Cartoonify an image

Group 18

Group Members:

Name	Roll No
Chaitanya Kapoor	AM.EN.U4AIE19022
Kiran Khanna	AM.EN.U4AIE19037
Dishith Reddy	AM.EN.U4AIE19041

Abstract

Cartoonifying an image is the process of adding a filter to an input image. We have developed a web interface to carry out cartoonify of images using machine learning techniques. We have used

In this project, we are cartoony an image and image to image translation using different algorithms. First, we convert an image (Image of my face) to a cartoon image using different algorithms. In the output, we will get a cartoon image of an input given image.

Second, we are applying style transfer and deep dream methods to an image. Style transfer takes two input images one is a content image (image of a person) and a style image(image of a painting) and blends them together so the output image looks like the content image, but "painted" in the style of the style reference image. Deep Dream takes an input image and visualizes the patterns learned by a neural network in the given image.

Third, Image to image translation takes two images as an input. It aims to transfer images from a source domain to a target domain while preserving the content representations.

Objectives

Give one or two images using different algorithms we play with images by converting them into different output images using different algorithms and methods.

- Given an image converts to cartoon image.
- Given a style and content image, It blends them together and gives a content-style image
- Given an image(sky containing clouds image) It recognizes the pattern and outputs the recognized pattern image.
- In Image to image translation,
    - pix2pix learns a mapping from input images to output images. Given an input image and a real image, it predicts the imaginary image based on input image for real image
    - CycleGAN can translate from one domain to another without a one-to-one mapping between the source and target domain. The example has given a horse image and predicts the zebra image as output.

Assumptions

- **Deep Dream**
  It accomplishes this by sending a picture through the network and then calculating the image's gradient in relation to the activations of a certain layer. The image is then altered to boost these activations, strengthening the patterns detected by the network and producing a dream-like visual.
- **Style Transfer**
  This is accomplished by adjusting the output image's content statistics to match the content image's content statistics and the

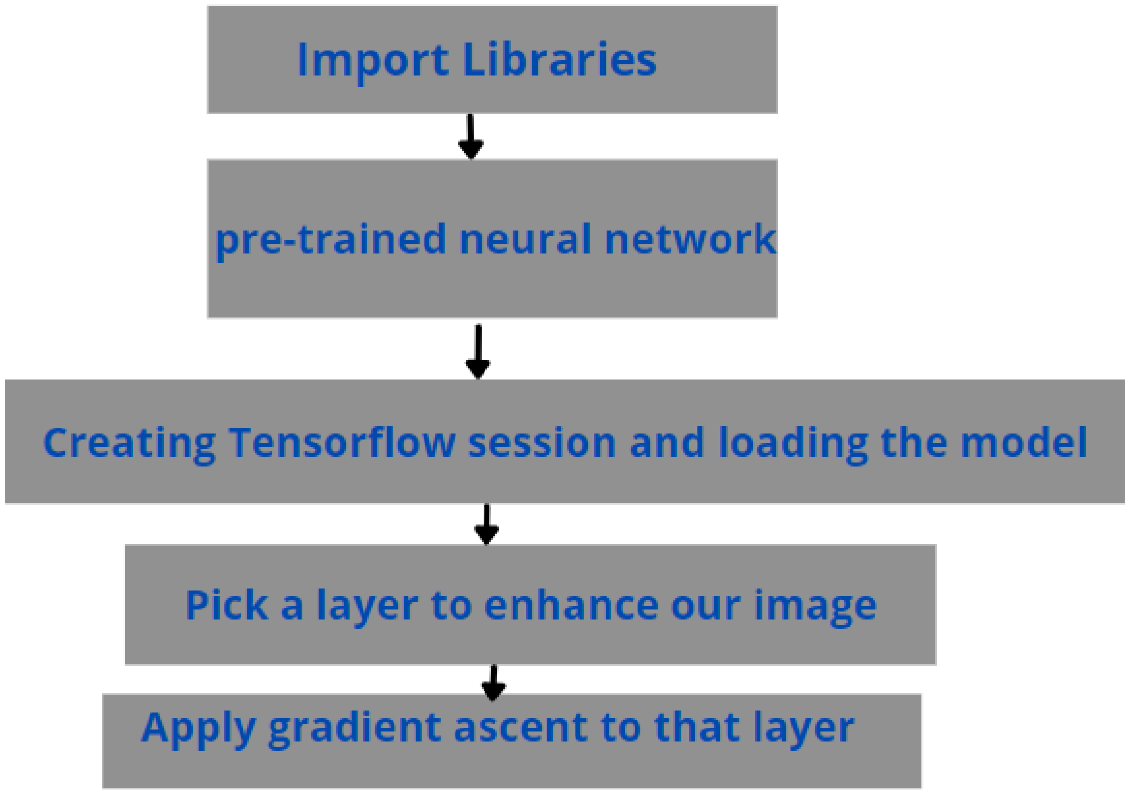
style reference image's style statistics. A convolutional network is used to extract these data from the pictures.

