NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]
ISSN: 2394-3696 Website: ijiert.org
VOLUME 8, ISSUE 6, June. -2021

GENERATIVE ADVERSARIAL NETWORK BASED PORTRAIT BUILDING SYSTEM FOR CRIMINAL IDENTIFICATION

Mahendra Pankaj
Department of computer science, Govt. Engineering College, Raipur, C.G.
pankajmahendra1006@gmail.com

Ashutosh Singh Netam
Department of computer science, Govt. Engineering College, Raipur, C.G.
ashutoshsinghnetam750@gmail.com

Janak Rajwade
Department of computer science, Govt. Engineering College, Raipur, C.G.
jrajwade39@gmail.com

ABSTRACT
In today's world the un-identification of a criminal is a major problem, causing which many of the cases to remain unsolved. So here we are using App based portrait building software from the photos of the criminals available with us. Extracting representative features of eyes, eyebrow, forehead, nose, ears, beard, hair styles etc. in various classes. In this Paper, a Generative adversarial network (GAN) [1] based Model is developed in which we train the model through a prepared database of criminal's facial descriptions

Keywords: Portrait Building System, Generative Adversarial Network, Discriminator, Generator.

INTRODUCTION
Crime scene in India has been drastically drafted and with it also the identification of the culprit. Which led the crime department of India to many unsolved cases throughout India. So, here we propose a system where A complete human-like sketch can be generated with the appropriate inputs and this can be altered based on the different inputs from the user. Domain interpretation of the PBS System is given in figure 1.

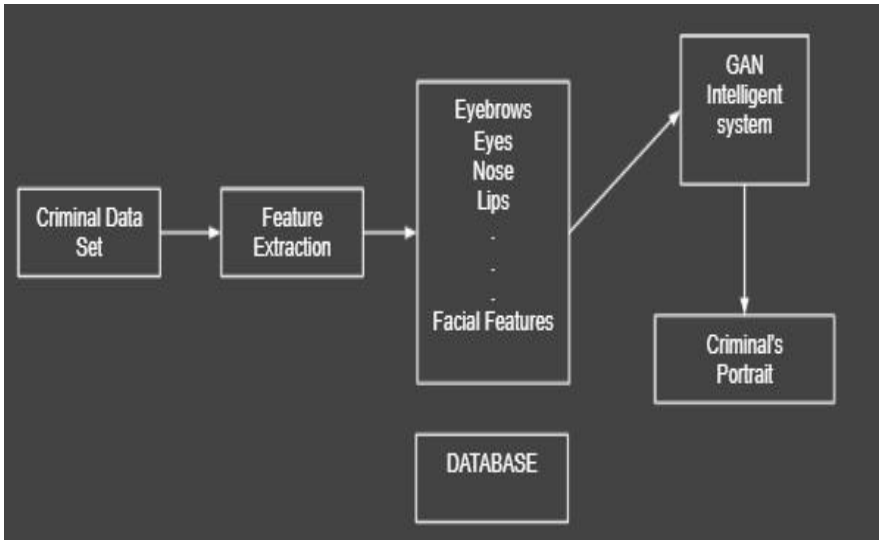

Figure 1: Domain specification App Based PBS System

The upper domains of Criminal's Facial Data extraction like eyes, nose, lips, eyebrows, etc. are extracted from the full face portrait. After which the data is fed to our Generative Adversarial Network (GAN)[1] Through which our model gets trained to provide the desired output.

