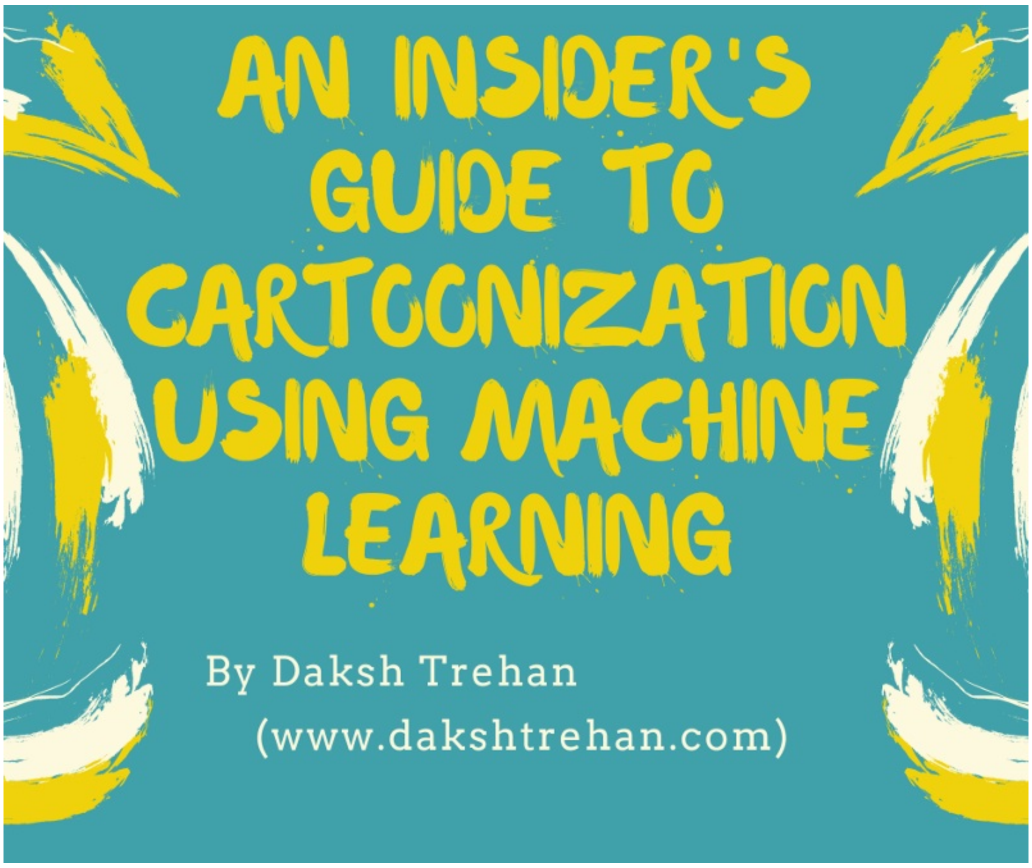
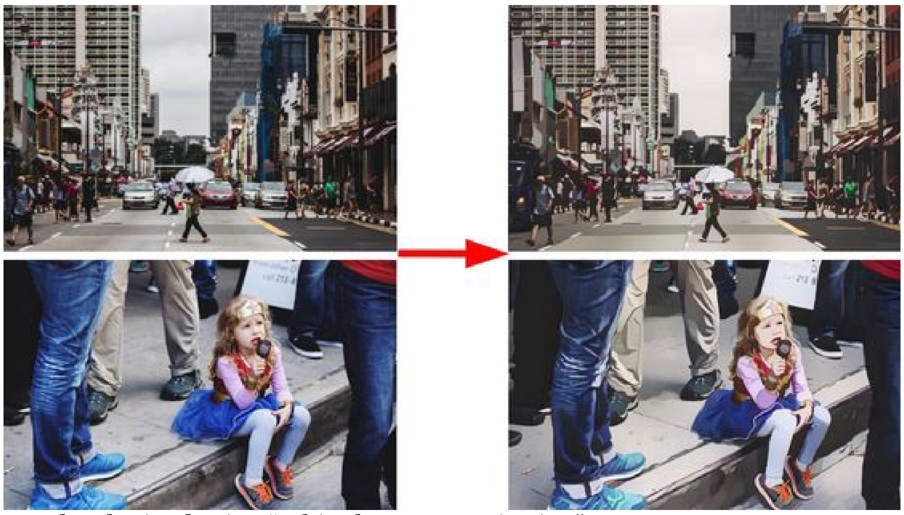
Deep Learning, Computer Vision

An Insider’s guide to Cartoonization using Machine Learning

Implementing white-box models to “cartoonize” real high-quality images.



Cartoonization is itself a classic art but, the evolution in the field of Machine Learning is covering almost every realm. Today, in this article we are going to learn about a new method called ‘ **White box Cartoonization’** that reconstructs a photo into an animation style focusing on the expression extracting elements, making the plans entirely controllable and flexible.