- **CycleGAN**
  CycleGAN uses a cycle consistency loss to enable training without the need for paired data. In other words, it can translate from one domain to another without a one-to-one mapping between the source and target domain.
- **Cartoon GAN**
  The key idea of a GAN model is to train two networks (i.e., a generator and a discriminator) iteratively, whereby the adversarial loss provided by the discriminator pushes the generated images towards the target manifold.

- **Pix2Pix**
  Pix2Pix is a service that converts your drawings and sketches into paintings in a matter of seconds. The image is created using artificial intelligence, machine learning, and conditional adversarial networks, among other things.
  A pix2pix network could be trained on a training set of corresponding pairs to learn how to make full-color from black & white images. Once a pix2pix network has been trained on such a dataset, it could then be used to color arbitrary black & white images.

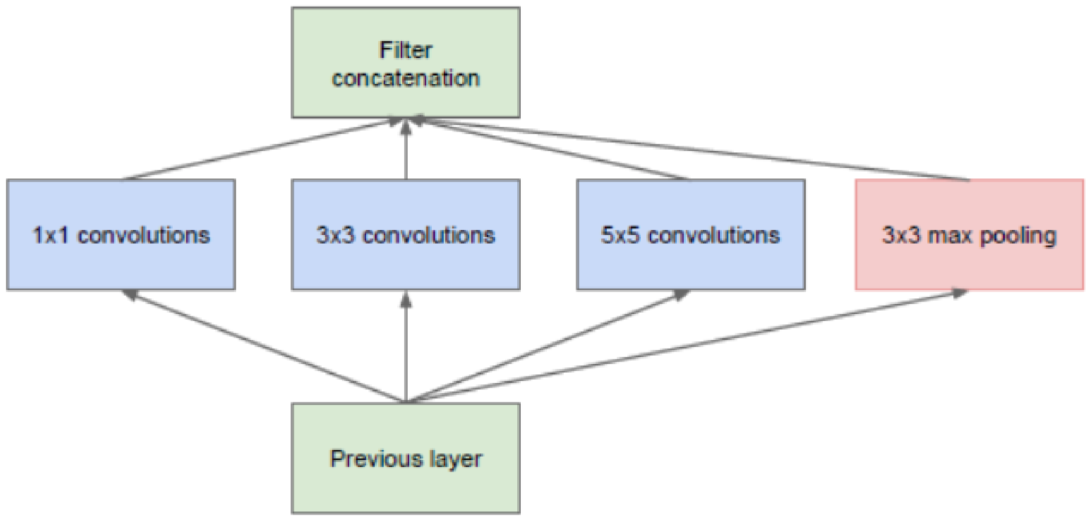
System Architecture

- **Deep Dream**


Import Libraries
pre-trained neural network
Creating Tensorflow session and loading the model
Pick a layer to enhance our image
Apply gradient ascent to that layer

