IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2320-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 10, Issue 2, February 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.512

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

Sketch to Face Conversion Using GAN for Forensic Research

Naumanurrahman Shaikh Mujiburrahman, Dr. Manoj E. Patil

UG Student, Dept. of Computer Engineering, SSBT COET, Bambhori, Jalgaon, India

Assoc. Professor., Dept. of Computer Engineering, SSBT COET, Bambhori, Jalgaon, India

ABSTRACT: In this paper we have successfully implemented the conversion of sketch into an colorful face image to use in Forensic Research for converting the sketch drawn from the information of the witness into real image which can be further used for the investigation purpose. We have used the celebrity dataset from kaggle for the face images .We have converted that dataset into the dataset useful for the pix2pix.Using GANs pix2pix we have achieved this goal. A generator network use existing data to generate new data. The discriminator network tries to differentiate between the real data and the data generated by the generator network. This process is repeated until the discriminator unable to differentiate between the generated image and real image and when this process happens we can get the image.

KEYWORDS: Generative Adversarial Networks (GAN),PIX2PIX , Generator, Discriminator,Real Data, Fake Data, Accuracy, Loss , Training, Validation

I. INTRODUCTION

In the Forensic department of any criminal investigation there lots of cases in which the face of culprit is unknown. But the Sketch of the culprit is available which is created the artist using the information provided by the eye witness of the incident these sketches are available in abundance at the office. In the paper we have successfully implemented the novel approach to convert these in the colorful images to help the further investigation.
The sketches are converted into the colorful images using Generative Adversarial Networks Pix2Pix implemented in Google Colaboratory platform using python language version 3.0. Using the dataset of celebrity faces available at the kaggle we have created the dataset useful for the pix2pix.

In Generative Adversarial Networks uses two networks one Generator and another Discriminator. Generator use the existing data in our case images to generate image and Discriminator differentiate between the data produced and the original data, in our case the images generated from the sketches.
Face Sketch Colorization via Supervised GANs[12] in this paper they have performed the operation in two steps first they converted Grayscale sketch into Grayscale image then that Grayscale image into color image But in this paper we proposed a method to directly convert the sketch into colorful image. We have to give honorary mention to the project of Hafsa Sheikh on edge to shoes conversion[9] from where we have taken inspiration.
The generated image can be used for further investigation like searching in database using the image and finding the persons information or the ground investigation can be done smoothly rather than just using sketch .

A. Generative Adversarial Networks (GAN)

A generative adversarial network (GAN) is a class of machine learning frameworks designed by Ian Goodfellow and his colleagues in 2014.[1] Two neural networks contest with each other in a game (in the form of a zero-sum game, where one agent's gain is another agent's loss).
Given a training set, this technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Though originally proposed as a form of generative model for unsupervised learning, GANs have also proven useful for semi-supervised learning,[2] fully supervised learning,[3] and reinforcement learning.[4]
The core idea of a GAN is based on the "indirect" training through the discriminator, which itself is also being updated dynamically.[5] This basically means that the generator is not trained to minimize the distance to a specific image, but rather to fool the discriminator. This enables the model to learn in an unsupervised manner.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

B. Generator Network

Generator Network is a network which generate the fake data using the dataset provided to it. GAN learns to create fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real.Generator training requires tighter integration between the generator and the discriminator than discriminator training requires. The portion of the GAN that trains the generator includes:

- random input
- generator network, which transforms the random input into a data instance
- discriminator network, which classifies the generated data
- discriminator output
- generator loss, which penalizes the generator for failing to fool the discriminator

C. Disciminator Network

The discriminator in a GAN is simply a classifier. It tries to distinguish real data from the data created by the generator. It could use any network architecture appropriate to the type of data it's classifying.\

The discriminator's training data comes from two sources:

- **Real data** instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
- **Fake data** instances created by the generator. The discriminator uses these instances as negative examples during training.

D. Loss Functions

EGANs try to replicate a probability distribution. They should therefore use loss functions that reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data.

