Neural Artistic Style Transfer

M R Abhishek
K Vagdevi
Prof. Shiv Ram Dubey

Goal

- Neural Artistic Style Transfer finds a wide range of applications to fancily modify images.
- When a content image and a style image are given as inputs, the output image is expected to contain the content image in the artistic style of the input style image.
- We implement these visual effects using the convolutional neural network.

Uses

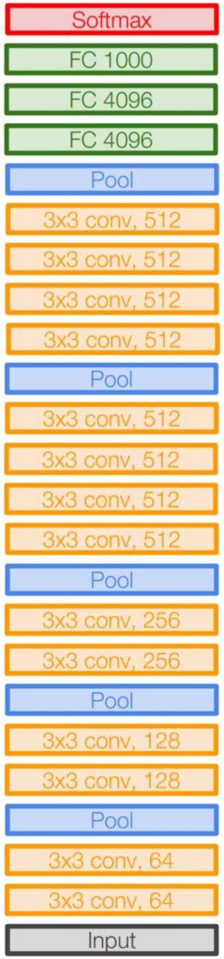
- This field has so much influenced the technical world that many apps, such as Prisma, have received great craze amongst the users.
- In the recent days, a decent work has also been done in this area, which served as a holy grail to our project.

How it is different from other deep learning methods

- In general, we will have a lot of data and we train the weights by minimizing the loss function.
- Unlike other deep learning methods, here we use pretrained weights which was trained on millions of images and update the pixel values to get stylized image.
- This is not one time training. We need to train the pixels every time to get stylized image for different content and style images.

Pretrained weights

- We used the pretrained weights of the VGG-19 model on the famous Imagenet database.
- We can use pretrained weights of any model that were trained on a large image database.



VGG19

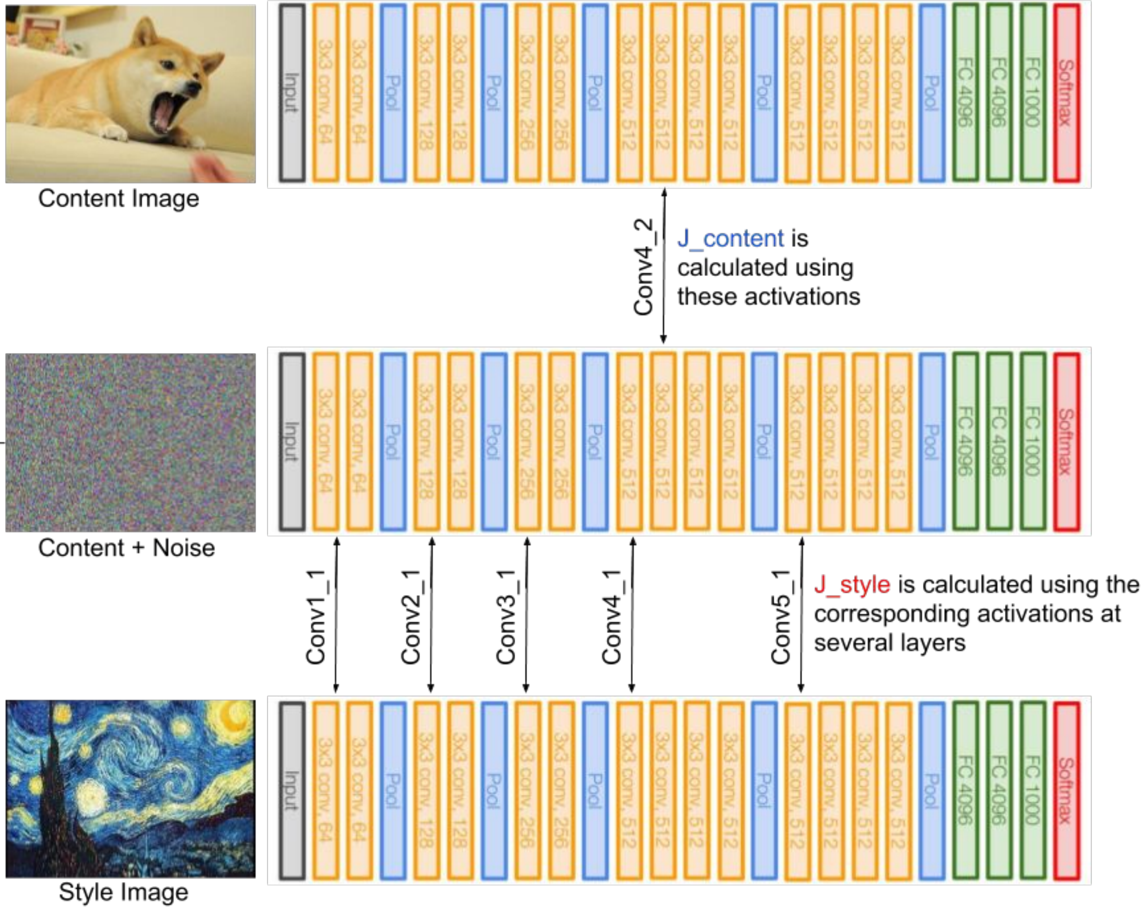
Workflow

Neural Artistic Style Transfer

- The pixels of the generated image are updated after every iteration by reducing the total cost.
- This updation is done till the stylized image is achieved.