Results obtained using “White-box-Cartoonization”, Source

Introduction to Model

I bet everyone must be fond of cartoons and they must have been an integral part of your childhood too. And apart from these relishing memories, it might be a career choice for some of us.

But Machine Learning is constantly evolving thus expanding in almost every field. And research work done by Xinrui Wang and Jinze Yu has enabled us to cartoonize real high-quality images with just a little training.

The process of converting real-life high-quality pictures into practical cartoon scenes is known as **cartoonization**.

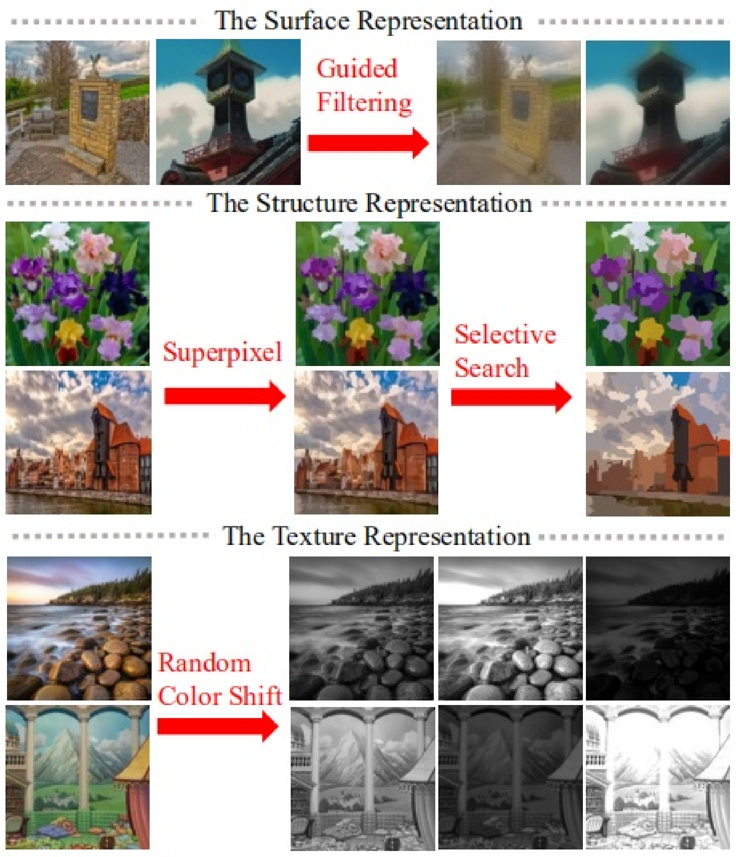
Earlier models that proposed the same approach used **black-box models**, the former model achieves great accuracies but downturns the stylization quality causing some bad cases. Like, every cartoon workflow considers different features, these variations pose a relevant effect on black-box models.

To overcome the drawbacks of the former model, more emphasis was given upon human painting behaviors and cartoon images of different styles, and a **white-box model** was developed.

The model decomposes images into three different cartoon representations, which further counsel the network optimization to generate cartoonized images.

1. **Surface Representation:** It helps to extract smooth surfaces of the image that contains a weighted low-frequency component where the color composition and surface texture are retained along with edges, textures, and details.
2. **Structure Representation**: It helps to derive global structural information and sparse color blocks, once done we implement adaptive coloring algorithms like the Felzenswalb algorithm to develop structural representation that can help us to generate sparse visual effects for celluloid styled cartoon process.
3. **Textured Representation**: It helps us to retain painted details and edges. The three-dimensional image is converted to single-channel intensity map that helps to retain pixel intensity compromising color and luminance, it follows the approach of manual artist that first draw a line sketch with contours and then apply colors to it.

The extracted outputs are fed to a Generative Neural Networks (GAN) framework, which helps to optimize our problem making the solution more flexible and diversified.


The Surface Representation
Guided Filtering
The Structure Representation
Superpixel
Selective Search
The Texture Representation
Random Color Shift