How do you capture the difference between two distributions in GAN loss functions? This question is an area of active research, and many approaches have been proposed. We'll address two common GAN loss functions here, both of which are implemented in TF-GAN:

- **minimax loss**: The loss function used in the paper that introduced GANs.
- **Wasserstein loss**: The default loss function for TF-GAN Estimators. First described in a 2017 paper.

TF-GAN implements many other loss functions as well.

II. METHODOLOGY

The generative network generates candidates while the discriminative network evaluates them.[1] The contest operates in terms of data distributions. Typically, the generative network learns to map from a latent space to a data distribution of interest, while the discriminative network distinguishes candidates produced by the generator from the true data distribution. The generative network's training objective is to increase the error rate of the discriminative network (i.e., "fool" the discriminator network by producing novel candidates that the discriminator thinks are not synthesized (are part of the true data distribution)).[1][6]

A known dataset serves as the initial training data for the discriminator. Training it involves presenting it with samples from the training dataset, until it achieves acceptable accuracy. The generator trains based on whether it succeeds in fooling the discriminator. Typically the generator is seeded with randomized input that is sampled from a predefined latent space (e.g. a multivariate normal distribution). Thereafter, candidates synthesized by the generator are evaluated by the discriminator. Independent backpropagation procedures are applied to both networks so that the generator produces better images, while the discriminator becomes more skilled at flagging synthetic images.[7] The generator is typically a deconvolutional neural network, and the discriminator is a convolutional neural network.

GANs often suffer from a "mode collapse" where they fail to generalize properly, missing entire modes from the input data. For example, a GAN trained on the MNIST dataset containing many samples of each digit, might nevertheless timidly omit a subset of the digits from its output. Some researchers perceive the root problem to be a weak discriminative network that fails to notice the pattern of omission, while others assign blame to a bad choice of objective function. Many solutions have been proposed.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

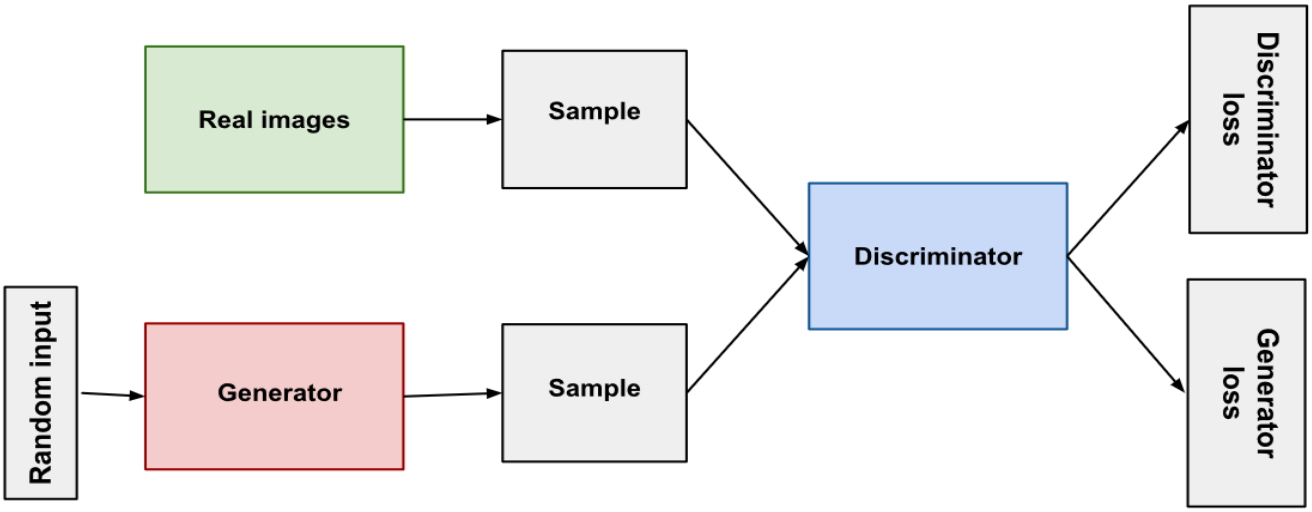


Fig1:-GAN Network

Both the generator and the discriminator are neural networks. The generator output is connected directly to the discriminator input. Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights.

