropagation. We additionally not utilizing precisely 1 incentive as a mark for the genuine names. We utilized a 0.9 incentive for smoothening the yield.

Adam is an advanced calculation that can be utilized rather than the traditional stochastic angle plunge methodology to refresh network loads iterative dependent on preparing information. Adam was introduced by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper (banner) named "Adam: A Method for Stochastic Optimization". We referred to liberally from their paper in this post aside from whenever communicated something different.

Adam is a substitution advancement calculation for stochastic slope plummet for preparing profound learning models. Adam consolidates the best properties of the AdaGrad and RMSProp calculations to give an enhancement calculation that can deal with inadequate slopes on boisterous issues. Adam is moderately simple to design where the default arrangement boundaries excel on most issues.

We used alpha 0.0005 also referred to as the learning rate or step size. We used beta1 0.1, the exponential decay rate for the first moment estimates beta2 0.99, the exponential decay rate for the second-moment estimates. The next step is the training of the whole Deep Convolution Neural Network. It is a two-step process where we train both our discriminator and generator. Firstly, we train our discriminator by using real images. The dataset that is used here is the celebA dataset. The pictures in this dataset cover huge posture varieties and foundation mess. CelebA has enormous varieties, huge amounts, and rich comments. The real images are passed through the discriminator where it learns that this is how real images look like. And generates a real loss for the said images Next during the first pass training random noise is passed through the generator where it generates fake images from it that are to be passed through the discriminator. A key point here is that these are images have exactly the size or dimensions as the input capacity of the discriminator. The discriminator then generates the loss for these fake images which is known as fake loss. The next step is to add these real and fake losses to do backpropagation. It is the strategy for adjusting loads of a neural net dependent on the mistake rate got in the past age. When we backpropagate we update the filter and weights of the fully connected layers based on the calculated losses.

The next step in training is generator training.

It is trained by passing latent vectors or random noise through a generator from which it generates fake images which are then passed through the discriminator but this time it generates a loss for fake images using flipped labels. The same backpropagation process is then repeated to finish the training.

All this training is done multiple times in the form of epochs. An epoch is essentially nothing but an iteration of the training cycle.

IV. RESULTS

After running 150 epochs of the network the following results have been gathered

1. Average and minimum loss after 150 epochs:
Avg Loss of Discriminator: 0.944
Avg Loss of Generator: 2.140
Minimum Loss of Disc: 0.54910
Minimum Loss of Gen:0.54958

