# CARTOONIZING IMAGE

*A Project Report submitted in the partial fulfillment of the requirements*

*for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

Shaik. Gouse Mastan Vali (17471A0502)

D. Krishna Vamsi (17471A0552)

K. Lakshmi Narayana (17471A0542)

A. Sairam(17471A0555)

**Under the Esteemed Guidance Of**

**P. Lakshmi Narayana M.Tech.**

Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET**

**(AUTONOMOUS)**

**(Affiliated to J.N.T.U, Kakinada, approved by AICTE & Accredited by NBA)**

**2020-2021**

**NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPETA**

**(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## CERTIFICATE

This is to certify that the main project entitled ”Cartoonizing Image” is a bonafide work done by “Shaik .Gouse Mastan Vali (17471A0502), D. Krishna Vamsi, K. Lakshmi Narayana, A. Sairam” in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in the Department of **COMPUTER SCIENCE AND ENGINEERING** during 2020-2021.

| | |
|---|---|
| PROJECT GUIDE | PROJECT CO-ORDINATOR |
| **P. Lakshmi Narayana, M.Tech.** | **M. Sireesha, M.Tech.,(Ph.D)** |
| HEAD OF THE DEPARTMENT | EXTERNAL EXAMINER |
| **Dr.S.N.Tirumala Rao, M.Tech.,Ph.D** | |

# ACKNOWLEDGEMENT

we wish to express my thanks to carious personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairman sir **M.V.Koteswara Rao**, **B.sc** who took keen interest in me in every effort throughout this course. We owe out gratitude to our principal **Dr. M.Sreenivasa Kumar,M.Tech,Ph.D** for his kind attention and valuable guidance throughout the course.

we express my deep felt gratitude to **Dr.S.N.TirumalaRao,M.Tech, Ph.D** H.O.D. CSE department and our guide **P. Lakshmi Narayana M.Tech.** of CSE department whose valuable guidance and unstinting encouragement enable me to accomplish my project successfully in time.

we extend sincere thanks to **M. Sireesha,M.Tech.,(Ph.D)** Associate Professor & project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for me throughout this project work.

we extend my sincere thanks to all other teaching and non-teaching staff to department for their cooperation and encouragement during my B.Tech degree.

we have no words to acknowledge the warm affection, constant inspiration and encouragement that I receive from my parents.

we affectionately acknowledge the encouragement received from my friends and those who involved in giving valuable suggestions had clarifying out doubts which had really helped me in successfully completing my project.

**By**

Shaik. Gouse Mastan Vali (17471A0502)

D. Krishna Vamsi (17471A0552)

K. Lakshmi Narayana (17471A0542)

A. Sairam(17471A0555)

## ABSTRACT

This paper presents an approach for image cartoonization. By observing the cartoon painting behavior and consulting artists, we propose to separately identify three white-box representations from images: the surface representation that contains a smooth surface of cartoon images, the structure representation that refers to the sparse color blocks and flatten global content in the celluloid style workflow, and the texture representation that reflects high-frequency texture, contours, and details in cartoon images. A Generative Adversarial Network (GAN) framework is used to learn the extracted representations and to cartoonize images.
The learning objectives of our method are separately based on each extracted representation, making our framework controllable and adjustable. This enables our approach to meet artists' requirements in different styles and diverse use cases. Qualitative comparisons and quantitative analyses, as well as user studies, have been con- ducted to validate the effectiveness of this approach, and our method outperforms previous methods in all comparisons. Finally, the ablation study demonstrates the influence of each component in our framework.

# INDEX

# 1. INTRODUCTION

## 1.1 Introduction

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.

Main theme of our project is to Cartoonizing the image by providing a any kind of image as an input. For this, we are going to develop our project using Deep Learning Technology. For the processing of images, we are going to implement Convolutional Neural Networks (CNN). This project requires huge images dataset.

## 1.2 Existing System

Cartoonizing of images are taking huge time to prepare them manually. In the olden days people used to draw the images to make the cartoon character. The existing system is this, the cartoon pictures need to be developed by drawing or by using artists

### Disadvantages:

1. Artists may not be able to Draw the cartoon image properly.

2. Lot of time is taken to generate cartoon images.

3. It is difficult to cartoonize all types of images with same picture quality.

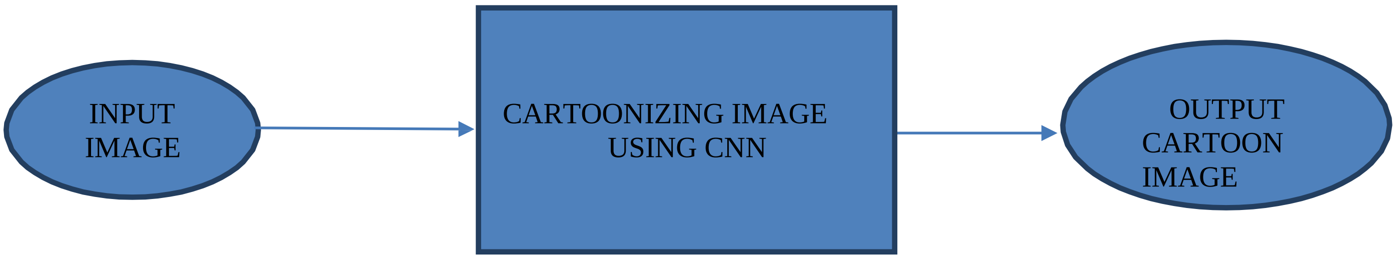
## 1.3 Proposed System

We proposed to develop a system which will help anyone to cartoonize their picture in a easiest manner. There will be no time delay to produce the cartoon image. The quality of the picture will remain same and will also get good quality

### Advantages:

1. It will provide immediate cartoon image to the user.

2. No need to compromise with the generated cartoon picture quality  this project.
3. All types of images can be cartoonized .



**1.3.1 Proposed System**

## 1.4. System Requirements

### 1.4.1 Hardware Requirements:

| | | |
|---|---|---|
| ● Processor | : | intel®core™i7-7500UCPU@2.70gh |
| ● Cache memory | : | 4MB(Megabyte) |
| ● RAM | : | 8 gigabyte (GB) |

### 1.4.2 Software Requirements:

| | | |
|---|---|---|
| ● Operating System | : | Windows 10 Home, 64 bit Operating System |
| ● Coding Language | : | Python |
| ● Python distribution | : | Anaconda, Jupyter |

# 2. LITERATURE SURVEY

## 2.1 Deep Learning

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

## 2.2 Deep Learning methods

Deep learning models are both effective and tackle problems that are too complex for
Human brain.