A. Training The Discriminator

The discriminator connects to two loss functions. During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss. We use the generator loss during generator training, as described in the next section.

During discriminator training:

1. The discriminator classifies both real data and fake data from the generator.
2. The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
3. The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

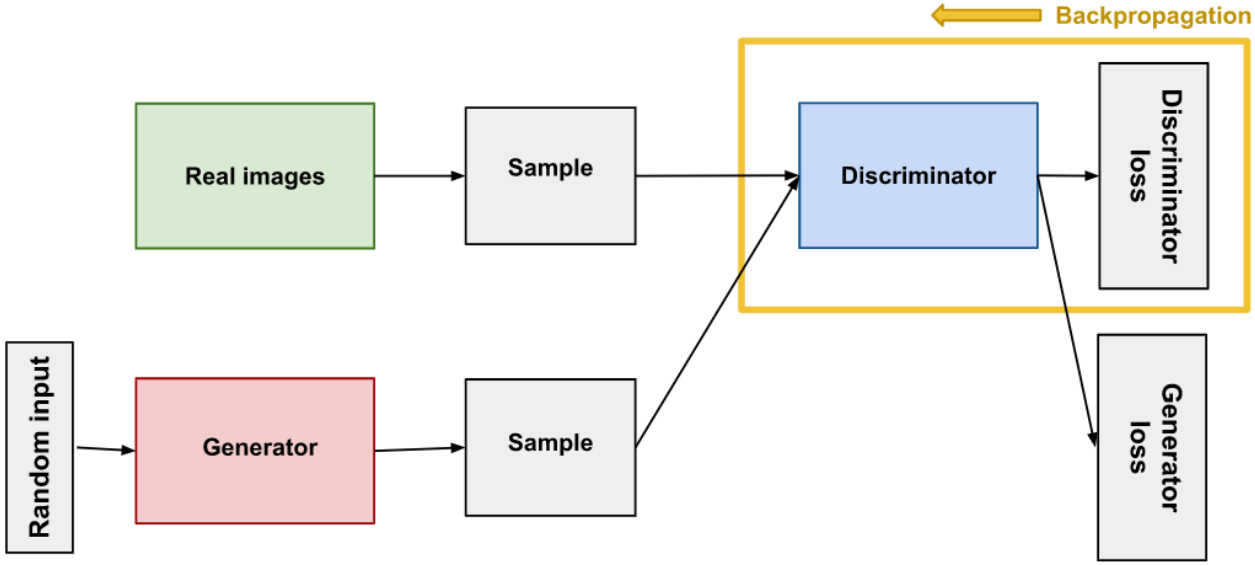


Fig2:-Backpropagation in discriminator training

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

B. Training the Generator

To train a neural net, we alter the net's weights to reduce the error or loss of its output. In our GAN, however, the generator is not directly connected to the loss that we're trying to affect. The generator feeds into the discriminator net, and the *discriminator* produces the output we're trying to affect. The generator loss penalizes the generator for producing a sample that the discriminator network classifies as fake.

