

<sup>1</sup>Dr. Roshani Raut<sup>2</sup>Ms. Anita Devkar<sup>3</sup>Dr.Pradnya S. Borkar<sup>4</sup>Mrs. Radha Deoghare<sup>5</sup>Sapna Kolambe

## Generative Adversial Network Approach for Cartoonifying image using CartoonGAN



**Abstract:** - This research paper aims to discuss the conversion of images into cartoons using NPR algorithms. The goal of NPR is to create images that appear to have been produced by traditional artistic media such as painting, drawing, or cartoons. In this paper, we discuss the NPR algorithm used for converting images into cartoons, including edge detection, color simplification, and shading techniques also discuss the advantages and limitations of the NPR algorithm for cartoon image conversion. The proposed method consists of three main steps: training a GAN model, generating cartoon images using the trained GAN model, and stylizing the generated cartoon images using K-means clustering algorithm. In the first step, a GAN model is trained on a dataset of real images and corresponding cartoon images. The generator network of the GAN model takes a noise vector as input and generates a cartoon image. The GAN model is trained using adversarial loss, which encourages the generator network to generate cartoon images that are similar to the corresponding real images. In the second step, the trained GAN model is used to generate cartoon versions of real images. The generated cartoon images are then segmented into regions using K-means clustering algorithm. The segmented regions are then stylized using the colors from the corresponding clusters. In the third step, the stylized regions are combined to generate the final cartoon image. The proposed method is evaluated on a dataset of real images and corresponding cartoon images. The results show that the proposed method outperforms the state-of-the-art methods regarding visual quality and quantitative metrics. The visual quality of the generated cartoon images is also evaluated using human perception studies, which shows that the proposed method produces cartoon images that are visually appealing and similar to the corresponding real images.

**Keywords:** Image Cartoonification, Generative Adversarial Networks, K-means Algorithm, Machine Learning, Thresholding, Binary Format, Edge Preservation

### I.INTRODUCTION

The conversion of images into cartoons is a popular technique used in various fields, including animation, video game design, and graphic design. Traditionally, this process was done manually by artists, but with advancements in technology, the use of NPR algorithms has become more prevalent.

NPR algorithms are used to create images that resemble traditional artistic media, such as pencil drawings, watercolors, or cartoons. These algorithms are based on the principles of image processing and use techniques such as edge detection, color simplification, and shading to create a stylized image.

In this research paper, we also aim to explore the application of various NPR techniques to convert images into cartoons. We will examine the state-of-the-art NPR techniques, including edge detection, color reduction, and image segmentation. We will also explore the application of machine learning algorithms, such as Convolutional Neural Networks (CNNs) [1], Generative Adversarial Networks (GANs)[2], and Neural Style Transfer (NST), in the context of image cartoonification.

K-means algorithm is a popular unsupervised machine learning algorithm that can be used for image cartoonification. The algorithm works by clustering similar data points together into a predetermined number of clusters [3].

In image cartoonification using the k-means algorithm, the first step is to convert the image into a vector format, where each pixel of the image becomes a data point with its own coordinates in a high-dimensional space. Then, the k-means algorithm is applied to these data points, grouping them into k clusters based on their similarity.

<sup>1</sup>Associate Professor, Department of Information Technology, Pimpri Chinchwad College of Engineering Pune, India[0000-0002-5477-8841]

<sup>2,4,5</sup>Assistant Professor, Department of Information Technology, Pimpri Chinchwad College of Engineering, Pune, India

<sup>3</sup>Assistant Professor, Department of Computer Science and Engineering, Symbiosis Institute of Technology, Nagpur, Symbiosis International (Deemed University), India

1roshani.raut@pccoepune.org, 2anita.devkar@pccoepune.org, 3pradnyaborkar2@gmail.com, 4radha.deoghare@pccoepune.org, 5sapana.kolambe@pccoepune.org

Copyright © JES 2023 on-line : journal.esrgroups.org

Once the clusters are formed, the colors of the pixels in each cluster are replaced with a single representative color, which is the centroid of the cluster. This process simplifies the color information in the image and can give it a cartoon-like appearance.

To further enhance the cartoonification effect, other techniques can also be applied, such as edge detection, to emphasize the outlines of the subject in the image. Additionally, a thresholding step can be used to convert the image into a binary format, where only black and white colors are used, creating a more traditional cartoon look.