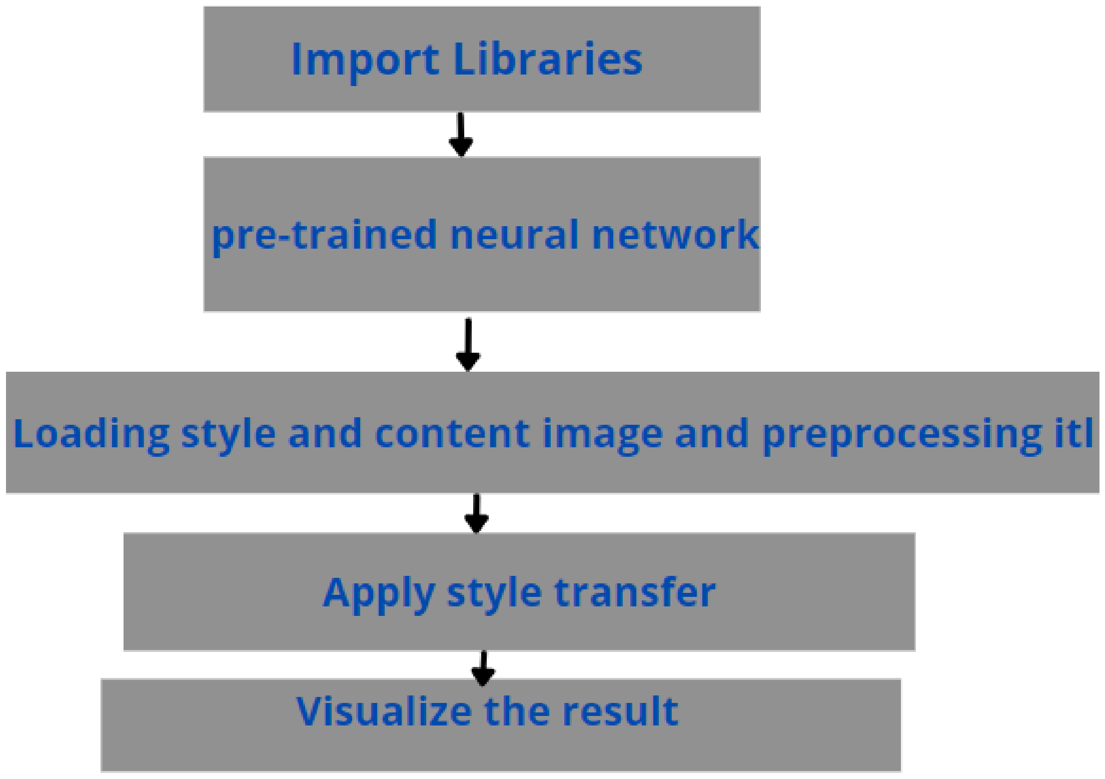
- **InceptionNet**

  Inception v3 is a convolutional neural network that was developed as a Google net module to aid with picture processing and object recognition. Inception does 1✕1 convolutional transformation, 3✕3, 5✕5, and more others, and some pooling layers. Then stack all the outputs of the transformation and let the model choose how to use that information.
  Inception basic module



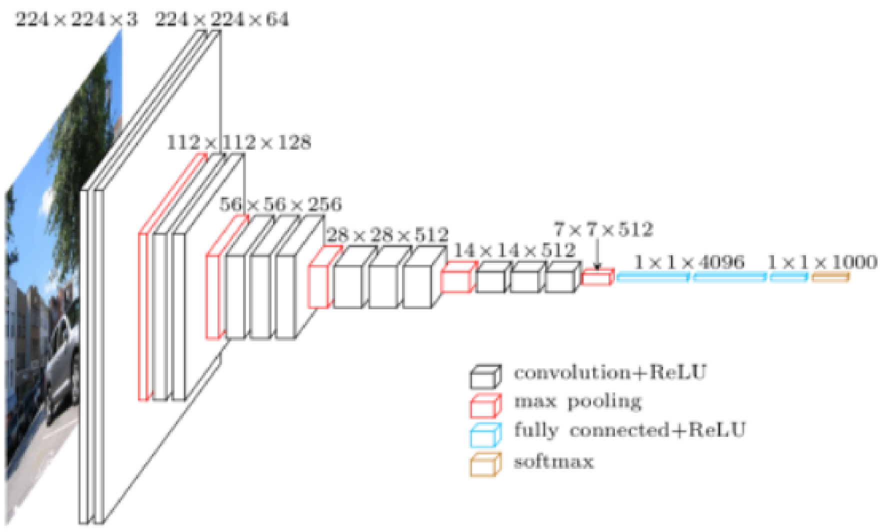
Inception v2, 3×3 convolutions are used in the space of 5×5 and boosts the performance. This also decreases computational time and thus increases computational speed because a *5×5* convolution is 2.78 times more expensive than a *3×3* convolution. So, Using two *3×3* layers instead of *5×5* increases the performance of architecture.
Inception v3 is almost similar to Inception v2 except for some updates. Use of RMSprop optimizer, Batch Normalization in the fully connected layer of the Auxiliary classifier, and Use of *7×7* factorized Convolution.