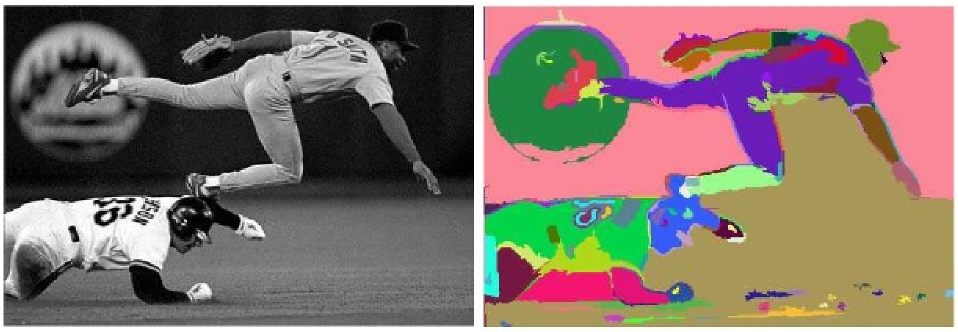
Source

Proposed Approach

Preprocessing

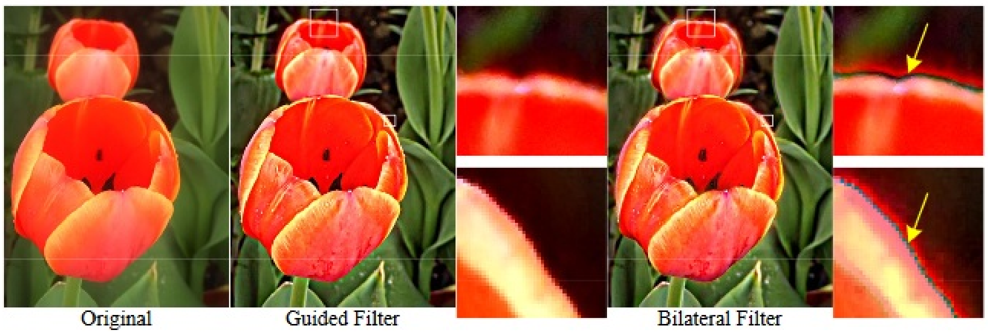
Along with the proposed three-step approach, preprocessing is an important part of our model. It helps to smoothen the image, filter the features, converting it to sketches, and translating the output from a domain to another. After implementing these related work we can be sure that the output generated by our model will give us the best output that retains the highest quality features.

- **Super-pixel and Structure Extraction:** This method is used to divide the image into regions and defining a predicate for measuring the boundary between two regions. Based on the predicate segmentation, an algorithm is developed whose decision is based on a greedy technique but still helps to satisfy global properties. After identification of contours, we implement ***Gradient Ascent*** to initialize the image with rough clusters and iteratively amend the clusters until convergence. Advancing our process, to develop a cartoon-like segmentation method we use **the *Felzenszwalb algorithm*** that helps us to seize global content information and produce practically usable results for celluloid style cartoon workflows.



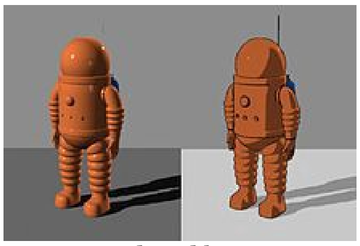
Before vs After, Source

- **Image Smoothening**: To extract smooth and cartoon resembling surfaces from images, ***Guided filters*** are used. A guided filter is an advanced version of ***Bilateral filters*** with better near the edge behavior. The goal is simply *removing/significantly decreasing* the noise and obtaining useful image structures. The filtering output of the guided filter is an optimal linear transform of an input image. Following the approach of Bilateral filters it retains smoothing property and in addition, is free from gradient reversal artifacts.



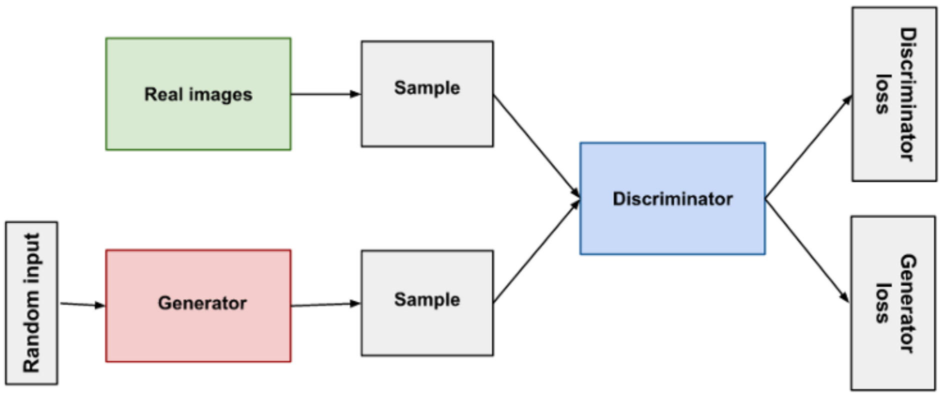
Original vs Guided filter vs Bilateral filter, Source

- **Non-photorealistic Rendering:** It helps to convert images into artistic styles such as sketching, painting, and water-coloring. To expand its functionality we use it with ***Neural Style Transfer Methods*** that helps to sum up the style of one image and another. The combined piece of code helps to mark semantic edges while segregating image details. But in the "White box cartoonization" method a single image is utilized and learns cartoonist features from a set of animated visuals allowing our model to produce high-quality output on diverse cases.