- **Linear function**: Rightly termed, it represents a single line which multiplies its inputs with a constant multiplier.
- **Non-Linear function:** It is further divided into three subsets: Sigmoid function, Hyperbolic tangent, Rectified Linear Unit.
- **Artificial Neural Networks** is an information processing model that is inspired by the way biological nervous systems, such as the brain, process information. In simpler terms it is simple mathematical model of the brain which is used to process nonlinear relationships between inputs and outputs in parallel like human brain does every second. It is used for variety of tasks; popular use is for classification. It has ability to learn so quickly is what makes them so powerful and useful for variety of tasks. For an artificial neural network to learn, it has to learn what it has done wrong and is doing right, this is called feedback. Neural networks learn things in exactly the same way as the brain, typically by a feedback process called back-propagation (this is sometimes shortened to "backprop").
- **Convolutional Neural Networks** is an advanced and high potential type of classic artificial neural network model. It is built for tackling higher complexity, preprocessing, and data compilation. It takes reference from order of arrangement of neurons present in visual cortex of an animal brain. The network model can help derive relevant image data in form of smaller units or chunks. Once input data is imported into convolutional model, there are four stages involved in building the CNN: Convolution, Max-Pooling, Flattering, Full Connection. The CNNs are adequate for tasks, including image recognition, image analyzing, image segmentation, video analysis, and natural language processing.

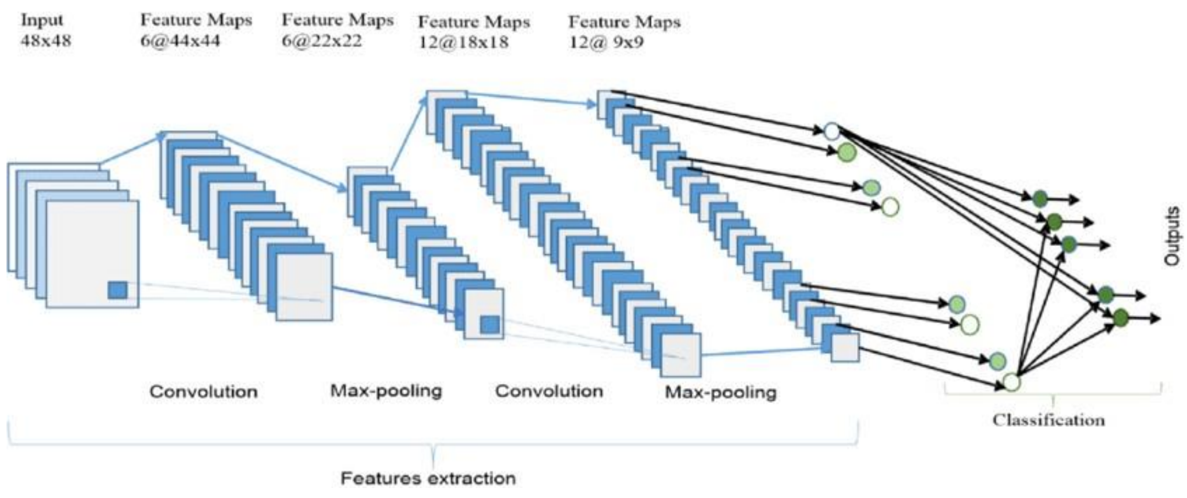


**Fig: 2.1** Convolutional Neural Network Architecture

- **Recurrent Neural Networks** were first designed to help predict sequences, for example, the long short term memory algorithm is known for its multiple functionalities. Such networks work entirely on data sequences of variable input length. There are two overall types of RNN designs that help in analyzing problems. They are LSTMs and Gated RNNs.
- **Generative Adversarial Networks** is a combination of two deep learning techniques of neural networks – a generator and a discriminator. While the generator network yields artificial data, the discriminator helps in discerning between a real and false data. It would be followed by an Image Detector network to differentiate between the real and fake images. Starting with 50% accuracy chance, the detector needs to develop its quality of classification since the generator would grow better in its artificial image generation.

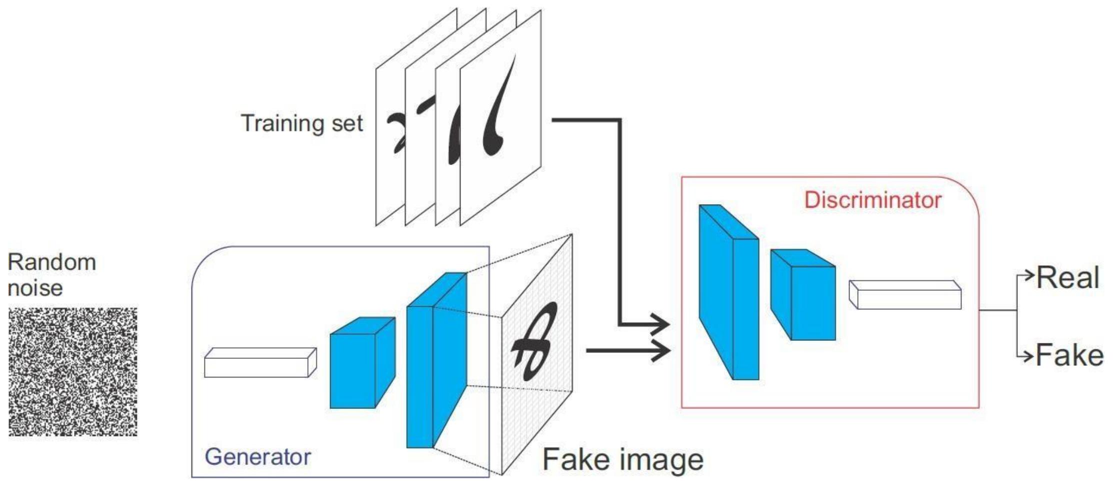


**Fig: 2.2** Generative Adversarial Network Architecture

## 2.3 Applications of Deep Learning

1. Self-Driving Cars
2. News Aggregation and Fraud Detection
3. Natural Language Processing
4. Adding sounds to silent movies
5. Language Translation
6. Pixel Restoration

## 2.4 Importance of Deep Learning in Organizations

Deep Learning's ability to learn from data on its own without any human intervention. The whole objective of Deep Learning is to solve problems with no set rules. Some of well-known companies already utilizing deep learning technologies are Apple, Google, IBM.

