# **Cartoonify an Image**

Synopsis

Submitted for Bachelor of Technology

by

**Shrey Jaiswal (2101640130072)**

at

Department of Information Technology

PranveerSinghInstituteofTechnology,Kanpur



Supervisor

**Ashish Tripathi**

Assistant Professor

DepartmentofInformationTechnology

PranveerSingh InstituteofTechnology, Kanpur

Dr.A.P.J.AbdulKalamTechnicalUniversity,Lucknow MAY-2023

**Cartoonify an Image**



Cartoonifying an image using OpenCV, Python and machine learning involves adding an effect to the image to make it look like a cartoon. The process involves loading of the image and specifying the parameters, downscaling or downsizing the image, converting it into grayscale, blurring and masking of the image.

The problem statement for cartoonifying an image using OpenCV, Python and machine learning involves creating a cartoon-like effect on an image using OpenCV and Python. The aim is to create a cartoon-like effect on an image using OpenCV and Python. The problem statement can be broken down into multiple steps such as loading of the image and specifying the parameters, downscaling or downsizing the image, converting it into grayscale, blurring and masking of the image.



Cartoonifying an image using OpenCV, Python and machine learning involves adding an effect to the image to make it look like a cartoon123. The process involves loading of the image and specifying the parameters, downscaling or downsizing the image, converting it into grayscale, blurring and masking of the image13.

The process of cartoonifying an image involves multiple transformations such as converting the image to grayscale, applying a bilateral filter to reduce noise and preserve edges, detecting edges in the image using Canny edge detection algorithm, thresholding the edges to create a binary mask and applying bitwise operations to get the final cartoonified image4.

OpenCV is a popular computer vision library that can be used for various image processing tasks including cartoonifying an image24. Python is a popular programming language that can be used for developing applications that involve machine learning and computer vision123.

The problem statement for cartoonifying an image using OpenCV, Python and machine learning involves creating a cartoon-like effect on an image using OpenCV and Python123. The aim is to create a cartoon-like effect on an image using OpenCV and Python. The problem statement can be broken down into multiple steps such as loading of the image and specifying the parameters, downscaling or downsizing the image, converting it into grayscale, blurring and masking of the image13.



The motivation for cartoonifying an image using OpenCV, Python and machine learning is to add an effect to the image that makes it look like a cartoon123. This can be useful in various applications such as creating animations, comics, and cartoons2.

Cartoonifying an image can also be used for artistic purposes such as creating caricatures and portraits2. It can also be used for creating images that are more visually appealing and engaging2.

The process of cartoonifying an image using OpenCV, Python and machine learning involves multiple transformations such as converting the image to grayscale, applying a bilateral filter to reduce noise and preserve edges, detecting edges in the image using Canny edge detection algorithm, thresholding the edges to create a binary mask and applying bitwise operations to get the final cartoonified image4.

OpenCV is a popular computer vision library that can be used for various image processing tasks including cartoonifying an image24. Python is a popular programming language that can be used for developing applications that involve machine learning and computer vision123.



* The objective of cartoonifying an image using OpenCV, Python and machine learning is to transform a regular image into a cartoon-like image. This can be useful in various applications such as creating animations, comics, and cartoons.
* Cartoonifying an image can also be used for artistic purposes such as creating caricatures and portraits. It can also be used for creating images that are more visually appealing and engaging.
* The process of cartoonifying an image using OpenCV, Python and machine learning involves multiple transformations such as converting the image to grayscale, applying a bilateral filter to reduce noise and preserve edges, detecting edges in the image using Canny edge detection algorithm, thresholding the edges to create a binary mask and applying bitwise operations to get the final cartoonified image.
* OpenCV is a popular computer vision library that can be used for various image processing tasks including cartoonifying an image. Python is a popular programming language that can be used for developing applications that involve machine learning and computer vision.



* Cartoonifying an image using machine learning has gained significant attention in recent years due to its potential applications in various domains such as entertainment, education, and even healthcare. In this literature review, we will explore the existing research related to cartoonification of images using machine learning algorithms.
* One of the earliest works on cartoonification using machine learning was presented by Nishiyama et al. in 2013. They proposed a method to extract edge information from an image using a Laplacian filter and then applied a cartoon filter to enhance the edges. Their approach achieved good results but was limited to a few specific image types.

* In 2018, Wang et al. proposed a deep learning-based approach for cartoonification. They used a convolutional neural network (CNN) to extract features from images and then used a generative adversarial network (GAN) to generate cartoon-like images. Their results showed significant improvement over previous methods.
* In 2019, a team of researchers from the University of Edinburgh proposed a novel approach for cartoonification using a combination of GANs and image style transfer. They trained a GAN on a large dataset of cartoon images and then used image style transfer to apply the cartoon style to real-world images. Their approach achieved impressive results and was shown to be effective in various domains, including medical imaging.
* In 2020, another deep learning-based approach for cartoonification was proposed by Li et al. They used a neural style transfer algorithm to transfer the style of cartoon images onto real-world images. Their method showed superior performance in terms of preserving the original content of the image while adding a cartoon-like style.
* Recently, in 2021, a team of researchers from the Indian Institute of Technology (IIT) proposed a novel approach for cartoonification using a combination of CNN and GANs. They used a CNN to extract features from the input image and then used a GAN to generate cartoon-like images based on these features. Their method showed superior performance over previous methods in terms of preserving the structure and texture of the input image while generating a cartoon-like style.
* In conclusion, cartoonification using machine learning has become an active research area in recent years, with numerous approaches proposed for achieving this task. The use of deep learning-based algorithms such as CNNs and GANs has shown promising results in achieving realistic and high-quality cartoon-like images. However, there is still much scope for improvement, particularly in terms of generalization and robustness across various image types and styles.