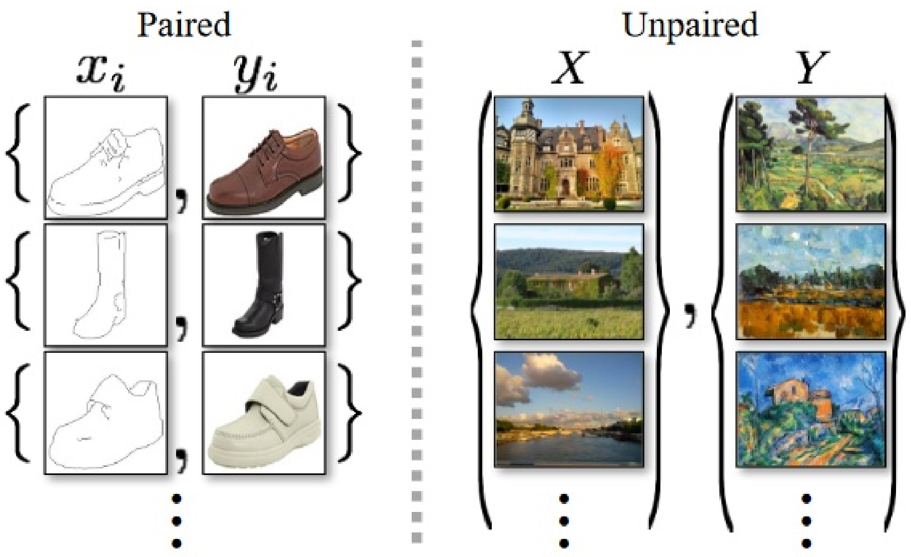
Output produced by NPR, Source

- **Generative Adversarial Network:** It is an image synthesizer that helps to generate new data using joint probability. To generate new images it uses Generator and Discriminator. The generator makes images and Discriminator checks images to be real or fake and then sends feedback to the generator thus asking him to generate better data. The more both networks are trained, the better images we get.



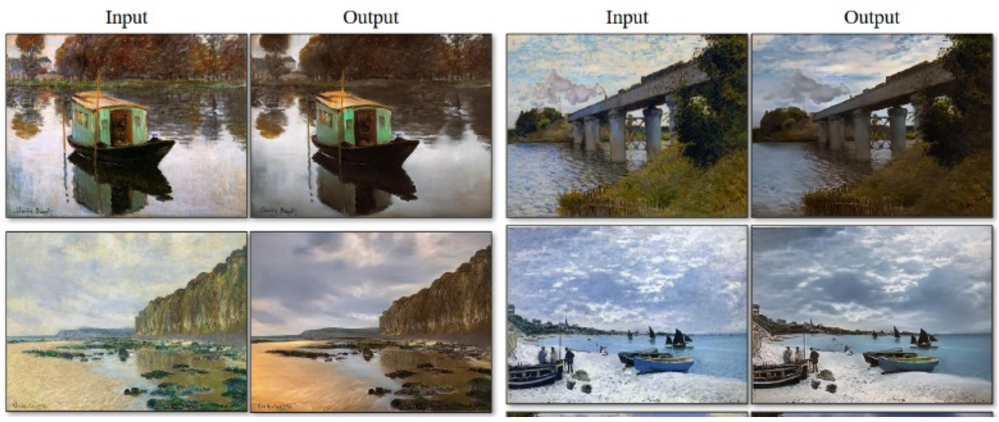
GAN Model Architecture, Source

- **Image-to-Image Translation**: The drawback with GAN is, it only works for given training data, but paired training data isn't always available. To overcome the drawback we employ cycleGAN where the goal is to translate an image from a source domain X to a target domain Y even in absence of paired training data.