The Processing of the data extracted by the user GAN[1] Starts it's training process in which it first concatenates the facial descriptions provided by the user and after which it proceeds to the GAN Network Components[3] One-by-One. Flow of data processing shown in figure 2,

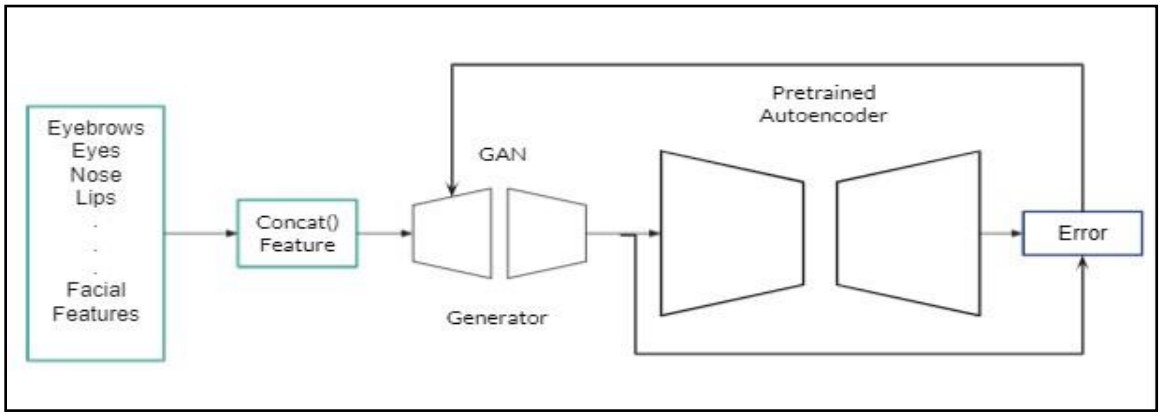

Figure 2: Data Flow in Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN)

A generative adversarial network (GAN) is a deep neural [3] network framework which is able to learn from a set of training data and generate new data with the same characteristics as the training data A generative adversarial network [1] trained on portrait image of human faces can generate realistic-looking faces which can be entirely fictional.

Generative adversarial networks [1] consist of two neural networks[3] in itself, the generator and discriminator, they both compete against each other to provide the better output. The generator is trained to generate fake data, and the discriminator [1] is trained to distinguish the fake data from the generator and real examples. The generator is punished if it produces fake data, such as an image, whose discriminator can identify as fake and easily recognizable. With time, the generator is trained to generate more convincing examples.

Generator

It is a mechanism that learns to produce realistic fake data from random Data. The fake examples produced by the generator are used as negative samples to train the discriminator. Both the generator fake samples and the real sample training sets are randomly fed into the discriminator network. The discriminator is not aware of whether a particular input is from the generator or from the training set.

Discriminator

It is a mechanism which learns to distinguish fake data from realistic data. If the generator produces unconvincing images, the discriminator penalizes the generator. Initially, before training has begun, the generator's fake output can be very easily recognizable for the discriminator.

Since the generator's output is fed directly to the discriminator as input, this means that when the discriminator classifies the generator's output, it can back-propagation algorithm[2] across the network to update the generator's weight.

The discriminator is just a binary classifier, ending with an appropriate function like the Soft-max function. The discriminator outputs an array such as

[0.71] - Estimated Probability that the input image can be fake by discriminator
[0.29] - Estimated Probability by the discriminator that input image can be real

The Discriminator's input comes from two basic sources:
1. The Training Data Set, Where it contains the real photos of faces
2. The generator, which generates synthetic faces
We hold the generator's [1] weights constant while training the discriminator to produce negative examples for the discriminator [1].

Discriminator Training Process
At first Pass some real examples from the dataset, and some fake examples from the generator, into the discriminator as an input.
The discriminator classifies the input into real and fake.
Then calculate the discriminator's loss using a suitable function as the cross-entropy loss.
Updating of the discriminator's weights through back-propagation.[2]

Generator Training Process
At the starting of the training, initialisation of both the generator and discriminator with random weights to be done.
For each training iteration, passing a random seed (Photo) into the generator as input. Some random noise is propagated through the generator after which outputs a synthetic example, such as an image.
The generator's output is then passed as an input into the discriminator network, and therefore the discriminator classifies it as 'real' or 'fake'.
Calculate the loss function of the generator. The generator's loss function indicates how much the generator was gimmicking the discriminator.
After then we use the back-propagation algorithm [2] through both the discriminator and generator, to determine and adjust the generator's weights in order to improve the generator loss function

Training Process of Generative Adversarial Network (GAN)
Generators and discriminators need to be trained separately as they have their own individual loss functions. It can be done by alternating between the two:
1. Training the discriminator in one or more epochs while keeping the generator's weight constant.
2. Training the generator for one or more epochs, keeping the discriminator weights constant.
Repeating steps (1) and (2) until we determine that the network has converged.