## 2.5 Implementation of Deep Learning using Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum. It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse, Anaconda which are particularly useful when managing larger collections of Python files.
Python was designed for its readability. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.
In the older days, people used to perform Deep Learning tasks manually by coding all the algorithms and mathematical and statistical formula. This made the process time consuming, tedious and inefficient. But in the modern days, it is become very much easy and efficient compared to the olden days by various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many

languages in the industry, one of the reason is its vast collection of libraries. Python libraries that used in Deep Learning are:

- Numpy
- Scipy
- Scikit-learn
- TensorFlow
- Keras
- PyTorch
- Pandas
- Matplotlib

**NumPy** is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Deep Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

**SciPy** is a very popular library among Deep Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

**Scikit-learn** is one of the most popular Deep Learning libraries for classical Machine Learning algorithms. It is built on top of two basic Python libraries, NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit- learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with Deep Learning.

**TensorFlow** is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

**Keras** is a very popular Deep Learning library for Python. It is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One ofthe best thing about Keras is that it allows for easy and fast prototyping.

**PyTorch** is a popular open-source Deep Learning library for Python based on Torch, which is an open-source Deep Learning library which is implemented in C with a wrapper in Lua. It hasan extensive choice of tools and libraries that supports on Computer Vision, Natural Language Processing (NLP) and many more DL programs. It allows developers to perform computations on Tensors with GPU acceleration and also helps in creating computational graphs.

**Pandas** is a popular Python library for data analysis. It is not directly related to Deep Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handyas it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping,combining and filtering data.

**Matplotlib** is a very popular Python library for data visualization. Like Pandas, it is not directlyrelated to Deep Learning. It particularly comes in handy when a programmer wants to visualizethe patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for datavisualization, histogram, error charts, bar chats, etc.

## 2.6 Deep Learning Products

Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don’t even know about it. One of the popular applications of AI is Deep Learning (DL), in which computers, software, and devices perform via cognition (very similar to human brain). Here in, we share few examples of deep learning that we use every day and perhaps haveno idea that they are driven by DL.

- Fraud Detection
- Autonomous Cars
- Virtual Assistants
- Super Computing
- Facial recognition System

## 2.7 Cartoonizing Images using Deep Learning

Previously this Cartoonizing image was developed by Lot of people. They were used Convolutional Neural Networks to achieve the cartoon images . Their applied methodology has shown significant result with an average validation accuracy of **89.63%** and an average training accuracy of **90.36%**.

# 3. SYSTEM ANALYSIS

## 3.1 Scope of the Project

This project mainly focuses on Cartoonizing the images by training thousands of pictures to get perfect cartoon image.

## 3.2 Analysis

Human face and landscape data are collected for generalization on diverse scenes. For real-world photos, we collect 10000 images from the FFHQ dataset for the human face and 5000 images from the dataset in for landscape. For cartoon images, we collect 10000 images

from animations for the human face and 10000 images for landscape. Producers of collected animations include Kyoto animation, P.A.Works, Shinkai Makoto, Hosoda Mamoru,

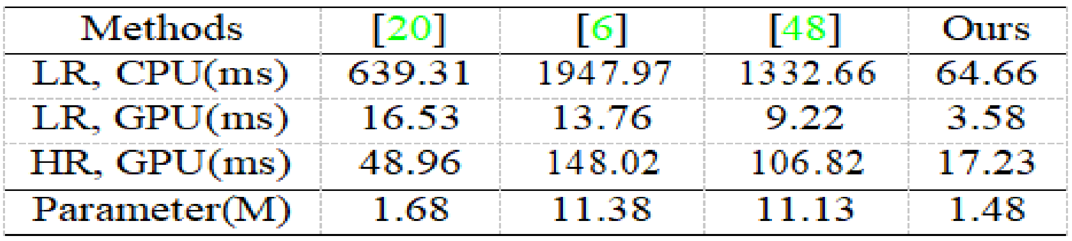
and Miyazaki Hayao. For the validation set, we collect 3011 animation images and 1978 real-world photos. Images shown in the main paper are collected from the DIV2K

dataset, and images in user study are collected from the Internet and Microsoft COCO  dataset. During training, all images are resized to 256*256 resolution, and face

images are feed only once in every fifive iterations

Previous Methods. We compare our method with four algorithms that represent Neural Style Transfer, Image to-Image Translation, Image Abstraction and Image Cartoonization respectively. Evaluation metrics.In qualitative experiments, we present results with details of four different methods and original images, as well as qualitative analysis. In quantitative experiments, we use Frechet Inception Distance (FID) to evaluate the performance by calculating the distance between source image distribution and target image distribution. In the user study, candidates are asked to rate the results of different methods between 1 to 5 in cartoon quality and overall quality. Higher scores mean better quality.Time Performance and Model Size. Speeds of four methods are compared on different hardware and shown in

Table 1. Our model is the fastest among four methods on all devices and all resolutions, and has the smallest model size. Especially, our model can process a 720*1280 image on GPU within only 17.23ms, which enables it for real-time



| Methods | [20] | [6] | [48] | Ours |
|---|---|---|---|---|
| LR, CPU(ms) | 639.31 | 1947.97 | 1332.66 | 64.66 |
| LR, GPU(ms) | 16.53 | 13.76 | 9.22 | 3.58 |
| HR, GPU(ms) | 48.96 | 148.02 | 106.82 | 17.23 |
| Parameter(M) | 1.68 | 11.38 | 11.13 | 1.48 |

Table 1: Performance and model size comparison, LR means 256*256 resolution, HR means 720*1280 resolution.

