PAINTING ASSISTANT

The initial goal of the project was to develop a model that could decompose a painting into a series of layers, so that through this decomposition, it could make proposals on how to approach the creation of that painting.

Setting aside the artistic part and turning it into a methodical and structured process, with the weakness of losing the artist's spontaneity but as a conforted tool for people who need guidance. It would be a helpful tool (or support) that would lead to a right outcome, leaving frustrations aside.

With that objective in mind, painting suggester, https://github.com/conchiVB/Painting_suggester, was created. This project consisted on the following stages:

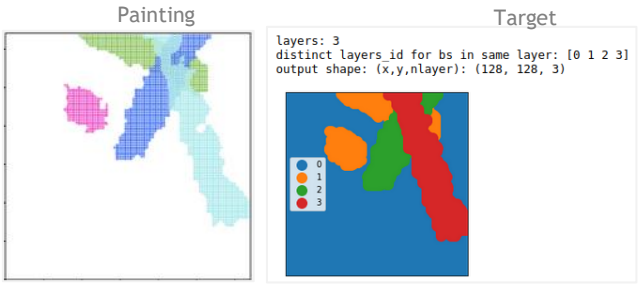
- Generate an autonomous database, so the volume would not be a problem.
- Train a model/s
- Deployment

To generate the paintings database, the following needs were required:

- Produce random brush strokes on a frame and
- Register the layer (deep) of the brush stroke in the painting

This first stage was concluded so complete paintings could be produced on demand.

And also simple paintings, in order to test a potencial model.



Training the model/s has been a difficult task. The model was generated from scratch and throughout the process, different needs have resulted in a significant amount of time being spent. The results have also not been fully satisfactory, so I made the decision to put the model on hold and dedicate the time to carry out a project for which the outcome would be more promising.

I would be happy to share the progress and difficulties encountered, to answer many questions and above all, to learn a little more about the process.

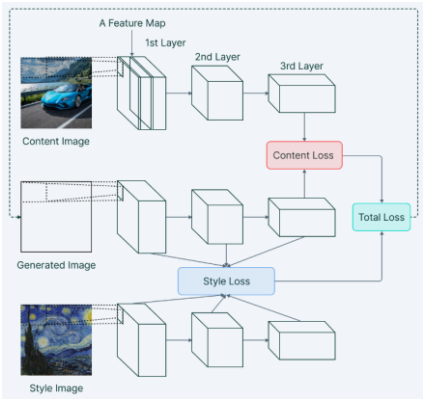
In order to work on a project related to the initial theme, although not the project I initially had in mind, I decided to work on style transfer as it offered me the opportunity to learn a little about transfer learning.

NEURAL STYLE TRANSFER

As you all probably know, Style Transfer is a technique in computer vision and deep learning that allows the transfer of the style of one image to the content of another image.

The goal of this project is to generate a new image that retains the content of the original image and has the style of the reference image, where the style reference image is a set of brush strokes, thus transforming the image into a painting-like appearance.

VGG19 is a popular convolutional neural network architecture that was trained on the ImageNet dataset. It has 19 layers and is one of the deeper models developed for image classification.



In the style transfer project using VGG19, the network is used to extract feature representations from both the content image and the style reference image. These feature representations are then combined to generate a new image that retains the content of the original image and incorporates the style of the reference image.

Although this project is part of the main project, it has been separated into a dedicated repository in order to have a clear and organized view of it, https://github.com/conchiVB/Painting_with_Neural_Style_Transfer_aid.

The loss calculation is computed as a weighted sum of content loss and style loss.

The content loss is measured as the Mean Squared Error (MSE) between the feature representations of the generated image and those of the content image, using the "block5_conv1" layer.

The style loss is calculated based on the difference of the GRAM matrices between the feature representations of the style image and the generated image across multiple layers: "block1_conv2", "block2_conv2", "block3_conv3", "block4_conv3", and "block5_conv1".

The model was trained on Google Colab, as it was necessary to have access to more powerful computing units. Although the model has been tested on different styles, the priority was always to have the model work well for the desired style, brush strokes.