2. PSNR value graph of 4 samples out of 16

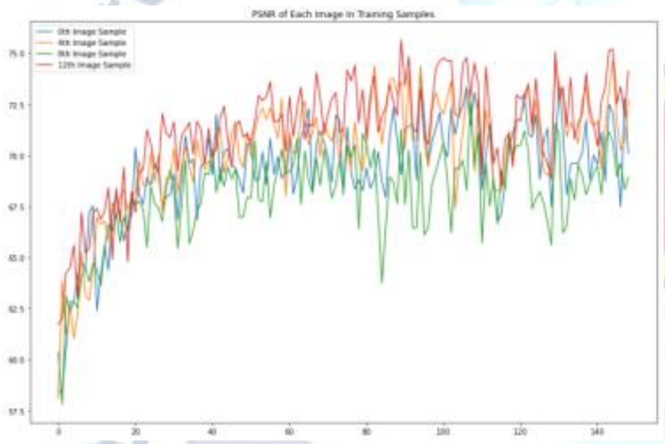

Fig1. Psnr of 4 samples

3. Average PSNR value graph

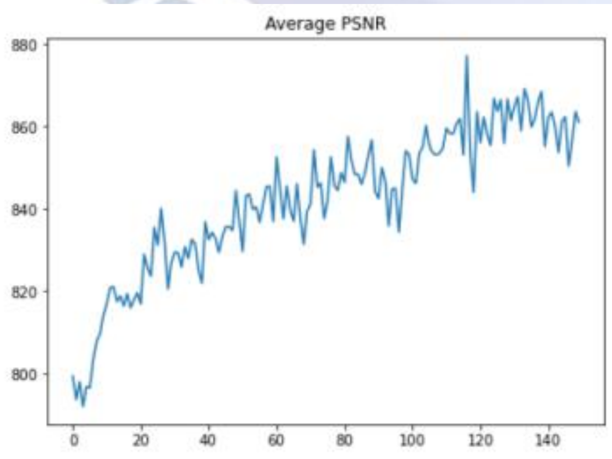

Fig2. Average Psnr

4. Image samples after 10 epochs

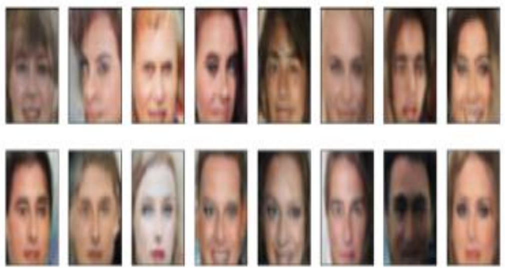

Fig3. Image samples after 10 epochs

5. Image samples after 50 epochs

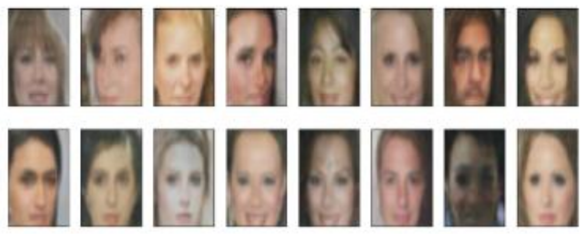

Fig4. Image samples after 50 epochs

6. Image samples after 100 epochs

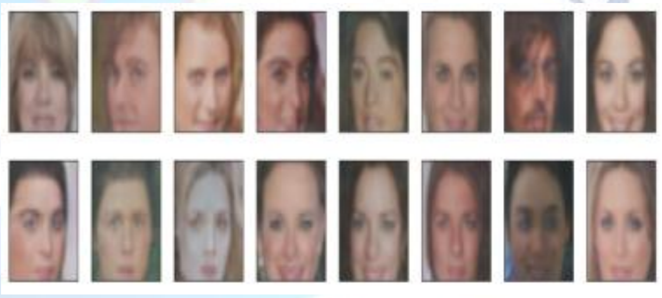

Fig5. Image samples after 100 epochs

7. Image Samples after 150 epochs

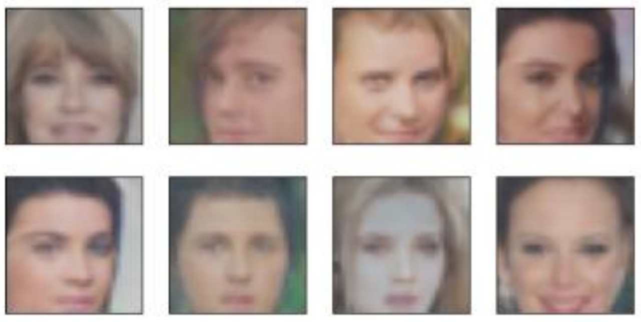

Fig6. Image samples after 150 epochs

VI. FUTURE SCOPE AND CONCLUSION

The boom in interest in GANs is not only because of their ability to transform latent data into meaningful information but it is also because of their potential to generate large amount of results with unlabelled data. Using this ability, we have prepared a generative model that learns from an existing dataset and expands upon it. Our model uses celebA dataset and uses it to train the discriminator and generator.

Which produces the desired image data. There are many oppurtunities to use this kind of procedure to expand upon existing datasets naturally.

REFERENCES

[1] Generative Adversarial NetworkIan J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio 2014.

[2] Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks Alec Radford, Luke Metz, Soumith Chintala 2016

[3] Generative Adversarial Networks: An Overview Antonia Creswell , Tom White, Vincent Dumoulin , Kai Arulkumaran§ , Biswa Sengupta and Anil A Bharath§ , Member IEEE BICV Group, Dept. of Bioengineering, Imperial College London. School of Design, Victoria University of Wellington, New Zealand MILA, University of Montreal, Montreal H3T 1N8 Cortexica Vision Systems Ltd., London, United Kingdom 2017

[4] A NOTE ON THE EVALUATION OF GENERATIVE MODELS Lucas Theis, Aaron van den Oord ,Matthias Bethge 2016

[5] Z. Pan, W. Yu, X. Yi1, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (GAN): A survey," IEEE Access, vol. 7, pp. 36322–36333, 2019.

[6] Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A.A. Bharath, “Generative adversarial networks: An overview,” IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53– 65, 2018

[7] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye,” A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications,” arXiv preprint arXiv: 2001.06937, 2020.

[8] Z. Wang, Q. She, and T.E. Ward, “Generative Adversarial Networks: A Survey and Taxonomy,” arXiv preprint arXiv: 1906.01529, 2019.

[9] L. Tran, X. Yin, and X. Liu, “Disentangled representation learning gan for pose-invariant face recognition,” in IEEE Conference on Computer Vision and Pattern Recognition, 2017.