- **Style Transfer**


Import Libraries
pre-trained neural network
Loading style and content image and preprocessing itl
Apply style transfer
Visualize the result

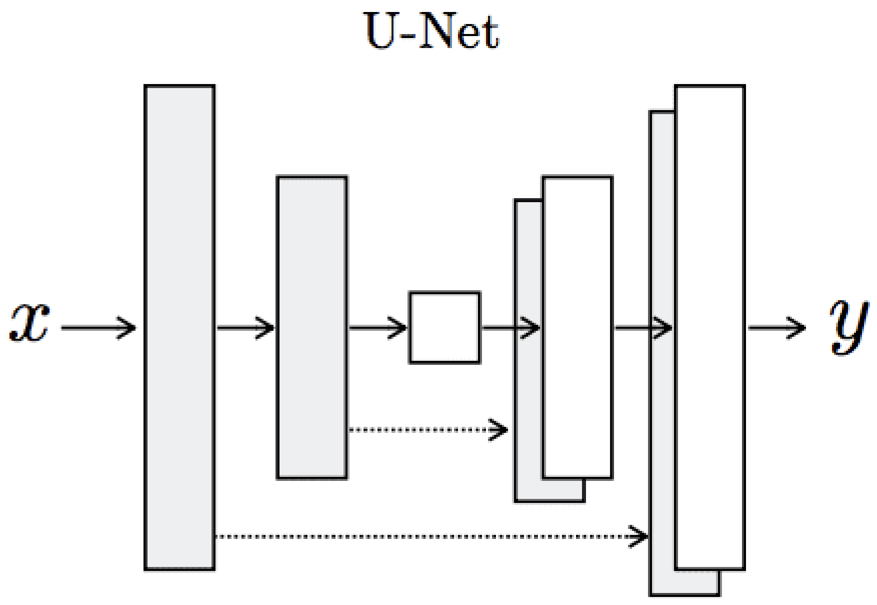
- **VGG16**

  It is regarded as one of the best vision model architectures ever created. The most distinctive feature of VGG16 is that, rather than having a huge number of hyper-parameters, they concentrated on having 3x3 filter convolution layers with a stride 1 and always utilized the same padding and max pool layer of 2x2 filter stride 2. In the end, it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it having 16 layers that have weights. This network is a pretty large network and it has about 138 million parameters.



Architecture of VGG16

- **Pix2Pix**
  The Pix2Pix model is a sort of conditional GAN, in which the output picture creation is dependent on input, in this case, a source image. A source picture and a target image are given to the discriminator, who must assess if the target is a real transformation of the source image.
  U-Net Generator
  The U-Net model is an encoder-decoder model for image translation where skip connections are used to connect layers in the encoder with corresponding layers in the decoder that have the same sized feature maps.
  The encoder element of the model is made up of convolutional layers that downsample the input source picture to a bottleneck layer using a 2 x 2 stride. The model's decoder reads the bottleneck output and upsamples it to the appropriate output picture size using transpose convolutional layers.



So that the first downsampling layer is connected to the last upsampling layer, the second downsampling layer is connected to the second last upsampling layer, and so on, skip links are established between the levels with the same sized feature maps. The connections join the channels of the downsampling layer's feature map with those of the upsampling layer's feature map.

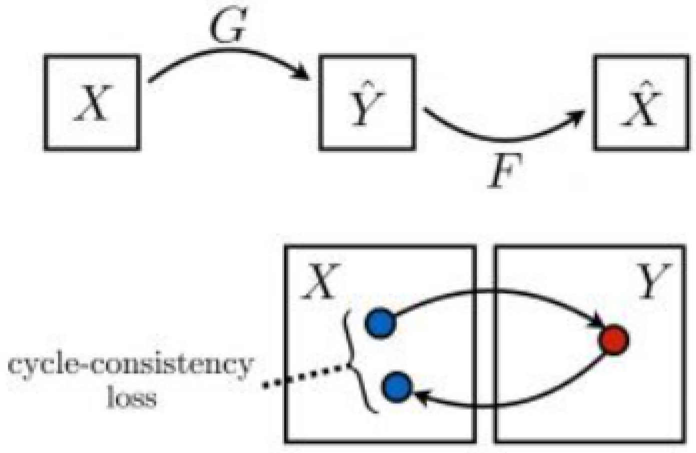
PatchGAN Discriminator

The PatchGAN discriminator used in pix2pix is another unique component of this design. The PatchGAN / Markovian discriminator works by classifying individual (N x N) patches in the image as “real vs. fake”, as opposed to classifying the entire image as “real vs. fake”.