Convergence in a Generative Adversarial Network
Once the generator is able to produce fake photos of people that are indistinguishable from real examples, the discriminator has to go through a different level. For a perfect generator, the discriminator will have only 50% accuracy in distinguishing fakes from genuine examples of the dataset.
Which means that the discriminator gives feedback, which is used to train the generator, becomes less meaningful over time, and ultimately becomes completely random.
If we continue to train the network after this point, then the discriminator's feedback can cause the generator's quality to go down. For this, it is important to evaluate the quality of the generated output and stop training once the discriminator has 'lost' the game to the generator.

$$L = E_x[\log D(x)] + E_z[\log(1 - D(G(z)))]$$

Figure 3: Loss Function of a Generative Adversarial Network

The generator tries to minimize the output of the above loss characteristic and the discriminator tries to maximize it. From this way a single loss function can be used for both the generator and discriminator.

Training of our Generative Adversarial Network (GAN)

We have made a dataset of random faces for training a generative adversarial network to generate a new facial portrait of a person according to the facial description. Below is a sample of a Portrait image of a person from our dataset. This portrait image dataset is a database of 619 images of random persons, with dimensions 200x200 pixels.

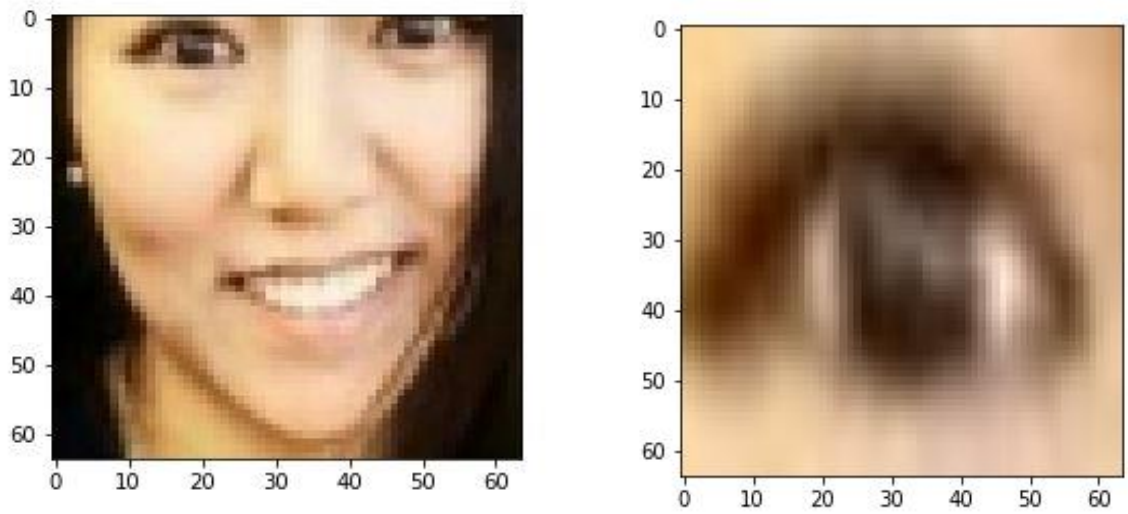

Figure 4 & 5 Sample of the data from Portrait Image Dataset

After we initialize the generative adversarial network, the images generated by the generator will be pure noise, and will look like this:

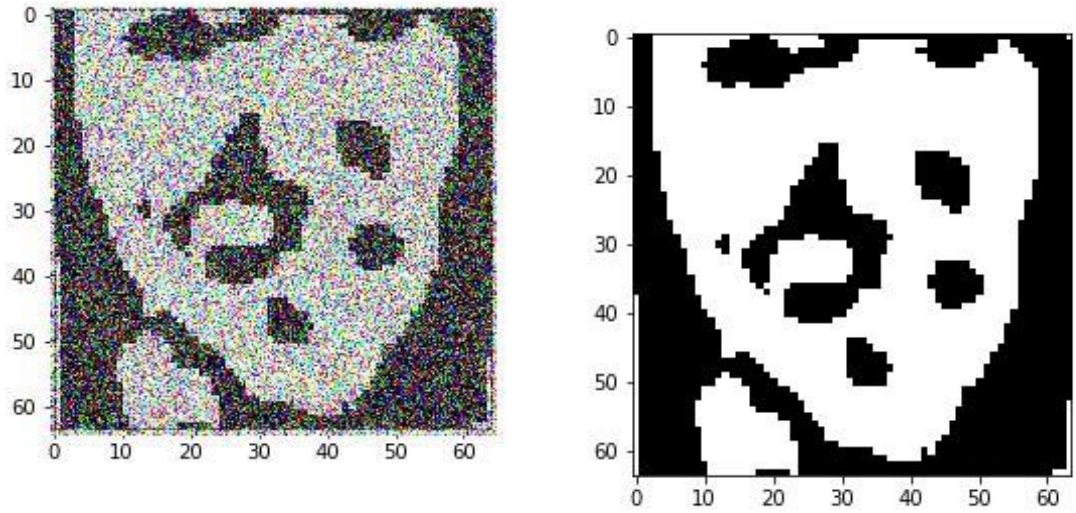

Figure 6: Noisy portrait generated by the generator

Because this noise is very different from a portrait image of the person, the discriminator immediately learns to distinguish between the generated and fake data.

The generator then starts to figure out how to deceive the discriminator. After seven epochs (passing the whole dataset through the generative adversarial network seven times) the generator starts producing random images that resemble the portrait image of a person. The discriminator's job is becoming more difficult.

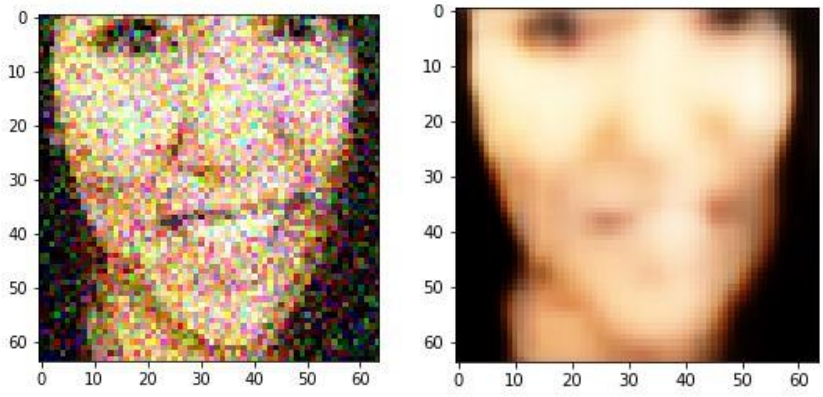

Figure 7, 8 & 9: The continuous progress made by generator in terms of portrait building

After further epochs the generator's output starts becoming recognizable. Above are the generator's output at 10, 20 and 30 epochs respectively. We can see that even a person would have a hard time recognising that this image is artificial, and the discriminator's ability to spot fake samples has now decreased to zero and hence can generate indistinguishable portrait images of a person according to their facial descriptions.