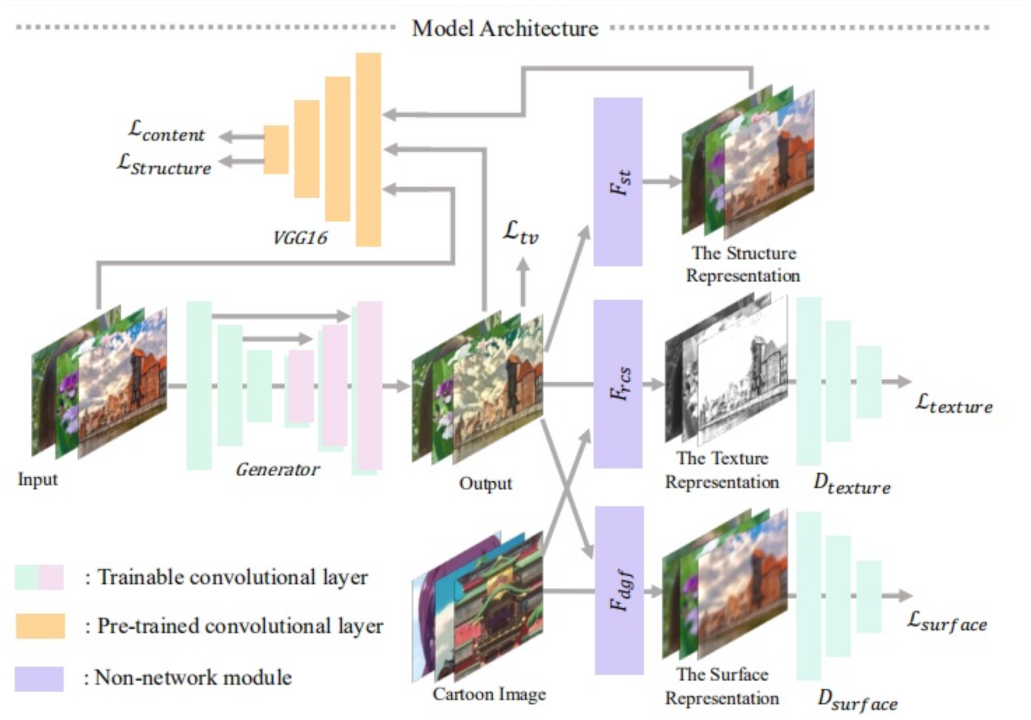
Paired vs Unpaired data, Source

To continue with the process, we segregate image features which enforces the network to learn different features with separate objectives, making the process more robust.



Output of cycleGAN aka Image-to-Image translation, Source

Understanding the whole model

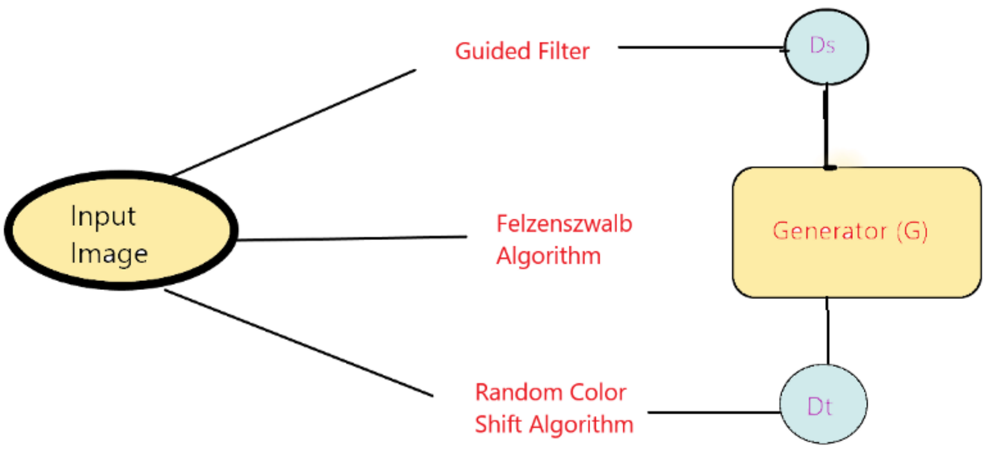


Model layout, Source

The input image is dissolved in three parts wiz *Surface Representation, Structural Representation*, and *Texture Representation*. A GAN model with a generator G and two discriminators Ds and Dt are introduced. The goal of Ds is to characterize surface features extracted from model outputs and cartoons, whereas Dt is responsible for separate textural information from model outputs and cartoons. To pluck high-level features and to impose a spatial constraint on global content between outputs and provided paired cartoons we use pre-trained *VGGNetwork*.

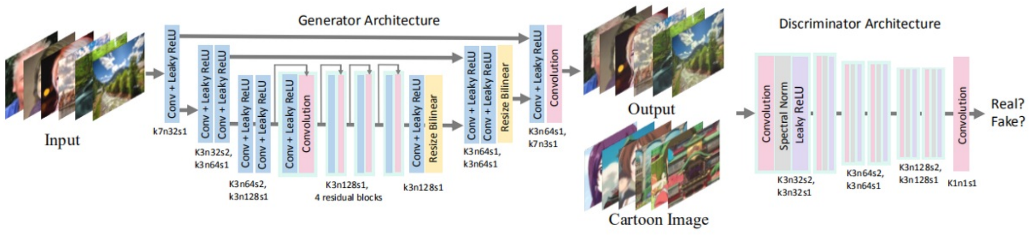
The Workflow

- The input is first passed through Surface representation where Structural and Textural features are removed, once we imitate cartoon painting style and smooth surfaces, the output is passed through guided filters in order to retain smooth edges. A discriminator *Ds* is proposed to verify whether result and paired cartoon images have similar surfaces, and regulate the generator *G* to learn the information stored in the extracted surface representation.
- The structural features are then passed through Structural representation that clear boundaries in the cellular style framework and then we implement Felzenszwalb algorithm to segment the areas. The algorithm assists us in coloring each segment with an average pixel value. To impose a spatial constraint on global content between outputs and provided paired cartoons we use pre-trained *VGGNetwork*.
- As discussed earlier the variation of luminance and color information are non-trivial issues to the model, therefore, we choose a random color shift algorithm to convert three-channel input to single-dimension outputs that cling to high-quality features. A discriminator Dt is then proposed to verify textual features from the result and paired cartoon image, and regulates generator G to learn the information stored in extracted texture representation.