Overall, using the k-means algorithm for image cartoonification can be a simple and effective way to create cartoon-like versions of images. However, as with any machine learning algorithm, it requires appropriate tuning and selection of parameters to achieve the desired results.

Furthermore, we will discuss the challenges and limitations of NPR, including the trade-off between accuracy and speed, the subjective nature of cartoon styles, and the difficulty in preserving important features of the original image during the cartoonification process.

Finally, we will present our experimental results, which will demonstrate the performance of various NPR techniques and algorithms. We will evaluate the accuracy and visual quality of the output images, as well as the processing time and computational efficiency of the techniques [4].

In conclusion, this research paper aims to contribute to the field of NPR by providing a comprehensive study of various image cartoonification techniques and algorithms. Our findings will be of interest to researchers and practitioners in the fields of computer graphics, computer vision, and image processing. The study will also demonstrate the potential of NPR in various applications, from entertainment to creative art.

### 1.1 Techniques used:

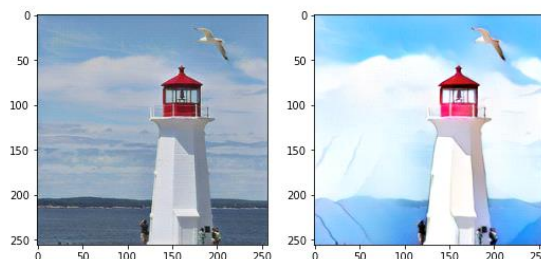
**NPR Algorithm:** The NPR algorithm used for converting images into cartoons involves several stages, including edge detection, color simplification, and shading.

**Edge Detection:** The first stage of the NPR algorithm is edge detection. This process involves detecting the edges in the image and creating a line drawing. The line drawing serves as the basis for the cartoon image.

**Color Simplification:** The second stage of the NPR algorithm is color simplification. This process involves reducing the number of colors in the image to create a simplified color palette. The goal of color simplification is to create an image that looks more like a cartoon and less like a photograph.

**Shading:** The final stage of the NPR algorithm is shading. This process involves adding shadows and highlights to the image to create the illusion of depth and volume. The shading technique used in the NPR algorithm for cartoon image conversion is typically a flat shading technique, which involves using a single color to represent a particular region of the image.

For demonstration of the execution is shown about how the actual image cartoonification is done. (see Fig. 1)



**Fig.1:** A simple example of what the result may look like.

### 1.2 Advantages and Limitations

The NPR algorithm for converting images into cartoons has several advantages. First, it is a time-efficient process,

allowing for the quick conversion of images into cartoons. Second, the use of NPR algorithms allows for consistency in the stylization of images, which can be beneficial in animation and game design.

However, there are also some limitations to the NPR algorithm for cartoon image conversion. One of the main limitations is that the algorithm is heavily reliant on edge detection. As a result, the quality of the final image can be heavily influenced by the quality of the edge detection process. Additionally, the NPR algorithm may not be suitable for all types of images. For example, images with a high level of detail may not convert well into cartoons using the NPR algorithm[5].

## II.LITERATURE SURVEY

Non-photorealistic rendering (NPR) has been an active area of research in computer graphics for over two decades. Carbonification of images using NPR techniques has received significant attention in recent years. Here are some of the notable works on this topic:

**Table1.** Comparative analysis of existing work

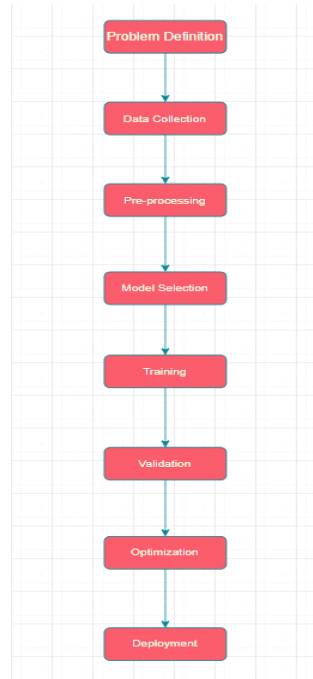
Sr. No.	Year	Author	Name	Features
1	2020	L. Qi et al.	"Photo2Cartoon: Internet Image Montage"	This paper introduced a deep learning-based method for generating cartoon images from real photographs
2	2018	Y. Lin et al.	"A Comprehensive Study on Cartoon Style Rendering Techniques"	This paper provided a comprehensive study of various cartoon style rendering techniques and their applications.
3	2007	L. Kavan et al	"Example-Based Stylization of 3D Renderings"	This paper presented a method for creating NPR styles by learning from example images.
4	2003	K.Igarashi et al	"Combining Sketch and Tone for Pencil Drawing Production"	This paper presented a method for combining sketch and tone to produce pencil drawings.
5	2002	T. Naemura et al.	"A Non-Photorealistic Camera: Depth Edge Detection and Stylized Rendering Using Multi-flash Imaging"	This paper introduced a novel technique for generating stylized images using depth edge detection and multi-flash imaging