The default style image is

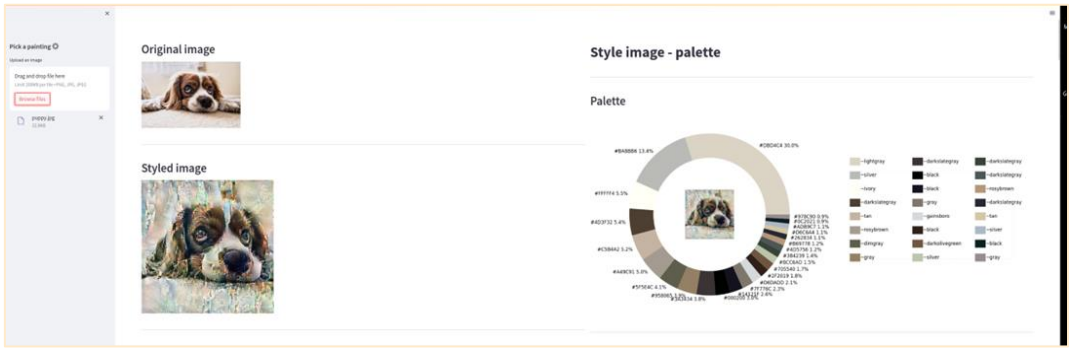
The model performs well on 2000 epochs and is how it has been set in the final visualitation (streamlit), so it can take a couple of minutes, depending on the machine.

Some of the results can be found in the appendix.

VISUALITATION

In order to link this project to the original goal, which was one of my biggest motivations, a couple of algorithms related to the initial objective on painting an image have been added.

The visualitation has been developed in Streamlit.



The client uploads an image, which is then processed by the style transfer model to generate a new image. The generated image is then segmented by color and the resulting palette is displayed.

Finally, a series of steps to paint the image are displayed in the form of images. This breakdown is based on color segmentation.

INSTALLATION GUIDE

- git clone https://github.com/conchiVB/Painting_with_Neural_Style_Transfer_aid.git
- Upload repository to Google Drive
- Access Google Colab with an account: https://www.datahack.es/google-colab-for-data-science/

Developers - Training the model

From Google drive, open notebook 4_Neural_style_transfer.ipynb.

To replicate the model results with the same settings, run the complete notebook.

For any changes on the settings:

- Content and style images; look for the next lines in the code:
  - CONTENT_IMAGE_URL = path_content + 'puppy.jpg'
  - STYLE_IMAGE_URL = path_style + 'paint1.jpg'
- Style and content layers
  - Style-layer weights: weight1,weight2,weight3,weight4,weight5= 0.1, 0.3, 0.4, 0.1, 0.1
  - STYLE_LAYERS = [('block1_conv2', weight1),('block2_conv2', weight2),('block3_conv3', weight3),('block4_conv3', weight4), ('block5_conv1', weight5)]
  - Content layer in the call to the model; *LOAD_VGG19('BLOCK5_CONV1')*
- Remaining parameters:
  - LEARNING_RATE = 0.03 #0.05
  - ALPHA = 1
  - BETA = 0.3
  - EPOCHS = 2000
  - ADD_NOISE = True
  - NOISE_RANGE = 0.2

Client - Visualitation

From Google colab, open streamlit_tfm.py and run the complete code.

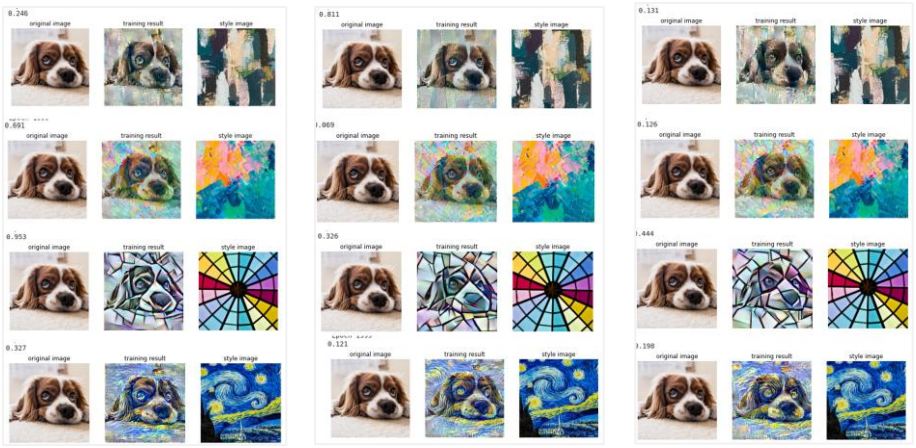
The last line !npx localtunnel --port 8501 opens an url, follow the link.

ANNEX

The chosen set of parameters was selected based on the results of several trials and optimizations.

Selected parameters: it works well with the brush stroke style and strong patterns are smoothly integrated in this option

Some of the discarded options;



Left: borders of generated image is failing when style has strong patterns.

Middle: works well but include not expected effects in strong patterns, and losses content definition

Right: in strong patterns is capturing too mach effect from style image