The workflow for our model.

Model Architecture

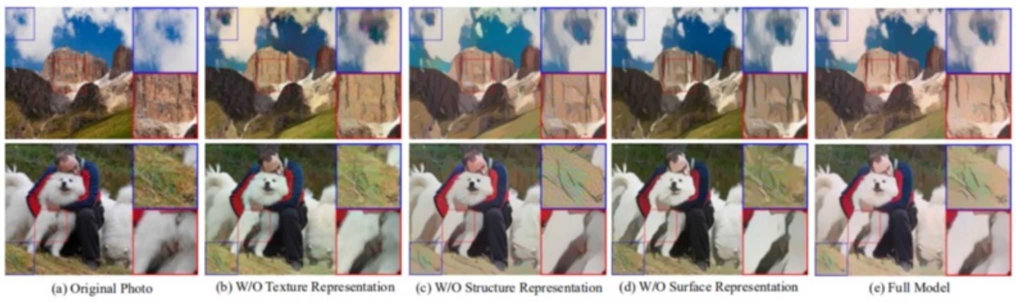


Model Architecture, Source

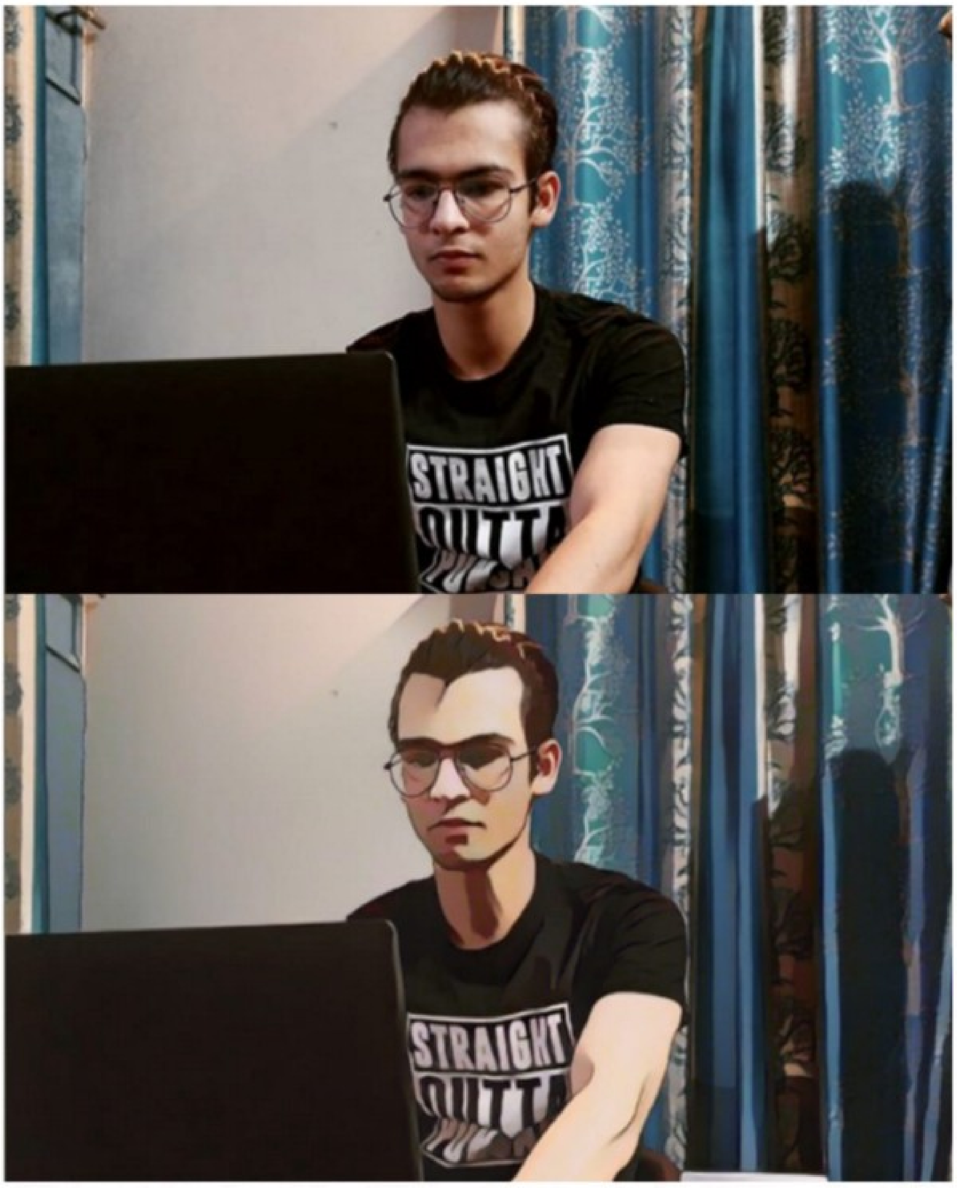
Since our model is based on GAN, we operate both Generator and Discriminator in our model. The generator is a fully-convolutional network stride=2. The network further consists of three kinds of layers: convolution, Leaky ReLU, and bilinear resize layers.