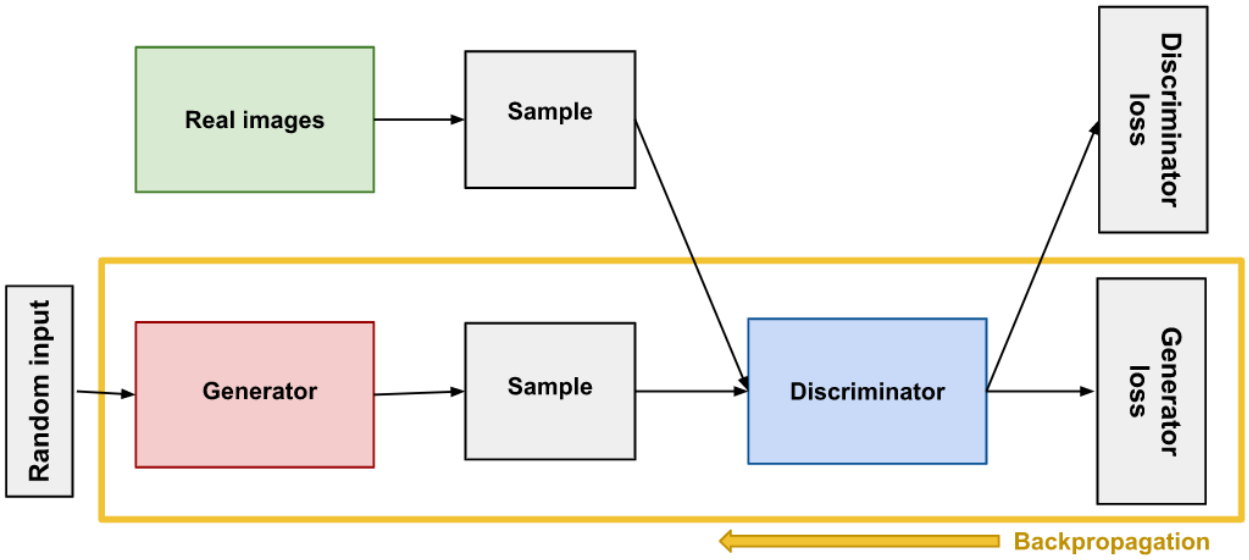
This extra chunk of network must be included in backpropagation. Backpropagation adjusts each weight in the right direction by calculating the weight's impact on the output — how the output would change if you changed the weight. But the impact of a generator weight depends on the impact of the discriminator weights it feeds into. So backpropagation starts at the output and flows back through the discriminator into the generator.

At the same time, we don't want the discriminator to change during generator training. Trying to hit a moving target would make a hard problem even harder for the generator.

So we train the generator with the following procedure:

1. Sample random noise.
2. Produce generator output from sampled random noise.
3. Get discriminator "Real" or "Fake" classification for generator output.
4. Calculate loss from discriminator classification.
5. Backpropagate through both the discriminator and generator to obtain gradients.
6. Use gradients to change only the generator weights.

This is one iteration of generator training. In the next section we'll see how to juggle the training of both the generator and the discriminator.



.Fig3:-Backpropagation in generator training

C. Training the GAN

Because a GAN contains two separately trained networks, its training algorithm must address two complications:

- GANs must juggle two different kinds of training (generator and discriminator).
- GAN convergence is hard to identify.
- **Alternating Training**

The generator and the discriminator have different training processes. So how do we train the GAN as a whole?

GAN training proceeds in alternating periods:

1. The discriminator trains for one or more epochs.
2. The generator trains for one or more epochs.
3. Repeat steps 1 and 2 to continue to train the generator and discriminator networks.

We keep the generator constant during the discriminator training phase. As discriminator training tries to figure out how to distinguish real data from fake, it has to learn how to recognize the generator's flaws. That's a different problem for a thoroughly trained generator than it is for an untrained generator that produces random output.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

Similarly, we keep the discriminator constant during the generator training phase. Otherwise the generator would be trying to hit a moving target and might never converge.
It's this back and forth that allows GANs to tackle otherwise intractable generative problems. We get a toehold in the difficult generative problem by starting with a much simpler classification problem. Conversely, if you can't train a classifier to tell the difference between real and generated data even for the initial random generator output, you can't get the GAN training started.

- **Convergence**

As the generator improves with training, the discriminator performance gets worse because the discriminator can't easily tell the difference between real and fake. If the generator succeeds perfectly, then the discriminator has a 50% accuracy. In effect, the discriminator flips a coin to make its prediction.
This progression poses a problem for convergence of the GAN as a whole: the discriminator feedback gets less meaningful over time. If the GAN continues training past the point when the discriminator is giving completely random feedback, then the generator starts to train on junk feedback, and its own quality may collapse.
For a GAN, convergence is often a fleeting, rather than stable, state.

III. IMPLEMENTATION

A. Dataset

This dataset contain 200000 images of the celebrity faces. From this dataset we have crated the dataset of 20000 images and the sketches associated with them. In this we have specially converted those images into sketches and then joined both image and sketch into the image of 512*256(width*height).