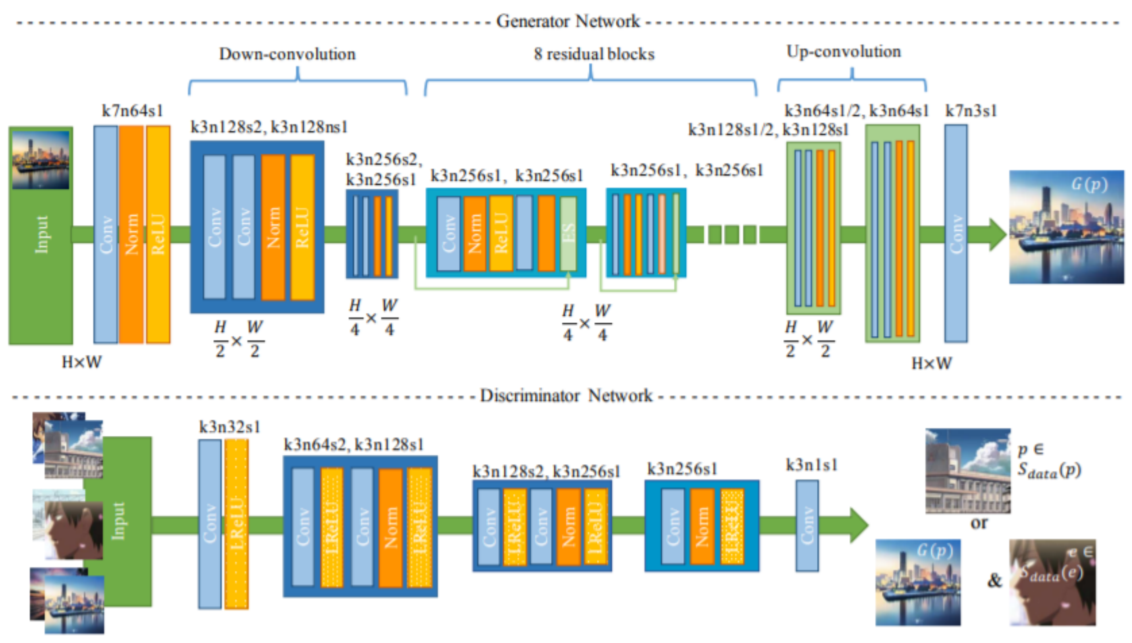
- **CycleGan**

  The key idea behind CycleGANs is that they can build upon the power of the PIX2PIX architecture, but allow you to point the model at two discrete, unpaired collections of images.

  The way CycleGANs are able to learn such great translations without having explicit X/Y training images involves introducing the idea of a full translation cycle to determine how good the entire translation system is, thus improving both generators at the same time.



- **Cartoon Gan**
  A GAN framework consists of two CNNs. One is the generator G which is trained to produce output that fools the discriminator. The other is the discriminator D which classifies whether the image is from the real target manifold or synthetic. The generator network G is utilized in CartoonGAN to map input pictures to the cartoon manifold.
  After the model has been trained, it is stylized into a cartoon.
  G starts with a flat convolution stage, then two down-convolution blocks to compress and encode the pictures spatially. This step extracts useful local signals for downstream transformation. The content and manifold feature are then built using eight leftover blocks with the same layout.
  Finally, the output cartoon-style images are reconstructed by two up-convolution blocks which contain a fractionally strided convolutional layer with stride 1/2 and a final convolutional layer with 7 × 7 kernels.



Contribution of each member of the team

Chaitanya Kapoor has done style transfer implementation and pencil sketch and Bilateral Filter implementation in cartoonify method.

Kiran Khanna has done Image to Image Translation(Pix2Pix gan and cycle Gan implementation) and also cartoon Gan implementation for cartoonify

Dishith Reddy has done deep dream implementation and Detail Enhancement for cartoonify .