## 3.2.1 Model Comparison

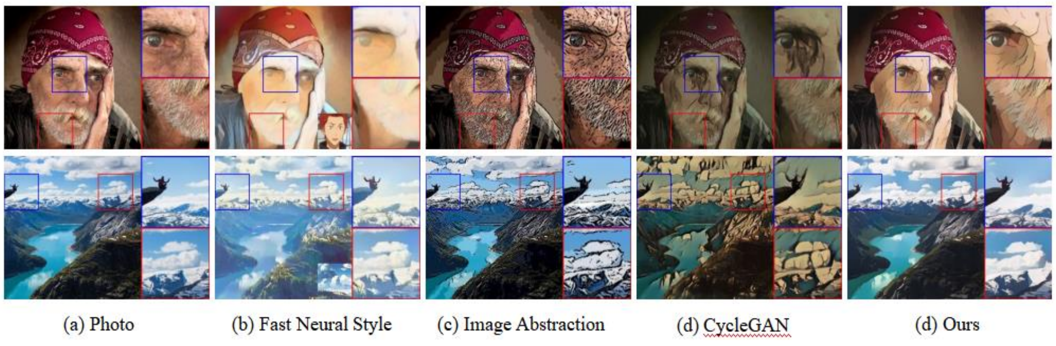


**Fig. 5 Qualitative comparison, second raw shows 4 different styles of Cartoon GAN**

| **Methods** | **Photo** | **Fast Neural Style** | **CycleGAN** | **Image Abstraction** | **Ours** |
|---|---|---|---|---|---|
| **FID to Cartoon** | 162.89 | 146.34 | 141.50 | 130.38 | 101.31 |
| **FID to Photo** | N/A | 103.48 | 122.12 | 75.28 | 28.79 |
| **Methods** | Shinkai style of | Hosoda style of | Hayao style of | Paprika style of | Ours |
| **FID to Cartoon** | 135.94 | 130.76 | 127.35 | 127.05 | 101.31 |
| **FID to Photo** | 37.96 | 58.13 | 86.48 | 118.56 | 28.79 |

TABLE 2
PERFORMANCE EVALUTION BASED ON FID

Generality to diverse use cases. We apply our model on diverse real-world scenes, including natural landscape, city views, people, animals, and plants, and show the results Validation of Cartoon Representations.

### Validation of Cartoon Representations

To validate our proposed cartoon representations reasonable and effective, a classifification experiment and a quantitative experiment based on FID are conducted, and the results are shown in Table 2. We train a binary classifier on our training dataset to distinguish between real-world photos and cartoon images. The classifier is designed by adding a fully-connected layer to the discriminator in our framework. The trained classifier is then evaluated on the validation set to validate the inflfluence of each cartoon representation.

| No. | Surface | Structure | Texture | Original |
|---|---|---|---|---|
| Acc | 0.8201 | 0.6342 | 0.8384 | 0.9481 |
| FID | 113.57 | 112.86 | 112.71 | 162.89 |

TABLE 3
CLASSIFICATION ACCURACY AND FID EVALUTION OF OUR PROPOSED CARTOON REPRESENTATION

We fifind the extracted representations successfully fool the trained classififier, as it achieves lower accuracy in all three extracted cartoon representations compared to the original images. The calculated FID metrics also support our proposal that cartoon representations help close the gap between real-world photos and cartoon images, as all three extracted cartoon representations have smaller FID compared to the original images.

boundaries, such as human face and clouds. Cartoon representations also help keep color harmonious. CycleGAN generates darkened images and Fast Neural Style causes oversmoothed color, and CartoonGAN distorts colors like human faces and ships. Our method, on the contrary, prevents improper color modififications such as faces and ships. Lastly, our method effectively reduces artifacts while preserves fifine details, such as the man sitting on the stone, but all other methods cause over-smoothed features or distortions. Also, methods like CycleGAN, image abstraction and some style of CartoonGAN cause high-frequency artifacts. To conclude, our method outperforms previous methods in generating images with harmonious color, clean boundaries, fifine details, and fewer noises.

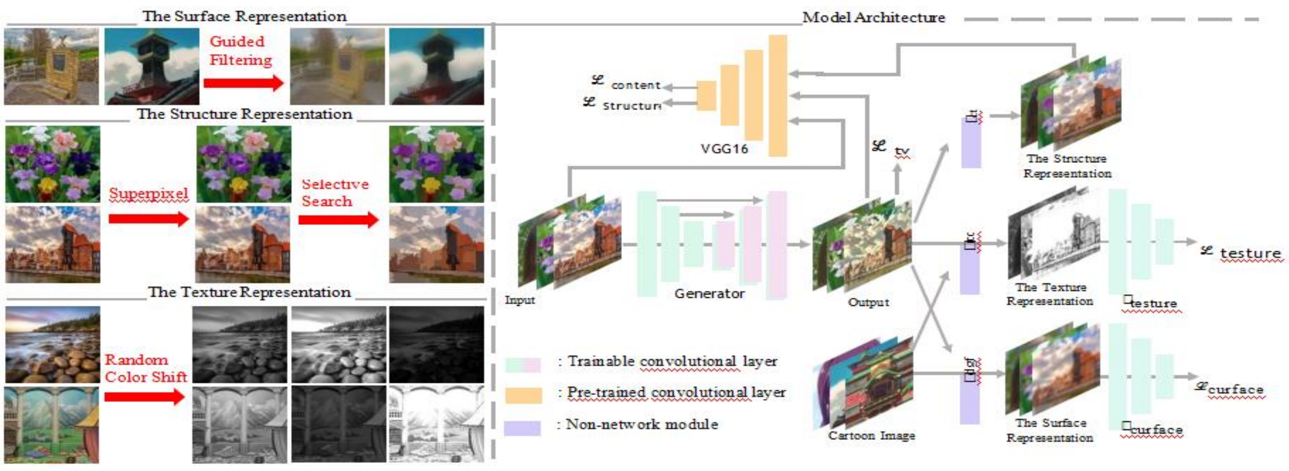

The Surface Representation
Guided Filtering
The Structure Representation
Superpixel
Selective Search
The Texture Representation
Random Color Shift
Model Architecture
content
structure
VGG16
tv
Input
Generator
Output
The Structure Representation
The Texture Representation
texture
Cartoon Image
The Surface Representation
surface
: Trainable convolutional layer
: Pre-trained convolutional layer
: Non-network module

Fig: Proposed image cartoonization system

## 3.3 Data Preprocessing

The steps used to prepare data for the proposed work has been discussed in detail below.

### 3.3.1 Data Cleaning

The collected dataset was manually filtered to remove low quality, blurred or partially obscured images. The resulting images after data cleaning process were 39209 in number.

### 3.3.2 Resizing of Images

The resulting images after data cleaning were uniformly resized to 3030 pixels before training the model effectively. This was done because sequential models perform better on square images of similar sizes.