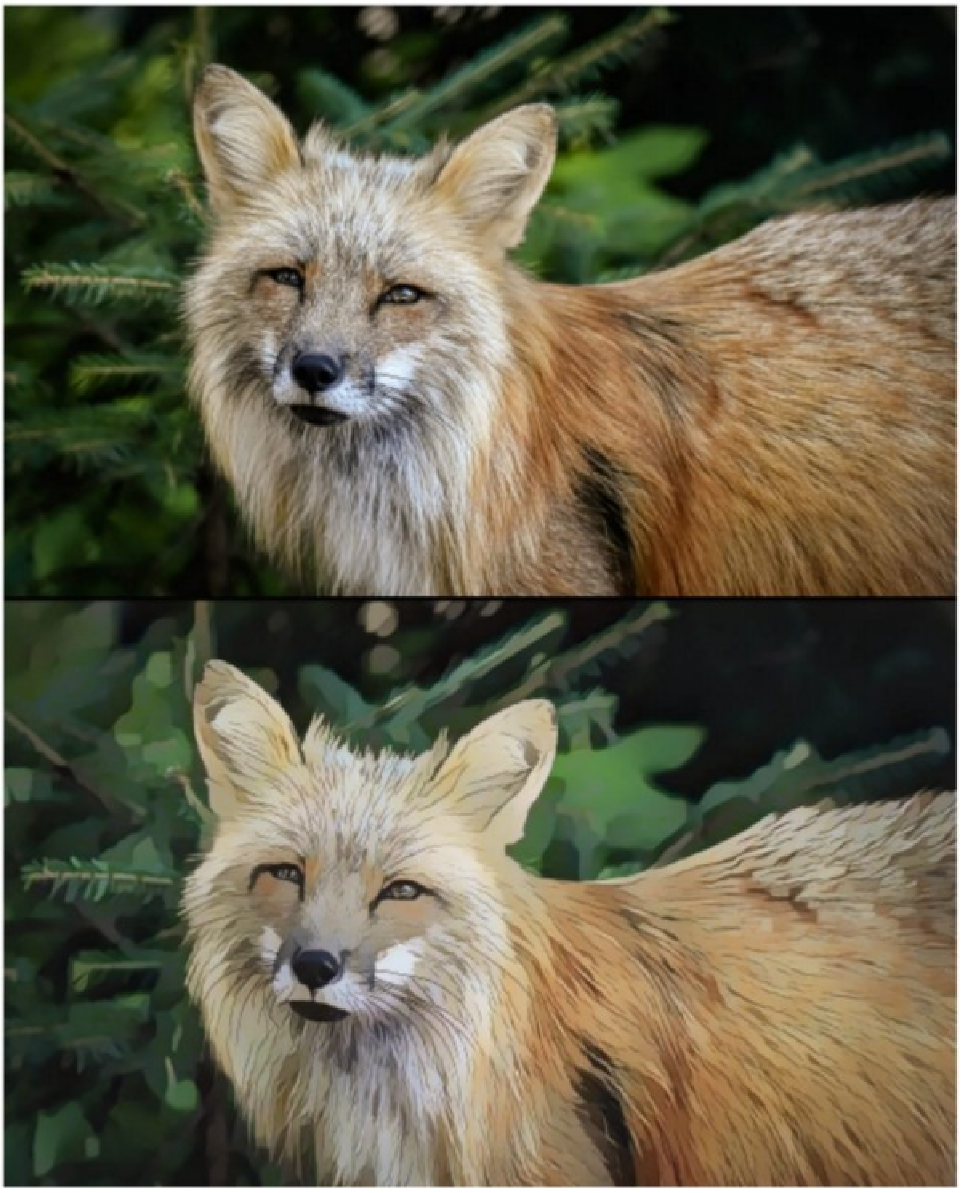
In the discriminator network, we employ PatchGAN, where the last layer is a convolution layer. Each pixel in the output feature map corresponds to a patch in the input image. Each patch size is used to verify whether the patch belongs to cartoon images or generated output. The Patch GAN enhances the operation of the discriminator and fastens up the training.