Input to the system

- **Cartoonify**

- **Style Transfer**
  - **Content Image**

  - **Style Image**

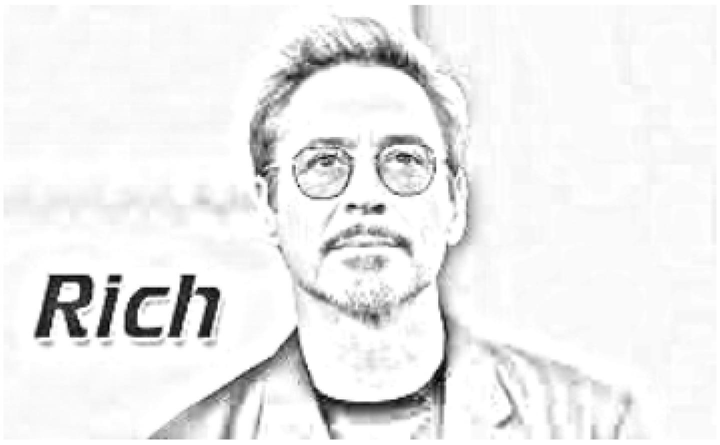
- **Deep Dream**

- **Pix2pix**

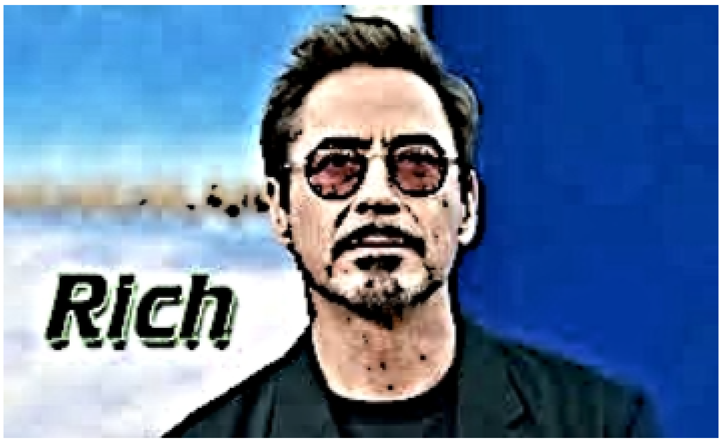
- **CycleGan**

Output

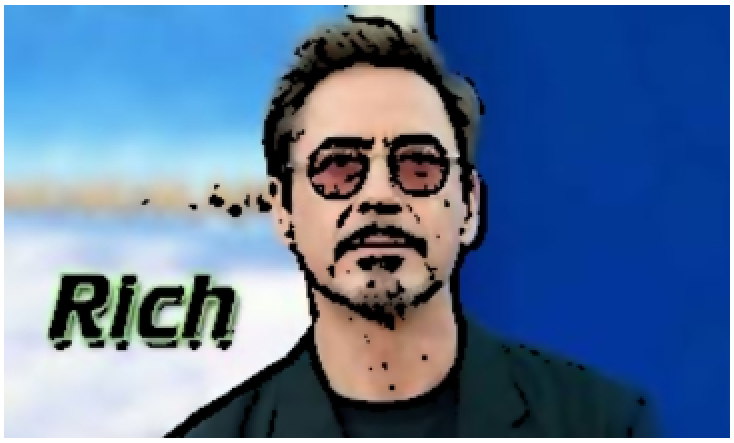
- Cartoonify
- Pencil Sketch



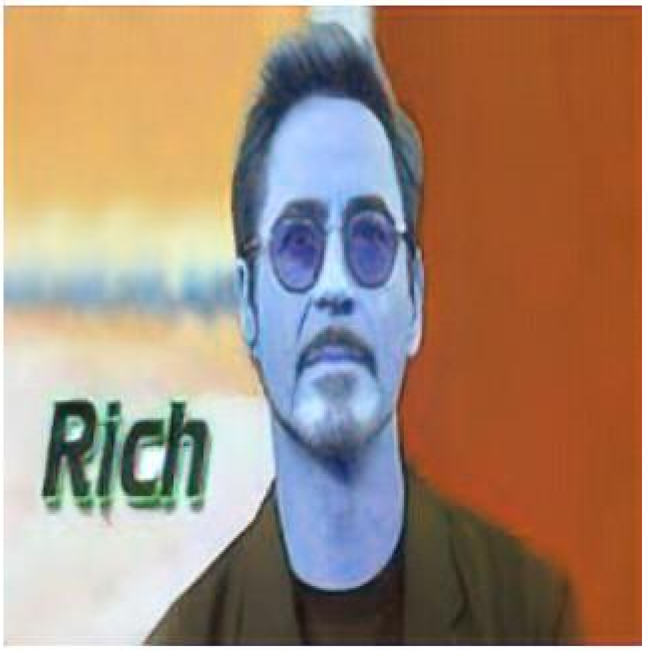
- Detail Enhancement



- **Bilateral Filter**



- **Cartoon Gan**



- **Pix2Pix**