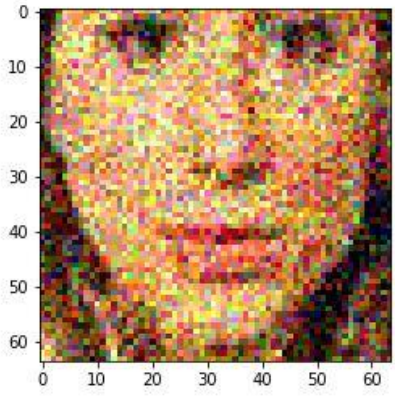

Figure 10: Portrait Generated by the Generator

EXPERIMENTAL SETUP

The Generative adversarial network (GAN) is trained using the Portrait Image dataset. The images of the selected dataset are set to 200*200 pixel size in RGB format, It has total 619 photos. Extracting representative features of eyes, eyebrow, forehead, nose, ears, beard, hair styles etc. in various classes, by cropping them and classifying them as an individual class. For processing and building the GAN network we have Used Jupiter Notebook which is a well-known python IDE for Data Science projects.

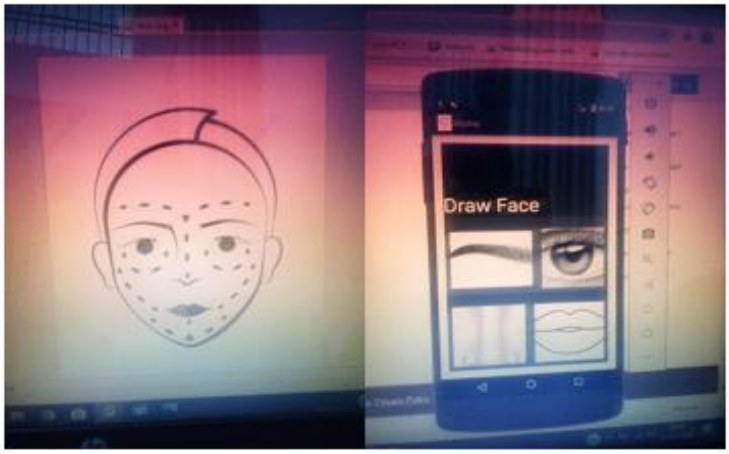

Figure 11 & 12: Portrait Building System Module for The Eye witness

RESULT AND DISCUSSION

The portrait making of a person which is in any need is showcased using different approaches based Generative Adversarial Network. The dataset consists of 619 images with 200x200 pixels. Discriminator and generator have been trained with the dataset to generate human-like portraits. The network performed the training till 50 epochs, according to the increasing numbers of the epochs; it will improve the accuracy rate invalidation as well.

The facial descriptions are described or picked by the Eye witness then the model concatenates all of the features and provides us with a real life-like portrait image of the culprit, or similar to the described facial features.

CONCLUSIONS

In this paper, we applied the generative adversarial network (GAN) using the Portrait Image Dataset which contains 619 photos in it, the network has been performed with a total of 50 epochs providing us with good results.

GANs [1] are very computationally expensive. They require high powered GPUs and a lot of time (a large number of epochs) to produce good results. For our project, we have used the facial features dataset and use it to produce a portrait of a person(culprit).The model is able to generate portrait image of a person's image based on the selection of their facial descriptions provided by an Eye witness, which can be further used to catch the victim of a specific crime

REFERENCES

1) Ian J. Goodfellow, Jean Pouget-Abadie∗ , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair† , Aaron Courville, Yoshua Bengio "Generative Adversarial Nets" Departement d'informatique et de recherche op ´ erationnelle ´ Universite de Montr ´ eal ´ Montreal, QC H3C 3J7
2) Bengio, Y., Thibodeau-Laufer, E., Alain, G., and Yosinski, J. (2014b). Deep generative stochastic networks trainable by backprop. In Proceedings of the 30th International Conference on Machine Learning (ICML'14)
3) Jakub Langr, Vladimir Bok "GANs in Action: Deep learning with Generative Adversarial Networks".