Style Transfer on Images

The problem tackled in this report is Transfering the style of on image to contents of another image and thus creating a third image having the style similar to first image and content similar to second one.
For achieving this I have used approaches from two different research papers and implemented them.

Style transfer on Images using "Leon A.Gatys 's paper on A Neural Algorithm of Artistic Style"

In this approach we used a VGG network with function to combine both the style and content loss and then by optimizing this loss in one iteration we will reduce the style and content loss thus making the produced image better.

Link to Research paper: https://arxiv.org/pdf/1508.06576.pdf

Major Disadvantage:

The overall process is very slow and takes too much time for a single image thus creating real time applications is not possible using this approach.

- For processing a single image on non gpu accelerated systems it take 4 to 5 hours. On my system it took 5 hours.
- For processing a single image on gpu accelerated systems it takes 1-2 minutes . On Google colab it took 1 minute.

Even on gpu accelerated system the results are not real time thus implementing this approach in real-time applications (webcams) is not a good idea.

Steps For Execution:

Using Google Colab (Recommended because fast and simple):

1. Open https://colab.research.google.com/drive/1dyNIUmj98xy5TSW9cgzzTv7mPyvSahpO#scrollTo=o86SxoacBs_z in your browser.
2. Upload your Style and Content Image by clicking the upload button in the console.

3. Change the name of content image from “night.jpg” to your Content image name (Second Cell)
4. Change the Name of Style image from “starry_night.jpg” to your style Image name (Third Cell)
5. Click on Runtime dropdown and chose run all button.
6. Scroll Down to the bottom of the page and see the result image

Note:
Google Colab is a free cloud service which now supports free GPU, Programmers use it to improve Python programming language coding skills. develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.

Using Terminal directly (Slow):

1. Extract the attached Zip1 file.
2. Change the Style and Content image file path as your desired images.
3. For Style change in line 21.
4. For Content change in line 16.
5. Run the code by using command “python StyleTransferAlgo1.py”.

Test Results:

I tested using *The Starry Night* painting by **Vincent van Gogh** (suggested) as style image and an Picture of a valley in a hilly area as content image.

Style Image:
https://artsandculture.google.com/asset/the-starry-night/bgEuwDxel93-Pg?hl=en-GB&avm=2

Content Image:
https://www.flickr.com/photos/jeffdai/24778020652/

Result Image:

Style transfer on Images using "Justin Johnson"s paper on "Perceptual Losses for Real-Time Style Transfer and Super Resolution "

In this approach we used a FeedForward network with Perceptual Loss function and by optimizing this perceptual loss we will be making the produced image better.
"Perceptual Loss" is the loss in order of higher level features of the image instead of pixel by pixel loss.

Link to Research paper:
https://cs.stanford.edu/people/jcjohns/papers/eccv16/JohnsonECCV16.pdf

I have implemented this approach for rendering video frames also so, we can also apply style transfer on live video captured from the webcams.

Major Advantage:

The overall process is very fast and takes hardly 1 second for a single image thus creating real time applications is possible using this approach.

Major Disadvantage:

This process will need the model to be trained on any new style image if that is requited , this process is a bit tedious but after this the weights can be directly used and the process is accelerated in comparison to previous process.

This approach is much suitable for developing applications like Prisma app as the no of style images there will be fixed.

Steps For Execution (for single static image):

1. Open Zip2 and extract all the files
2. Install all dependencies using requirement.txt file
3. Open styleTransfer.py file
4. Change the content image path as per your desire in line no 8
5. Change the style model path if required (by default **Starry Night** image by **Vincent Van Gough** is used) in line no 6 (few models of popular images are there available in Models folder.**(Skippable Step)**
6. Run the program using "Python styleTransfer.py" Command in the terminal

Steps For Execution (for WebCam / Video):

1. Open Zip2 and extract all the files
2. Install all dependencies using requirement.txt file
3. Open styleTransferOnVideos.py file
4. Change the style model path if required (by default **Starry Night** image by **Vincent Van Gough** is used) in line no 8 (few models of popular images are there available in Models folder.**(Skippable Step)**
5. Run the program using “Python styleTransferOnVideos.py” Command in the terminal
6. Style transferred webcam window will appear on the screen.

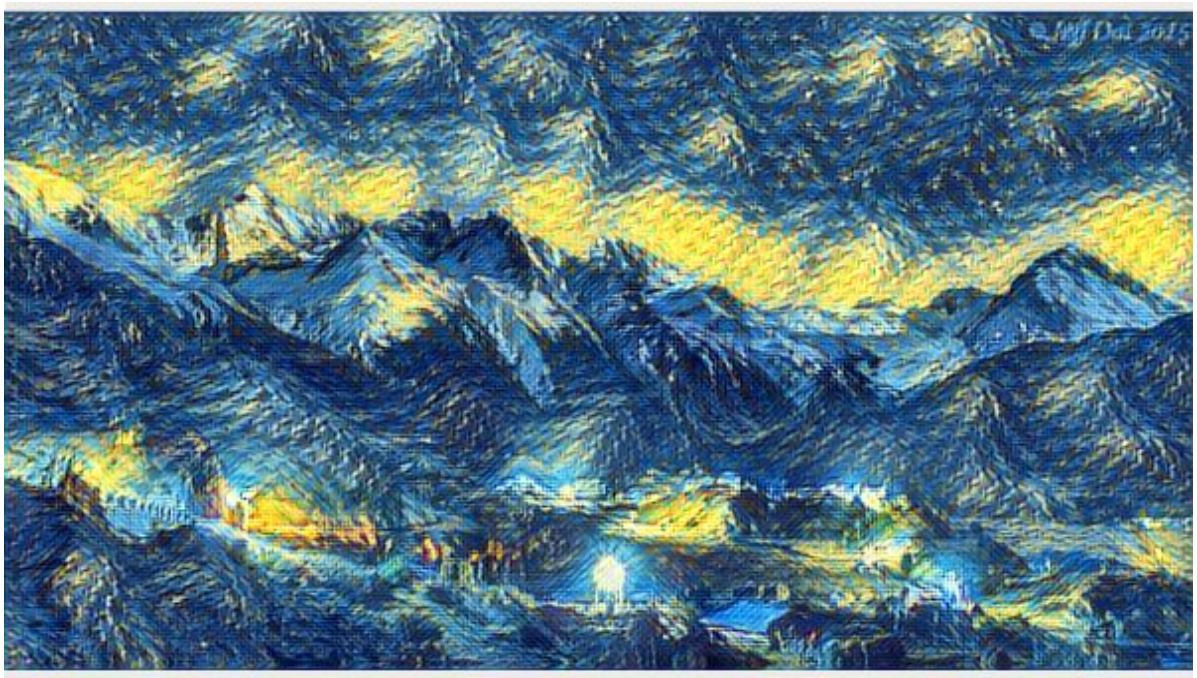
Test Results:

I tested using *The Starry Night* painting by **Vincent van Gogh** (suggested) as style image and an Picture of a valley in a hilly area as content image.

Style Image:
https://artsandculture.google.com/asset/the-starry-night/bgEuwDxel93-Pg?hl=en-GB&avm=2

Content Image:
https://www.flickr.com/photos/jeffdai/24778020652/

Result:



Summary:

In my point of view both the approaches have their own advantages and disadvantages like the first one is slow but will directly work for all images where as second one is fast but it is not directly fessible with any image.
It will surely depend on the type of application that is being developed that which algorithm should be used.
In my personal opinion i like the second approach more as it is simple and fast and gives real-time result.