Fig4:-Sample Image from dataset

B. Code

Import statement is used to import a library into the code so that we can use it's functionalities in the code. In this code we import different libraries to use the functionality of those libraries in the code.
We loaded the dataset from the local index of the computer and we applied random jittering and mirroring techniques to the training dataset.

- In random jittering, the image is resized to 286 x 286 and then randomly cropped to 256 x 256
- In random mirroring, the image is randomly flipped horizontally i.e left to right.

In this code load function is used to load an image from the dataset using tensorflow library.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

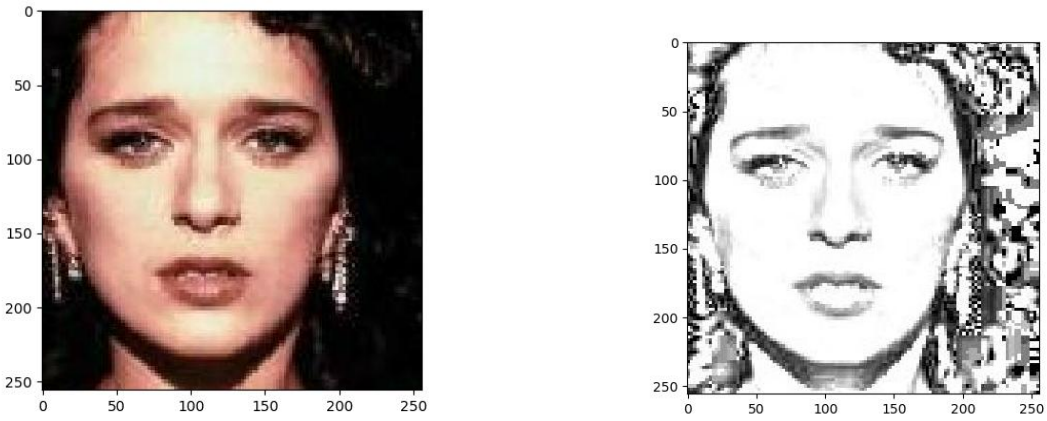


Fig5:Sample Images Loaded From Dataset

The image loaded looks like the above images.
Then we defined the resize and **random_crop functions()** to resizing the image by changing the values of image attributes.
Then we defined **normalize function()** to normalize an image by changing the range of pixel intensity value.
@tf.function() is used to construct a callable that execute a tensorflow graph created by trace-compiling the tensorflow operations in functions and random_jittering function is used randomly cropping for the image by Resize an image to bigger height and width. Randomly crop to the target size. Randomly flip the image horizontally. Using random jittering the image can be seen as:

Fig6:-Output after random jitter

load_image_train() function is used to load image for the training dataset.
load_image_test () function is used to load image for the testing dataset.
Input pipeline is used to chain multiple estimators into one and hence, automate the machine learning process. This is extremely useful as there are often fixed sequence of steps in processing the data.

Building Generator:
GANs has two neural network one is a generator that synthesizes new samples from scratch.The generator is not trained directly and instead is trained via a discriminator network. Under build a generator following functions are used.

- **Downsample()** will randomly sample a data set so that all classes have the same frequency as the minority class.
- **Upsample()** samples with replacement to make the class distributions equal.

Generator() is a special type of function which does not return a single value, instead it returns an iterator object with a sequence of values. It can be used in a for loop.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

Generator loss:

GANs try to replicate a probability distribution, therefore use loss function that reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data.It is a sigmoid cross entropy loss of the generated images and an array of ones. This allows the generated image to become structurally similar to the target image.

Build a Discriminator:

The discriminator is trained directly on real and generated image and is responsible for classifying images as real or fake. So the **Discriminator**() receives 2 inputs.

- Input image and the target image, which it should classify as real.
- Input image and the generated image (output of generator), which it should classify as fake.
- We concatenate these 2 inputs together in the code (tf.concat([inp, tar], axis=-1)

Discriminator loss:

As the generator as loss function as well as discriminator also has loss function The discriminatorloss function takes 2 inputs; real images, generated images real_loss is a sigmoid cross entropyloss of the real images and an array of ones(since these are the real images) generated_loss is asigmoid cross entropy loss of the generated images and an array of zeros(since these are the fakeimages). Then the total_loss is the sum of real_loss and the generated_loss.