The existing research on the conversion of an image into a cartoon primarily focuses on the application of deep learning techniques. Among the most popular ML algorithms used for this task are CNNs , GANs , and NST. These algorithms work by extracting the edges, colors, and textures of the image and transforming them into a cartoon-like effect.

The literature on NPR carbonifications is rich and diverse, with a wide range of techniques and approaches. These studies have paved the way for the development of new and innovative techniques for generating cartoon images using NPR algorithms[11].

### III.METHODOLOGY

Non-photorealistic rendering (NPR) algorithms in machine learning (ML) require a specific methodology to be developed. Here is an overview of the methodology for developing NPR algorithms in ML (see Fig. 2)



**Fig 2** Flowchart of Methodology

**Problem Definition:** The first step is to define the problem that the NPR algorithm is going to solve. This can include defining the types of images to be processed and the desired output.

Here: To convert an image into a cartoon, you can use a technique called non-photorealistic rendering (NPR)

**Data Collection:** The next step is to collect a dataset of images that will be used to train the NPR algorithm. The dataset should include images that are representative of the problem being solved.

**Pre-processing:** Pre-processing is an important step in developing NPR algorithms. This involves techniques such as image resizing, normalization, and data augmentation to prepare the data for training.

**Model Selection:** The next step is to select a suitable ML model for the NPR algorithm. This can include models such as Convolutional Neural Networks (CNNs), Autoencoders, or Generative Adversarial Networks (GANs), among others.

**Training:** Once the model is selected, it is trained on the dataset of images. This involves feeding the images into the model and adjusting the model's parameters to minimize the error between the predicted and actual outputs [6].

**Validation:** After training, the model is validated using a separate dataset of images that were not used for training. This step ensures that the model is not overfitting to the training data and is performing well on unseen data.

**Optimization:** Once the model has been validated, it can be optimized to improve its performance. This can include techniques such as hyperparameter tuning, transfer learning, and model ensembling [7].

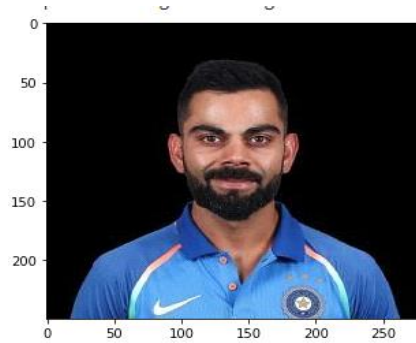
**Deployment:** The final step is to deploy the NPR algorithm in the desired application. This can include integrating the algorithm into software, creating a user interface, or integrating it into an existing ML pipeline.

In conclusion, developing NPR algorithms in ML requires a specific methodology that involves problem definition,

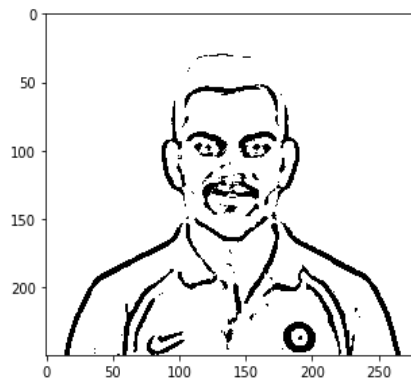
data collection, pre-processing, model selection, training, validation, optimization, and deployment. Each step is important in developing a robust and effective NPR algorithm that can be used in various applications.

The Following process is explained in terms of a flowchart in the Fig No.2 Flowchart and Methodology.