## 3.4 Classification

Classification is supervised learning and can be performed on both structured and unstructured data. It categorizes a set of data into classes and predicts the class of given data points. CNN are used to classify the data and predict them correctly.

**CNN:** Convolutional neural networks have proved to be efficient when using matrices to deliver good images. CNN consists of a pipeline of convolutionary layers which are pooled and completely connected. Convolution layer extracts key features from an image input and forms a map. The spatial relationship of the image pixels is preserved through this feature map. Input images with various kernels (filters) lead to the learning of trivial features such as edges and complex features such as patterns. Pooling Layer reduces the size of convolutionary characteristics maps by extracting only dominant features. The linear rectified (ReLU) unit is used for introduction of CNN non-linearity. CNN finally has one or more fully connected layers that are used to input flattened matrix values. In applications such as image classification, functional and object detection, CNN has been successfully applied in images, enabling them to form the basis for computer vision applications.

The model used in this project is a sequential model. The architecture of the CNN model the proposed work has been discussed as follows:

1. Conv2D – A convolution is used for feature extraction. It extracts the different features at different layers with the help of the filters. The first conv2d layer uses 32 filters and the kernel size of 5 x 5 and ReLU activation function. The output of ReLU is max(0,x). Then another identical Conv2D layer is applied again.
2. MaxPool2D – Max pooling is used to reduce the dimensions of the image. This is helpful in reducing the computation power required to process the image. The model uses a pool size of (2,2) which means it will select the maximum value of each 2 x 2 area and the dimensions of the image will reduce by a factor of 2.
3. Dropout – Next we insert dropout layer. It is used to prevent overfitting and reduces the complexity of the model. The model uses dropout rate as 0.25 which means 25% of neurons are removed randomly.
4. Conv2D, MaxPool2D, Dropout – In the next step these three layers are applied again for better training and performance of our model. This time the model uses 64 filters instead of 32 and a kernel size of 3 x 3 instead of 5 x 5, as used in previous steps.
5. Flatten- The flatten layer transforms the 2-D data into a long 1-D vector of features for a fully connected layer that can be fed into the neural network.

6. Dense – We use dense layer as a output layer. The last layer of a network is dense layer. It outputs 43 nodes as the traffic signs have been divided into 43 categories in the dataset. The last layer uses softmax activation function which gives the probability value (between 0 and 1) so that the model can predict which of the 43 options has highest probability.

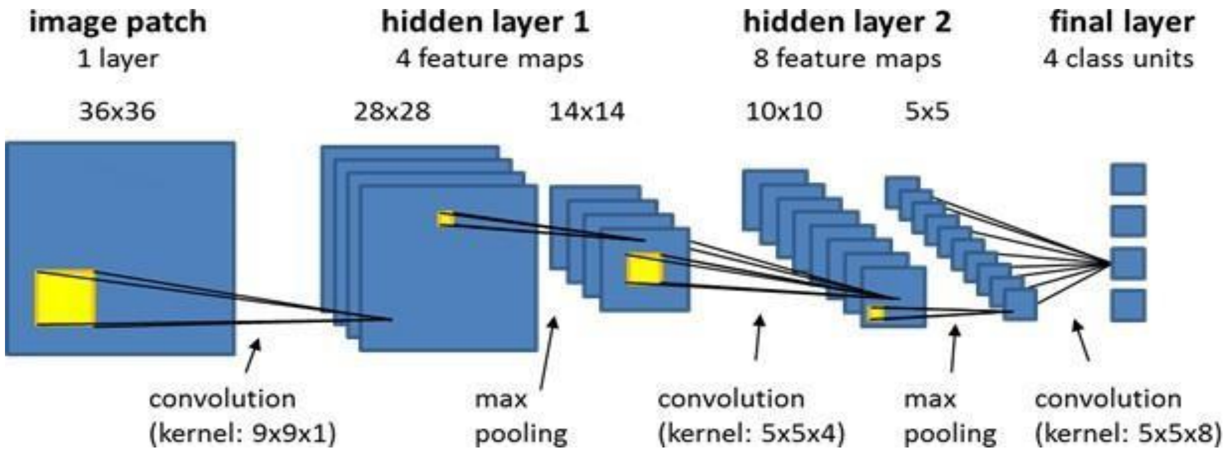


**Fig: 3.3.1** CNN Architecture

**GAN:** Generative Adversarial Network (GAN) is a state-of-the-art generative model that can generate data with the same distribution of input data by solving a min-max problem between a generator network and a discriminator net- work. It is powerful in image synthesis by forcing the gener- ated images to be indistinguishable from real images. GAN has been widely used in conditional image generation tasks, such as image in painting, style transfer, image car- toonization, image colorization. In our method, we adopt adversarial training architecture and use two discrim- inators to enforce the generator network to synthesize im- ages with the same distribution as the target domain.