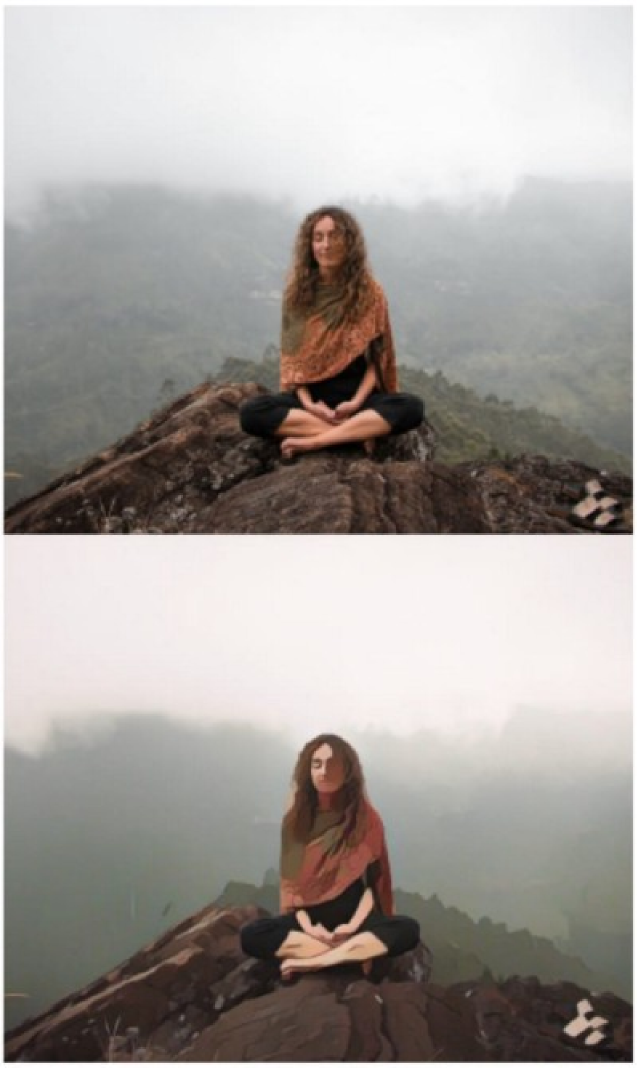
Output


(a) Original Photo
(b) W/O Texture Representation
(c) W/O Structure Representation
(d) W/O Surface Representation
(e) Full Model

Outputs wiz different Representations, Source


STRAIGHT
OUTTA




Outputs produced by Model, Source

Code

The code for model can be found at:

[dakshtrehan/White-box-Cartoonization](github.com)
*Join me at www.dakshtrehan.com ; www.linkedin.com/in/dakshtrehan Store test images in /test_code/test_images Run…*github.com

And

SystemErrorWang/White-box-Cartoonization
*Tensorflow implementation for CVPR2020 paper "Learning to Cartoonize Using White-box Cartoon Representations".*github.com

---

Conclusion

Hopefully, this article will help you to understand about cartoonization of images using Machine Learning in the best possible way and also assist you to its practical usage.

As always, thank you so much for reading, and please share this article if you found it useful!

References

[1] Wang, Xinrui, and Yu, Jinze. Learning to Cartoonize Using White-Box Cartoon Representations. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[2] Pedro F Felzenszwalb and Daniel P Huttenlocher. Effificient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[3] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided fifilter. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1838–1847, 2018.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[5] Kaiming He, Jian Sun, and Xiaoou Tang. *Guided Image Filtering*. Department of Information Engineering, The Chinese University of Hong Kong, Microsoft Research Asia, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China

[6] Non-photorealistic rendering

[7] CycleGAN: How Machine Learning Learns Unpaired Image-To-Image Translation

[8] Minimum spanning tree-based segmentation

[9] Overview of GAN Structure

[10] Cewu Lu, Li Xu, and Jiaya Jia. Combining sketch and tone for pencil drawing production. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, pages 65–73. Eurographics Association, 2012.

Feel free to connect:

> *LinkedIn ~ https://www.linkedin.com/in/dakshtrehan/*
>
> *Instagram ~ https://www.instagram.com/_daksh_trehan_/*

> *Github ~ https://github.com/dakshtrehan*

Follow for further Machine Learning/ Deep Learning blogs.

> *Medium ~ https://medium.com/@dakshtrehan*

Want to learn more?

Detecting COVID-19 Using Deep Learning

The Inescapable AI Algorithm: TikTok

Why are YOU responsible for George Floyd's Murder and Delhi Communal Riots?

Why Choose Random Forest and Not Decision Trees

Clustering: What it is? When to use it?

Start off your ML Journey with k-Nearest Neighbors

Naive Bayes Explained

Activation Functions Explained

Parameter Optimization Explained

Gradient Descent Explained

Logistic Regression Explained

Linear Regression Explained

Determining Perfect Fit for your ML Model

> *Cheers!*

By Daksh Trehan on July 10, 2020.

Canonical link

Exported from Medium on July 15, 2020.