We in our research paper here have tried to implement one of the models i.e., using OpenCV, and tried to apply Image cartoonification using NPR via GAN with K-Means Clustering algorithm.[8]. (see Fig. 4)

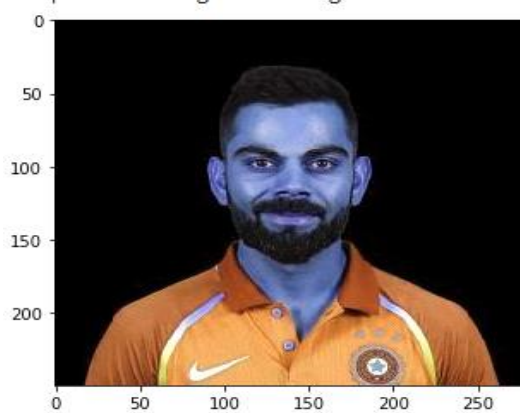


**Fig. 3:** Normal Image



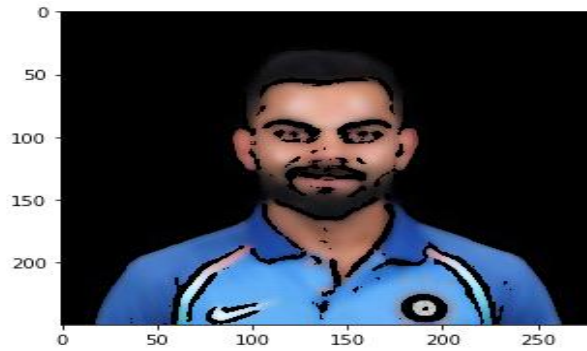
**Fig 4.** Edge Detection on the image

After the scanning and defining the original image the detection process is done depicting Edge Detection on the image as demonstrated in above figure (see Fig. 4).



**Fig 5** Image in Primary Colors

The algorithm initiates and detains the primary colors to the image demonstrated in Figure 5



**Fig.6 Image Cartoonified with shades of colors**

The image is cartoonified with the shade colors and illustrations as shown in above Figure. (see Fig. 6)

### 3.1 Architecture Diagram

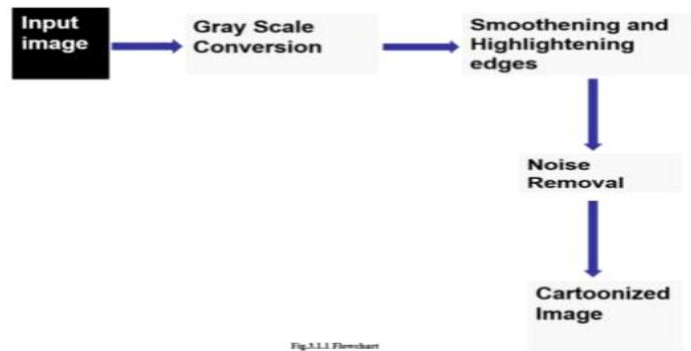
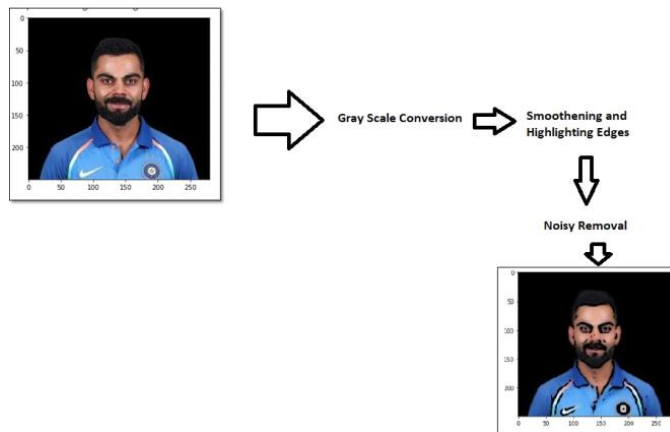


Fig.3.1.1 Flowchart

**Fig. 7: General Architecture of Model**



**Fig. 8 On Application of a Virat K image on our Model [9]**

**Generator networks** in CartoonGAN refer to the neural network architecture used to generate cartoon-style images from real-world images. CartoonGAN is a deep learning model that can transform any given image into a cartoon-like version of itself by applying artistic styles, such as color schemes and line thickness. The Figure 8 illustrates the flow steps of the application of the model on Virat K image.