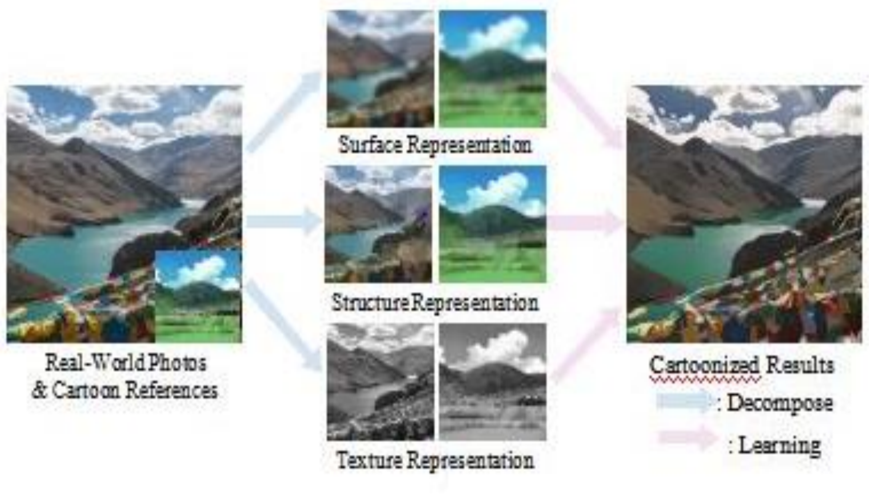


Fig: White box representation

# 4. IMPLEMENTATION CODE

## 4.1 Code for training the model:

```
from flask import Flask, render_template, request
import os
import cv2
import numpy as np
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import network
import guided_filter
from tqdm import tqdm


app = Flask(__name__)

app.config['UPLOAD_PATH'] = 'static\\normal_images'

@app.route('/')
def home():
    return render_template("index.html")

def resize_crop(image):
    h, w, c = np.shape(image)
    if min(h, w) > 720:
        if h > w:
            h, w = int(720*h/w), 720
        else:
            h, w = 720, int(720*w/h)
    image = cv2.resize(image, (w, h),
                    interpolation=cv2.INTER_AREA)
    h, w = (h//8)*8, (w//8)*8
    image = image[:h, :w, :]
    return image
```

```
@app.route('/cartoonize', methods = ['GET', 'POST'])
def cartoonize():
    if request.method == 'POST':
      f = request.files['ifile']
    if f.filename != '':
        f.save(os.path.join(app.config['UPLOAD_PATH'], f.filename))
    print("---------------------------------------")
    print(f)
    print(type(f))
    model_path = 'saved_models'
    load_folder = 'static\\normal_images'
    save_folder = 'static\\cartoonized_images'
    if not os.path.exists(save_folder):
        os.mkdir(save_folder)
    input_photo = tf.placeholder(tf.float32, [1, None, None, 3])
    network_out = network.unet_generator(input_photo)

    final_out = guided_filter.guided_filter(input_photo, network_out, r=1, eps=5e-3)

    all_vars = tf.trainable_variables()
    gene_vars = [var for var in all_vars if 'generator' in var.name]
    saver = tf.train.Saver(var_list=gene_vars)

    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    sess = tf.Session(config=config)

    sess.run(tf.global_variables_initializer())
    saver.restore(sess, tf.train.latest_checkpoint(model_path))
    name_list = os.listdir(load_folder)
    print("...............................................")
    print(name_list)
    for name in name_list:
        if name == f.filename:
            img = name
            break
    try:
        load_path = os.path.join(load_folder, img)
        save_path = os.path.join(save_folder, img)
        print(save_path)
        image = cv2.imread(load_path)
        image = resize_crop(image)
        batch_image = image.astype(np.float32)/127.5 - 1
        batch_image = np.expand_dims(batch_image, axis=0)
        output = sess.run(final_out, feed_dict={input_photo: batch_image})
        output = (np.squeeze(output)+1)*127.5
        output = np.clip(output, 0, 255).astype(np.uint8)
        cv2.imwrite(save_path, output)
    except:
        print('cartoonize {} failed.......'.format(img))
    return render_template('index.html', img_name = img)



if __name__ == '__main__':
    app.run(debug = True)
```

## 4.2 Code for filters:

```
import tensorflow as tf
import numpy as np
def tf_box_filter(x, r):
    k_size = int(2*r+1)
    ch = x.get_shape().as_list()[-1]
    weight = 1/(k_size**2)
    box_kernel = weight*np.ones((k_size, k_size, ch, 1))
    box_kernel = np.array(box_kernel).astype(np.float32)
    output = tf.nn.depthwise_conv2d(x, box_kernel, [1, 1, 1, 1], 'SAME')
    return output
def guided_filter(x, y, r, eps=1e-2):

    x_shape = tf.shape(x)
    #y_shape = tf.shape(y)
    N = tf_box_filter(tf.ones((1, x_shape[1], x_shape[2], 1), dtype=x.dtype), r)

    mean_x = tf_box_filter(x, r) / N
    mean_y = tf_box_filter(y, r) / N
    cov_xy = tf_box_filter(x * y, r) / N - mean_x * mean_y
    var_x  = tf_box_filter(x * x, r) / N - mean_x * mean_x

    A = cov_xy / (var_x + eps)
    b = mean_y - A * mean_x

    mean_A = tf_box_filter(A, r) / N
    mean_b = tf_box_filter(b, r) / N

    output = mean_A * x + mean_b

    return output
def fast_guided_filter(lr_x, lr_y, hr_x, r=1, eps=1e-8):

    #assert lr_x.shape.ndims == 4 and lr_y.shape.ndims == 4 and hr_x.shape.ndims == 4

    lr_x_shape = tf.shape(lr_x)
    #lr_y_shape = tf.shape(lr_y)
    hr_x_shape = tf.shape(hr_x)

    N = tf_box_filter(tf.ones((1, lr_x_shape[1], lr_x_shape[2], 1), dtype=lr_x.dtype), r)

    mean_x = tf_box_filter(lr_x, r) / N
    mean_y = tf_box_filter(lr_y, r) / N
    cov_xy = tf_box_filter(lr_x * lr_y, r) / N - mean_x * mean_y
    var_x  = tf_box_filter(lr_x * lr_x, r) / N - mean_x * mean_x

    A = cov_xy / (var_x + eps)
    b = mean_y - A * mean_x

    mean_A = tf.image.resize_images(A, hr_x_shape[1: 3])
    mean_b = tf.image.resize_images(b, hr_x_shape[1: 3])
    output = mean_A * hr_x + mean_b

    return output
if __name__ == '__main__':
    import cv2
    from tqdm import tqdm

    input_photo = tf.placeholder(tf.float32, [1, None, None, 3])
    #input_superpixel = tf.placeholder(tf.float32, [16, 256, 256, 3])
    output = guided_filter(input_photo, input_photo, 5, eps=1)
    image = cv2.imread('output_figure1/cartoon2.jpg')
    image = image/127.5 - 1
    image = np.expand_dims(image, axis=0)

    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    sess = tf.Session(config=config)
    sess.run(tf.global_variables_initializer())

    out = sess.run(output, feed_dict={input_photo: image})
    out = (np.squeeze(out)+1)*127.5
    out = np.clip(out, 0, 255).astype(np.uint8)
    cv2.imwrite('output_figure1/cartoon2_filter.jpg', out)
```