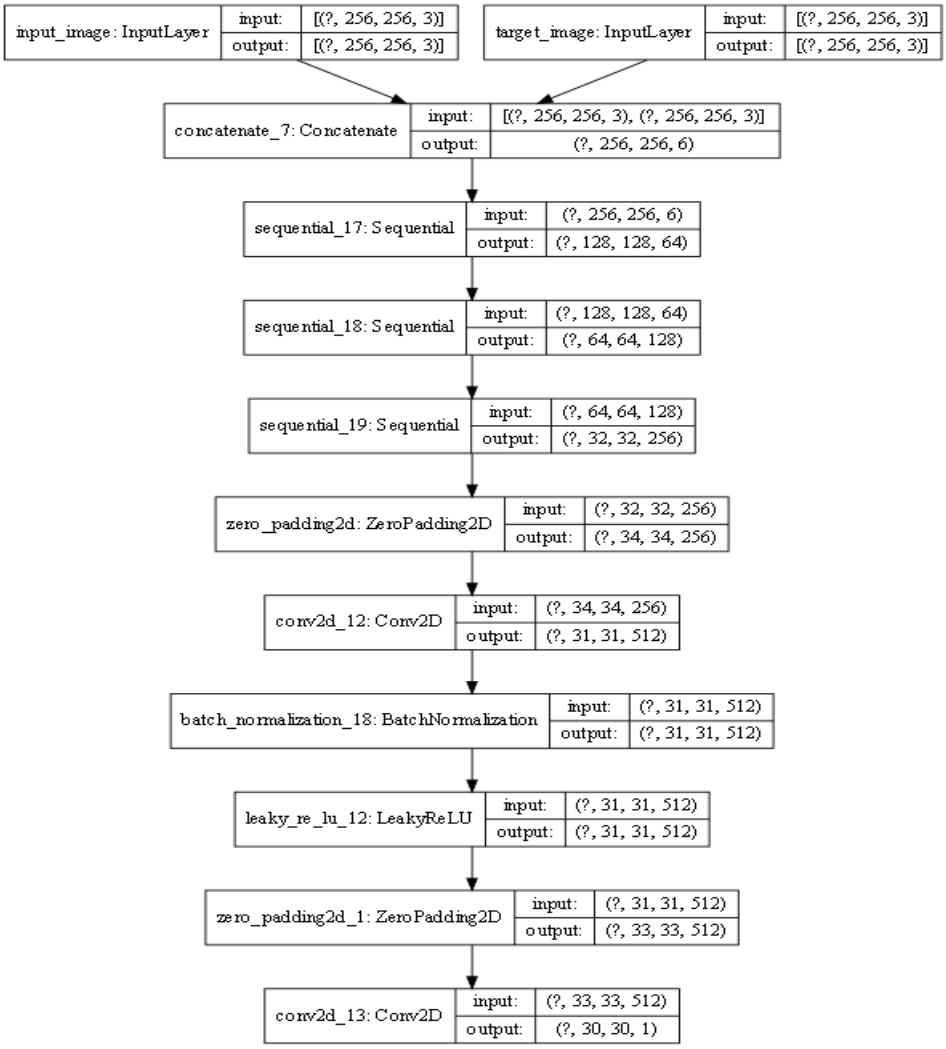


Fig7:-Network architecture

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

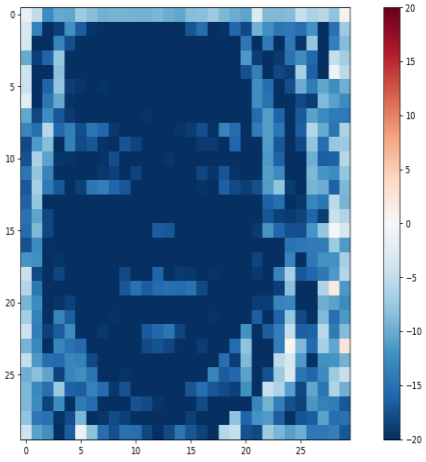


Fig8:-Output of colorbar()

Optimizers and Checkpoint-saver:

Optimizers are the extended class, which include added information to train a specific model. Itis used for improving speed and performance for training a specific model.Checkpoint are binaryfiles in a proprietary format which map variable names to tensor values. The best way toexamine the contents of a checkpoint is to load it using a server. Server can automaticallynumbers checkpoint filenames with counters.