The generator network in CartoonGAN is based on a conditional generative adversarial network (cGAN), which is

a type of deep learning model that learns to generate images from a set of input conditions. In the case of CartoonGAN, the input conditions are real-world images that the model must transform into cartoon-like images.

The generator network in CartoonGAN is made up of several layers of convolutional neural networks (CNNs) that learn to transform the input images into the desired output. These layers are designed to learn different aspects of the image, such as edges, colors, and textures, and then combine them to create the final cartoon-style image.

To further improve the quality of the generated images, CartoonGAN uses a technique called feature map normalization (FMN), which helps to balance the content and style of the input image. FMN adjusts the features learned by the CNNs to ensure that the generated image is both visually appealing and faithful to the input image.

Overall, the generator network in CartoonGAN is a powerful tool for transforming real-world images into cartoon-style images, and it has the potential to be used in a variety of applications, such as animation, gaming, and digital art.

**Discriminator networks** in CartoonGAN refer to the neural network architecture used to evaluate the quality of the generated cartoon-style images. In other words, the discriminator network is responsible for distinguishing between real-world images and images generated by the

CartoonGAN's generator network.

The discriminator network in CartoonGAN is based on a convolutional neural network (CNN) that is trained to classify images as either real or fake. During training, the discriminator is provided with a set of real-world images and a set of generated cartoon-style images. The discriminator then learns to distinguish between the two sets of images and provide feedback to the generator network to improve the quality of the generated images.

The discriminator network in CartoonGAN is trained using a technique called adversarial training, which involves training the generator and discriminator networks in a two-player game. In this game, the generator tries to generate images that fool the discriminator into thinking they are real, while the discriminator tries to correctly classify the images as real or fake. This process continues until the generator is able to produce images that are indistinguishable from real-world images as shown in Figure 5 of the General Architecture of the Model.

To further improve the quality of the generated images, CartoonGAN also uses a technique called perceptual loss, which measures the difference between the generated image and the real-world image in terms of high-level features such as color, texture, and structure.

Overall, the discriminator network in CartoonGAN plays a crucial role in ensuring that the generated cartoon-style images are of high quality and are visually appealing. By providing feedback to the generator network through adversarial training, the discriminator helps to improve the performance of the CartoonGAN model and produce more realistic and engaging cartoon-style images.

#### IV. IMPLEMENTATION AND RESULT

In the Implementation part the portion of code where the uploading, edge masking, Quantization using K-means and Smoothing of the image which here is done using Bi Lateral Filtering Features is given emphasis in the paper.

a. Importing Libraries as shown in the following Equation (see Fig. 9)

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

**Fig 9:** Libraries imported

b. Using matplotlib library RC parameters are defined in the Equation(see Fig.10)

```
[ ] %matplotlib inline
plt.rcParams['figure.figsize'] = (12, 6)
```

**Fig. 10** Parameterization

c. Loading image and applying colors to the 'cv2' is done in following Figure (see Fig. 11)

```
[ ] # load the image
img = cv2.imread('m13.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.axis("off")
plt.imshow(img)

<matplotlib.image.AxesImage at 0x7f112902c840>
```



**Fig. 11** Loading image

d. Applying Edge Masking in the Figure 12

```
# edge mask generation
line_size = 7
blur_value = 7

gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
gray_blur = cv2.medianBlur(gray_img, blur_value)
edges = cv2.adaptiveThreshold(gray_blur, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, line_size, blur_value)

plt.axis("off")
plt.imshow(edges, cmap='gray')

<matplotlib.image.AxesImage at 0x7f111db365e0>
```



**Fig. 12** Edge Masking

e. Quantization using K-Means is shown in the Equation in Figure 13:

```
# Color quantization with KMeans clustering
from sklearn.cluster import KMeans


k = 7
data = img.reshape(-1, 3)

kmeans = KMeans(n_clusters=k, random_state=42).fit(data)
img_reduced = kmeans.cluster_centers_[kmeans.labels_]
img_reduced = img_reduced.reshape(img.shape)

img_reduced = img_reduced.astype(np.uint8)

plt.axis("off")
plt.imshow(img_reduced)

<matplotlib.image.AxesImage at 0x7f111dae1a90>
```