## **Code for Network:**

```
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import numpy as np
import tf_slim as slim

def resblock(inputs, out_channel=32, name='resblock'):



    with tf.variable_scope(name):
        x = slim.convolution2d(inputs, out_channel, [3, 3],
                               activation_fn=None, scope='conv1')
        x = tf.nn.leaky_relu(x)
        x = slim.convolution2d(x, out_channel, [3, 3],
                               activation_fn=None, scope='conv2')

        return x + inputs

def unet_generator(inputs, channel=32, num_blocks=4, name='generator', reuse=False):
    with tf.variable_scope(name, reuse=reuse):

        x0 = slim.convolution2d(inputs, channel, [7, 7], activation_fn=None)
        x0 = tf.nn.leaky_relu(x0)

        x1 = slim.convolution2d(x0, channel, [3, 3], stride=2, activation_fn=None)
        x1 = tf.nn.leaky_relu(x1)
        x1 = slim.convolution2d(x1, channel*2, [3, 3], activation_fn=None)
        x1 = tf.nn.leaky_relu(x1)

        x2 = slim.convolution2d(x1, channel*2, [3, 3], stride=2, activation_fn=None)
        x2 = tf.nn.leaky_relu(x2)
        x2 = slim.convolution2d(x2, channel*4, [3, 3], activation_fn=None)
        x2 = tf.nn.leaky_relu(x2)

        for idx in range(num_blocks):
            x2 = resblock(x2, out_channel=channel*4, name='block_{}'.format(idx))

        x2 = slim.convolution2d(x2, channel*2, [3, 3], activation_fn=None)
        x2 = tf.nn.leaky_relu(x2)

        h1, w1 = tf.shape(x2)[1], tf.shape(x2)[2]

        x3 = tf.image.resize_bilinear(x2, (h1*2, w1*2))
        x3 = slim.convolution2d(x3+x1, channel*2, [3, 3], activation_fn=None)
        x3 = tf.nn.leaky_relu(x3)
        x3 = slim.convolution2d(x3, channel, [3, 3], activation_fn=None)
        x3 = tf.nn.leaky_relu(x3)

        h2, w2 = tf.shape(x3)[1], tf.shape(x3)[2]
        x4 = tf.image.resize_bilinear(x3, (h2*2, w2*2))
        x4 = slim.convolution2d(x4+x0, channel, [3, 3], activation_fn=None)
        x4 = tf.nn.leaky_relu(x4)
        x4 = slim.convolution2d(x4, 3, [7, 7], activation_fn=None)

        return x4

if __name__ == '__main__':

    pass
```

## Accuracy output:

```
Accuracy is: 89.93982581155978
```

## 4.3 Code for UI:

**Base.html:**

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">



<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>

{% block head %}
{% endblock %}

</head>

<body>

{% block body %}

{% endblock %}

</body>

</html>
```

**Index.html:**

```
{% extends 'base.html' %}
{% block head %}
  <title>Home</title>
  <style>
body {
        background-
image:url("C:\\Users\\bvenk\\Desktop\\deeeeeep\\test_code\\templates\\mountain.jpg");
                background-repeat:no-repeat;
                background-size:cover;

        }
        .button
        {
                padding:3px;
                border-radius:15%;
                cursor:pointer;
        }


        .button:hover
        {
                background-color:#d699ff;
                color:white;
        }
        .box {
   width:600px;
   text-align:center; /* center align the text inside the box */
   margin:auto; /* center this .box element, assuming it is block-level */
        }
```

```
            </style>

            <script>

                        var loadFile = function(event) {

                        var image = document.getElementById('outpt');

                        image.src = URL.createObjectURL(event.target.files[0]);

                        };

            </script>

{% endblock %}

{% block body %}


            <div class="">

<center><h1><b>Cartoonize your picture here<b><h1></center>
<center>
<form name="" method="POST" action="http://localhost:5000/cartoonize" enctype = "multipart/form-
data">
<input type="file" accept="image/*" name="ifile" id="file" onchange="loadFile(event)"/>
<input type="submit" class="button" value="Cartoonize"/>
</form>
</center>
<center>
            <div class="">
                        <img id="outpt" width="200" />
            </div>
            </center>