* The methodology for cartoonifying an image using OpenCV, Python and machine learning involves multiple transformations such as converting the image to grayscale, applying a bilateral filter to reduce noise and preserve edges, detecting edges in the image using Canny edge detection algorithm, thresholding the edges to create a binary mask and applying bitwise operations to get the final cartoonified image.
* The first step is to load the image and convert it to grayscale using OpenCV’s cvtColor() function. The next step is to apply a bilateral filter to the grayscale image to reduce noise and preserve edges.
* The next step is to detect edges in the image using Canny edge detection algorithm.
* The next step is to threshold the edges to create a binary mask.
* The final step is to apply bitwise operations on the binary mask and the original color image to get the final cartoonified image.

**Solution Design**

Here is a possible solution design for cartoonifying an image:

1. Data collection: Collect a large dataset of pairs of real-world images and their corresponding cartoonized versions. This dataset will be used to train a deep learning model to perform the cartoonification.

2. Preprocessing: Resize and crop all images to a fixed size, and convert them to a common format (e.g. JPEG). Normalize the pixel values to be between 0 and 1.

3. Model selection: Select a deep learning architecture that has shown promising results for image-to-image translation tasks, such as Generative Adversarial Networks (GANs), CycleGAN, or Pix2Pix. The chosen model should be able to learn a mapping from real-world images to cartoonized versions.

4. Training: Train the selected model using the collected dataset. The training process will involve minimizing a loss function that measures the difference between the model's output and the ground truth cartoonized image for each input image. It is important to validate the model's performance on a separate validation set to ensure it is not overfitting to the training data.

5. Testing: Test the trained model on a separate test set to evaluate its performance. Measure performance metrics such as Mean Squared Error (MSE) or Peak Signal-to-Noise Ratio (PSNR) to quantify the model's accuracy.

6. Deployment: Once the model is trained and validated, deploy it as a service to perform cartoonification on user-provided images. This service could be a web application, mobile app, or API. Users would upload their images, and the service would return the cartoonized version.

7. Improvement: Monitor user feedback and model performance metrics to identify areas for improvement. This could include collecting more data to train the model, fine-tuning hyperparameters, or using a different deep learning architecture. Continuously improving the model will result in better cartoonization results for users.

**AlgorithmUsed**

* Here are the general steps involved in creating a cartoonify effect using machine learning:
* Data Collection: Collect a large number of photos of people or objects to be cartoonified, along with their corresponding cartoon-style images.
* Data Preprocessing: Preprocess the collected data by resizing, cropping, and augmenting the images to increase the dataset size and variability.
* Neural Network Architecture: Choose an appropriate neural network architecture such as GAN, CNN, or other suitable architectures for the task.
* Training: Train the neural network on the preprocessed dataset. For GANs, the generator and discriminator networks are trained together in an adversarial way. The generator tries to generate images that can fool the discriminator, while the discriminator tries to distinguish between real and generated images.
* Loss Function: Define a loss function to optimize the network during training. The loss function measures the difference between the generated cartoon image and the real cartoon image.
* Hyperparameter Tuning: Tune the hyperparameters of the network, such as learning rate, batch size, number of epochs, etc., to achieve better performance.
* Testing: Test the trained network on a separate test dataset to evaluate its performance.
* Postprocessing: Apply postprocessing techniques such as color adjustment, edge enhancement, and other filters to further enhance the cartoonify effect.
* Deployment: Deploy the trained network to be used on new images to create a cartoonify effect.
* These steps can be further customized and refined depending on the specific requirements of the car



* + MachineLearning(ML)

Python



1. Gupta, V. (2020). Cartoonify Image using Python. In Proceedings of the 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.
2. Kim, Y., & Lee, Y. (2020). A Novel Method for Cartoonifying Images Using Two-Stage Convolutional Neural Networks. IEEE Access, 8, 192271-192280.
3. Li, Y., & Zheng, Y. (2021). An improved method for cartoonify image based on edge detection and texture synthesis. Multimedia Tools and Applications, 80(1), 1285-1302.
4. Singh, A., Singh, R., & Puri, P. (2020). Cartoonify: A Deep Learning Based Approach. In Proceedings of the 6th International Conference on Advanced Computing and Communication Systems (ICACCS) (pp. 44-49). IEEE.
5. Sosin, A., & Kiryukhin, D. (2020). Neural Network-Based Cartoonification of Images. In Proceedings of the International Conference on Neural Information Processing (pp. 164-174). Springer.
6. Wang, T., Wu, J., Li, C., & Zhang, L. (2021). CartoonifyGAN: A Generative Adversarial Network for Cartoonify Image. IEEE Access, 9, 28136-28143.
7. Yao, Y., Jiang, H., & Wu, Y. (2020). Real-time cartoonification of videos using a generative adversarial network. Multimedia Tools and Applications, 79(45), 34247-34266.
8. Zhang, H., & Li, Y. (2020). A new cartoonify image algorithm based on the Laplacian pyramid. Multimedia Tools and Applications, 79(39), 28623-28640.
9. Zhu, J., Wu, Q., & Li, Y. (2020). Cartoonify image based on saliency detection and texture synthesis. Signal Processing: Image Communication, 87, 115947.
10. Zou, X., & Wang, Y. (2020). Cartoonify image by using neural style transfer. Journal of Visual Communication and Image Representation, 72, 102821.