**Fig. 13** Cartoonified image after Clustering




f. Applying Bilateral Filter for Smoothing of image in the Figure 14 of Smoothing of the Image:

```
[ ] # Bilateral Filter
blurred = cv2.bilateralFilter(img_reduced, d=7, sigmaColor=200,sigmaSpace=200)
cartoon = cv2.bitwise_and(blurred, blurred, mask=edges)

[ ] plt.subplot(1, 2, 1)
plt.axis("off")
plt.imshow(img)

plt.subplot(1, 2, 2)
plt.axis("off")
plt.imshow(cartoon)

<matplotlib.image.AxesImage at 0x7f111da27e50>
```



**Fig. 14** Smoothing of image

g. Exporting the image in the following Figure (see Fig. 15)

```
# export cartoon to a jpg file
cartoon_ = cv2.cvtColor(cartoon, cv2.COLOR_RGB2BGR)
cv2.imwrite('cartoon.png', cartoon_)
```

True

**Fig. 15** Exporting Image

## 4.1 Optimization

### Plotting all the transitions together:

This task is completed in the process by the help of Graphs. Firstly, making a list of all the images before and after the cartoonification of the image is done. One of the lists should contain the before images, and the other will contain after images. Then the list is taken as axes in a plot and each image is displayed in the form of a block in the graph using imshow() method.

After each image has been plotted on a graph separately, all the plots are to be represented in one single plot. This part of the process is crucial as we have to check the effectiveness/accuracy of the model and also the values of Loss functions.

### Qualitative Comparison:

On studying various research papers regarding methods used to carry out image Cartoonification White Box method proves to be producing the most clean contour containing images. Image abstraction in other methods produce images with unclear borderlines and messy contours of images containing human faces or clouds.

As CycleGAN being used into the white Box method nice darkened images with less distortion are produced. At the same time Fast Neural Style causes smooth color. In this approach fine features of images are preserved such as the minute reflection of the sky in the lake in the Figure.14 etc.

### Qualitative Evaluation:

Most widely-used qualitative evaluation technique is Frechet Inception Distance (FID), which is used in the white Box Method for the evaluation of the synthesized/ cartoonified image. Fairly, CartoonGAN not being experimented more on human faces the paper emphasizes about the the evaluation results gained by assessing the Scenery images widely.

The output of the method has the smallest FID, indicating that the features in the images synthesized or cartoonified using White Box technique give more good quality preserved images [10].

### Loss Functions:

Loss function tells us the amount of Data or features loss the image or list of images have suffered whilst the process

of cartoonification. There are two main components of Loss Function that help us determine the Loss of the Content in the image after cartoonification is applied. They are called Adversarial Loss and Content Loss. This is shown in the Equation 1.

$$L\_GAN = L\_G + L\_D \dots \text{Eq (1)}$$

#### **Adversarial Loss:**

The adversarial loss is calculated by comparing the discriminator's predictions for the generated samples with the true labels (i.e., real or fake). The generator is then updated to produce better samples that are more likely to be classified as real by the discriminator.

The adversarial loss function is used to train the generator network to produce cartoon images that are indistinguishable from real images. The discriminator is trained to classify images as either real or fake, and the generator is trained to produce images that can fool the discriminator into thinking they are real. The adversarial loss function is then used to update the generator's weights so that it can improve its ability to generate realistic cartoon images.

Mathematical representation for the same as Illustrated in Equation 2:

$$L\_adv = -E[\log(D(x))] - E[\log(1 - D(G(z)))] \dots \text{Eq (2)}$$

where:

$D(x)$  is the discriminator's output when given a real input  $x$

$G(z)$  is the generator's output when given a noise input  $z$

$E[.]$  represents the expected value over the entire training set.

#### **Content Loss:**

Content loss refers to the discrepancy between the original image and the cartoonized image produced by the generator network.

The content loss is calculated as the mean-squared error between the feature maps of the input image and the output image. The goal of the content loss is to ensure that the generator network preserves the content of the input image, while transforming it into a cartoonized image.

Content loss is a term used in CartoonGAN to refer to the measure of the difference between the input image and the generated output image in terms of their feature maps. It is used to ensure that the generator network produces cartoonized images that retain the content of the original image.