            {% if img_name %}
            <center>
            <div class="row">
      <div class="card jumbotron col-3 mx-4 p-1">
        <div class="card-body">
                                    {% set t = 'normal_images/'~img_name %}
                                    <img src="{{url_for('static', filename = t)}}" width="200" height="300">
                            </div>
              </div>
                    <div class="card jumbotron col-3 mx-4 p-1">
               <div class="card-body">
                                    {% set c = 'cartoonized_images/'~img_name %}
                                    <img src="{{url_for('static', filename = c)}}" width="200" height="300">
                            </div>
              </div>
            </div>
                  </center>
                  {% endif %}
        {%endblock%}
```

# 5. RESULT ANALYSIS

In this project, the model created was used to classify the traffic signs. Various parameters used to train the model have been listed out in below table

| No. | Surface | Structure | Texture | Original |
|---|---|---|---|---|
| Acc | 0.8201 | 0.6342 | 0.8384 | 0.9481 |
| FID | 113.57 | 112.86 | 112.71 | 162.89 |

**Table: 5.1** Parameters used for the model

# 6. SCREEN SHOTS

## HOME PAGE:

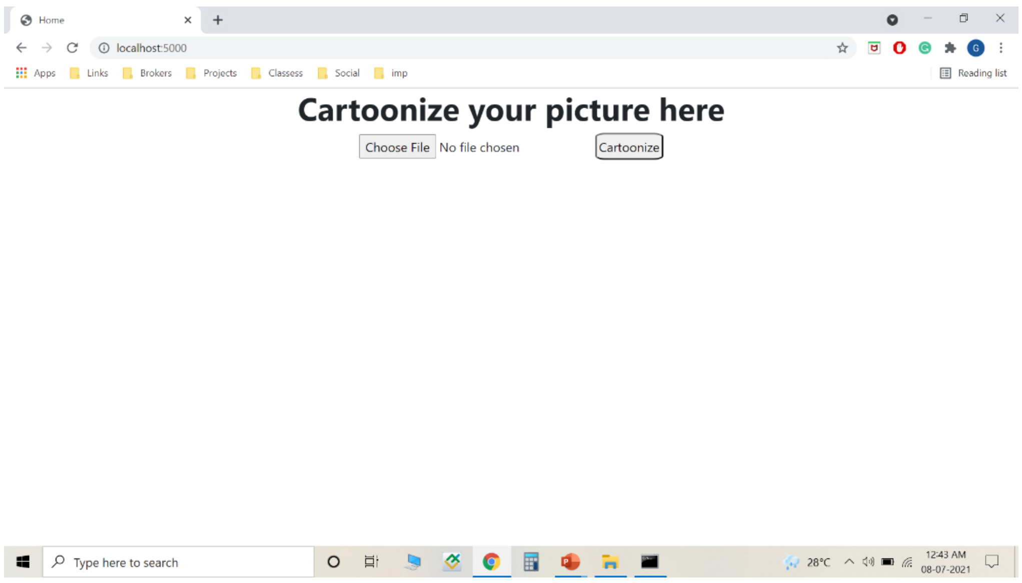


**Fig:6.1** Output screen for Home page

## IMAGE UPLOADING PAGE:

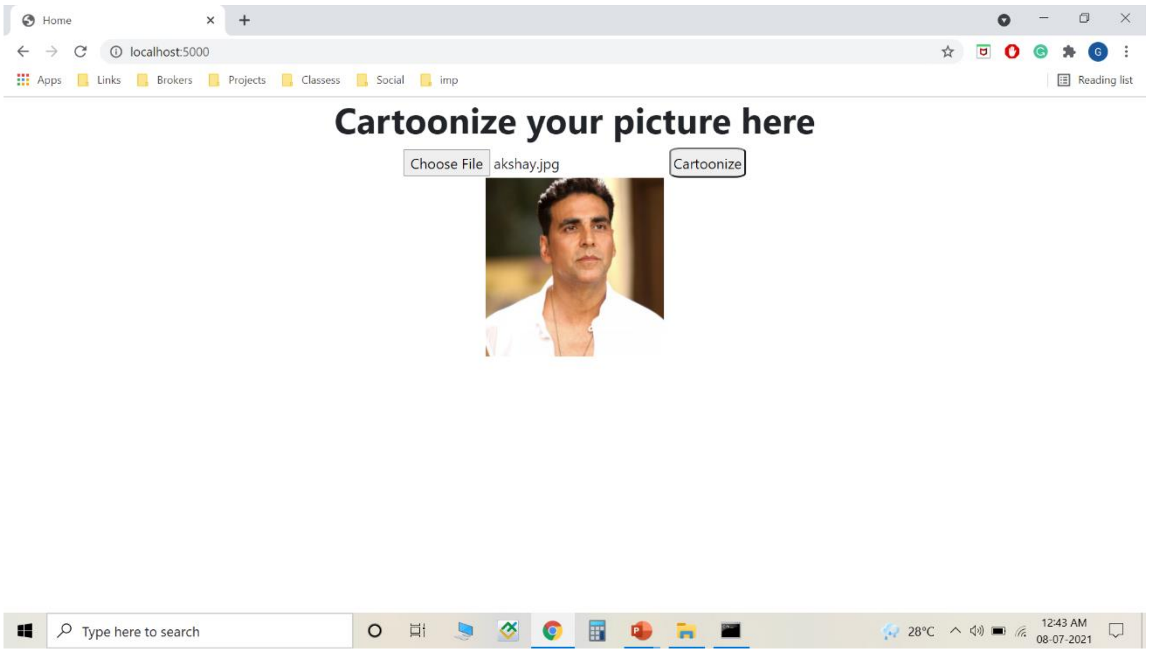


**Fig:6.2** Output screen for Cartoon Image uploading Page

## CLASSIFING IMAGE PAGE:

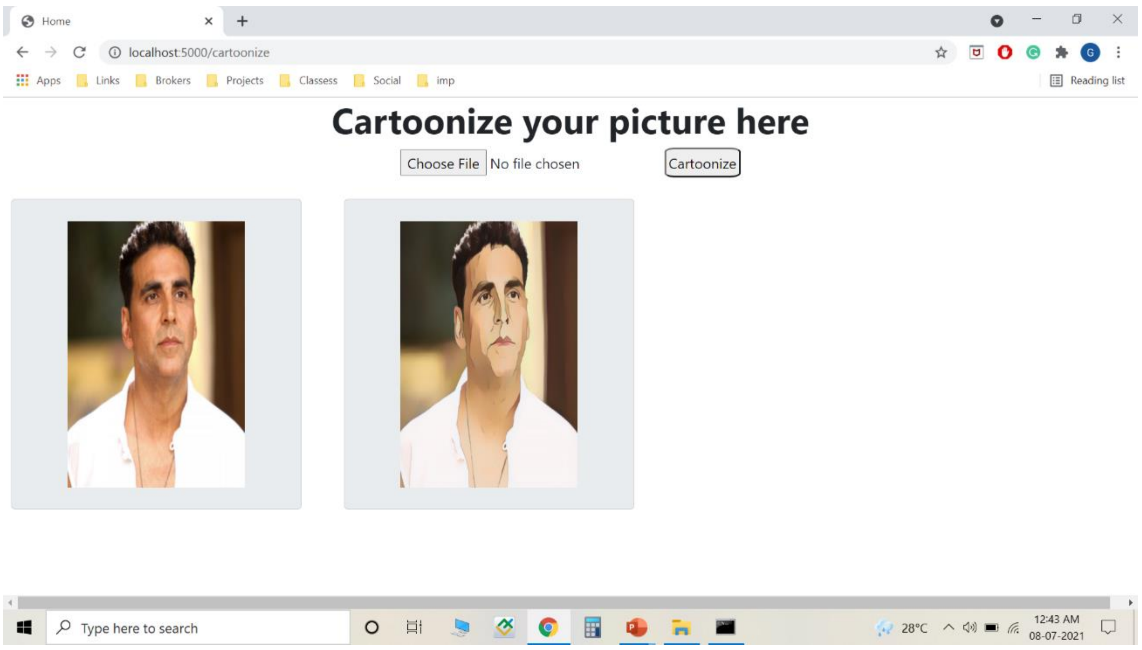


**Fig:6.3** Output screen page

# 7. FUTURE ENHANCEMENT

- We will upgrade this project to cartoonize videos
- This Project will also be implemented in mobile phone to capture live cartoon images or videos

## 8. CONCLUSION

- Finally, the user can be able to cartoonize any picture easily.

- There will be no issue with the cartoonized image quality and no delay in producing the cartoonpicture

## 9. REFERENCES

- https://towardsai.net/p/deep-learning/an-insiders-guide-to-cartoonization-using-machine-learning

- https://www.youtube.com/watch?v=SJe0CCAvilo&ab_channel=UnfoldDataScience

- https://www.bilibili.com/video/av56708333

- https://github.com/SystemErrorWang