Training:

To train our dataset discriminator will be used and do the following steps:

- The discriminator receives the input_image and the generated image as the first input. The second input is the input_image and the target_image.
- Next, we calculate the generator and the discriminator loss.
- Then, we calculate the gradients of loss with respect to both the generator and the discriminator variables(inputs) and apply those to the optimizer.
- Then log the losses to TensorBoard.

The actual training loop:

- Iterates over the number of epochs. •

On each epoch it clears the display, and runs generate_images to show it's progress.

- On each epoch it iterates over the training dataset, printing a '.' for each example.
- It saves a checkpoint every 20 epochs.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

IV. RESULTS AND DISCUSSION

A. *Result*

After completion of 0 the image generated by the generator is.

Fig9:-Image Generated After Epoch 0

Then the Training Process is done the training dataset for epoch And the model is saved for the further use in future. After 1 epoch the image generated is.

Fig10:-Image Generated After 1 epoch

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 10, Issue 2, February 2021 ||

DOI:10.15680/IJIRSET.2021.1002069

V. CONCLUSION AND FUTURE SCOPE

A. Conclusion

After performing the whole process we have successfully converted the given sketch into the image of an colorful face. With less computation power machine and even with a moderate gpu GAN can perform well. After the completion of the program the trained model is saved for the future reference and use.

B. Future Scope

The image generated in the process can be further used in the program with face recognition ability to find the information of a person if he is in the database. A model can be build using face recognition and the image generator that can verify the person straight from the sketch.

REFERENCES

[1] Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). Generative Adversarial Networks (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[2] ^ Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec; Chen, Xi (2016). "Improved Techniques for Training GANs". arXiv:1606.03498 [cs.LG].

[3] ^ Isola, Phillip; Zhu, Jun-Yan; Zhou, Tinghui; Efros, Alexei (2017). "Image-to-Image Translation with Conditional Adversarial Nets". Computer Vision and Pattern Recognition.

[4] ^ Ho, Jonathon; Ermon, Stefano (2016). "Generative Adversarial Imitation Learning". Advances in Neural Information Processing Systems: 4565–4573. arXiv:1606.03476. Bibcode:2016arXiv160603476H.

[5] ^ "Vanilla GAN (GANs in computer vision: Introduction to generative learning)". theaisummer.com. AI Summer. Archived from the original on 2020-06-03. Retrieved 20 September 2020.

[6] ^ Luc, Pauline; Couprie, Camille; Chintala, Soumith; Verbeek, Jakob (2016-11-25). "Semantic Segmentation using Adversarial Networks". NIPS Workshop on Adversarial Training, Dec, Barcelona, Spain. 2016. arXiv:1611.08408. Bibcode:2016arXiv161108408L.

[7] ^ Andrej Karpathy; Pieter Abbeel; Greg Brockman; Peter Chen; Vicki Cheung; Rocky Duan; Ian Goodfellow; Durk Kingma; Jonathan Ho; Rein Houthooft; Tim Salimans; John Schulman; Ilya Sutskever; Wojciech Zaremba, Generative Models, OpenAI, retrieved April 7,2016

[8] ^ Lin, Zinan; et al. (December 2018). "PacGAN: the power of two samples in generative adversarial networks". NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 1505–1514. (also available arXiv:1712.04086)

[9] https://github.com/HafsaSheikh/edges2shoes

[10] https://en.wikipedia.org/wiki/Generative_adversarial_network

[11] https://developers.google.com/machine-learning/gan

[12] Face Sketch Colorization via Supervised GANs Ramya Y.S., Soumyadeep Ghosh, Mayank Vatsa, Richa Singh IIIT Delhi, India {yellapragada15117, soumyadeepg, mayank, rsingh}@iiitd.ac.in

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.512

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details