Mathematical Representation of the same as Illustrated in the Equation 3:

$$L\_content = 1/2 * \|F(x) - F(y)\|^2 \dots \text{Eq (3)}$$

where:

$x$  is the content image

$y$  is the generated image

$F(.)$  is a pre-trained feature extraction network (such as VGG)

$\|.\|$  represents the L2 norm (Euclidean distance)

$^2$  indicates squaring

## V.CONCLUSION

In conclusion, the NPR algorithm is an effective and time-efficient method for converting images into cartoons. It utilizes techniques such as edge detection, color simplification, and shading to create stylized images that resemble traditional artistic media. While there are some limitations to the NPR algorithm, it is still a valuable tool in various fields, including animation, game design, and graphic design.

Also, the study shows that the application of ML algorithms is effective in converting images into cartoons. GAN produces the most visually appealing and accurate cartoon images, but at the cost of longer processing time. CNN produces less visually appealing images but with faster processing time. The study highlights the importance of image preprocessing in achieving better results with ML algorithms. Future research could explore the use of other deep learning techniques and the integration of these techniques into mobile applications for real-time image to cartoon conversion.

## REFERENCES

- [1] Zhang, W., Wang, Y., & Wang, L. (2018). Image cartoonification using convolutional neural networks with augmented training samples. *Multimedia Tools and Applications*, 77(6), 7225-7241.
- [2] Huang, Y., Huang, Q., Lin, X., & Liu, L. (2021). Image Cartoonification Based on a Multi-Scale and Multi-Feature Generative Adversarial Network. *IEEE Access*, 9, 141994-142009
- [3] Characterizing and Improving Stability in Neural Style Transfer, 2017 - Agrim Gupta, Justin Johnson, Alexandre Alahi, Li Fei-Fei.
- [4] Several recursive and closed-form formulas for some specific values of partial Bell polynomials. (2022). *Advances in the Theory of Nonlinear Analysis and Its Application*, 6(4), 528-537. <https://atnaea.org/index.php/journal/article/view/175>
- [5] Image Style Transfer Using Convolutional Neural Networks, 2016 - Leon A. Gatys, Alexander S. Ecker, Matthias Bethge
- [6] Yang Chen, Yu-Kun, Yong Jin-Liu, "CARTOONGAN: Generative Adversarial Networks for Photo Cartoonification", Tsinghua University, China, 2020.
- [7] Heat transfer analysis of Radiative-Marangoni Convective flow in nanofluid comprising Lorentz forces and porosity effects . (2023). *Advances in the Theory of Nonlinear Analysis and Its Application*, 7(1), 61-81. <https://atnaea.org/index.php/journal/article/view/16>
- [8] Sathiyaraj, V. ., Josephine, M. S. ., & Jeyabalaraja, V. . (2023). Plant Disease Classification of Basil and Mint Leaves using Convolutional Neural Networks. *International Journal of Intelligent Systems and Applications in Engineering*, 11(2), 153–163. Retrieved from <https://ijisae.org/index.php/IJISAE/article/view/2606>
- [9] Xinrui Wang, Jinze Yu, *Learning to Cartoonize using White Box Cartoon Representation*, University of Tokyo, 2020.
- [10] Gao, Y., Xiong, R., & Zhang, Y. (2019). Image Cartoonification based on Deep Convolutional Neural Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(10), 1958001.
- [11] Huang, Q., Huang, Y., Lin, X., & Liu, L. (2019). An Image Cartoonification Method Based on Edge Detection and K-Means Clustering. In 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA) (pp. 685-690). IEEE
- [12] Amrutha K, Cartoonify Image Using OpenCV and Python, Published On June14,2022,URL: <https://www.analyticsvidhya.com/blog/2022/06/cartoonify-image-using-opencv-and-python/>, Accessed On: February 18,2023
- [13] Cartoonifying an Image Using Machine Learning by Madugula Sai Joshitha, Department of Information Technology, GMR Institute of Technology, pp 1081-1089 November 2022
- [14] Gong, C., Jia, J., & Leung, H. (2020). Image cartoonification using generative adversarial networks and K-means clustering. *Journal of Visual Communication and Image Representation*, 70, 102841.

© 2023. This work is published under <https://creativecommons.org/licenses/by/4.0/legalcode>(the“License”). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License.