Stylized Content Image
(Expected Output)

After updations



J_total = (alpha*J_content) + (beta*J_style)

Workflow - Cont’d

- The generated image should contain content of content image and style of style image.
- So, we need to reduce two cost functions combinedly, one for content and other for style.

J_total = alpha*J_content + beta*J_style

- J_content cost is to make sure that the content in the content image remain same in the generated image.
- J_style cost is to style the generated image using the style in the style image.

J_content

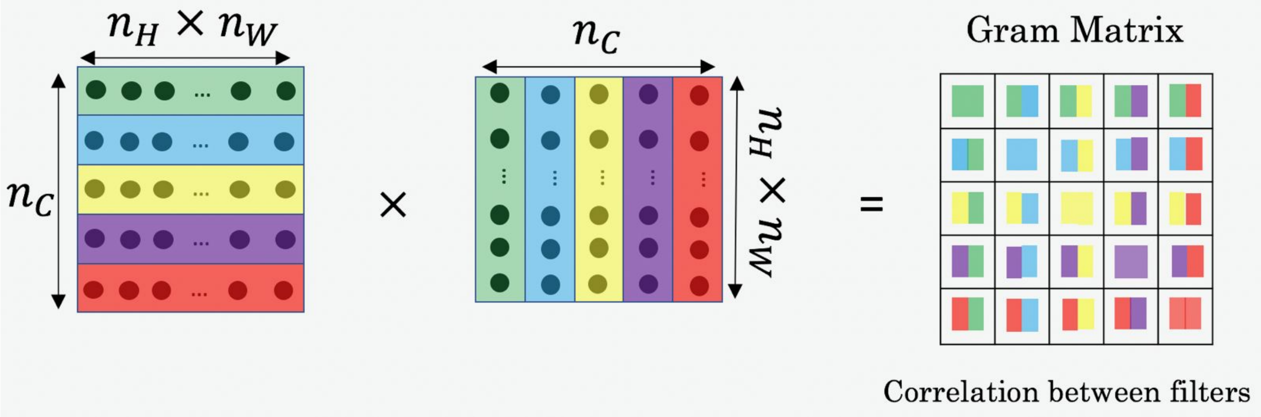
- We use activations of only one layer for content cost. We can also use more than one layer.
- The content cost is the mean square error between the activation layers.

J_style

- For style cost, we use activations at five different layers as to impart style completely into the generated image.
- After training for around 3000 - 5000 iterations, we get satisfactory results.

J_style - Cont'd

- The style loss, J style , could be calculated using the Gram Matrices at various layers considered for capturing the style of the style image.



- The style cost is the sum of mean square error between the Gram matrices at different layers.

Experiment results

Content image

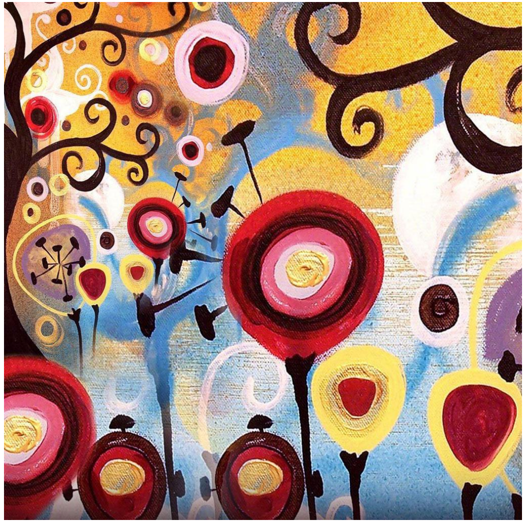
Style image

Generated Image

Experiment results

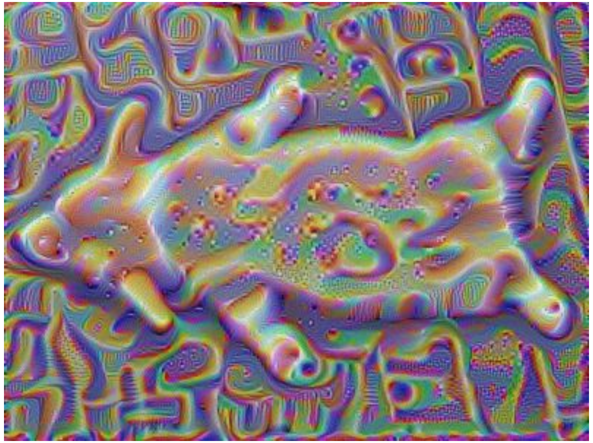
Content image

Style image

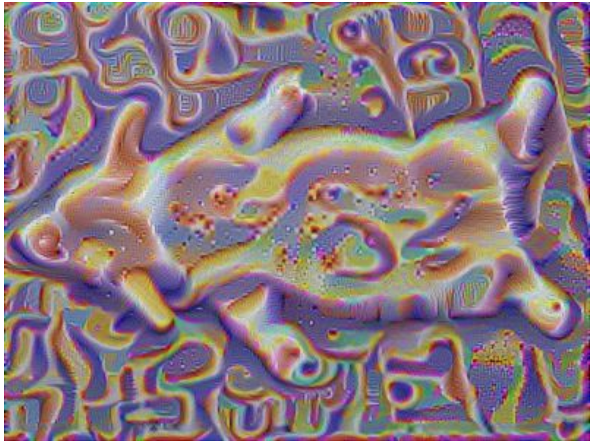


Experiment results

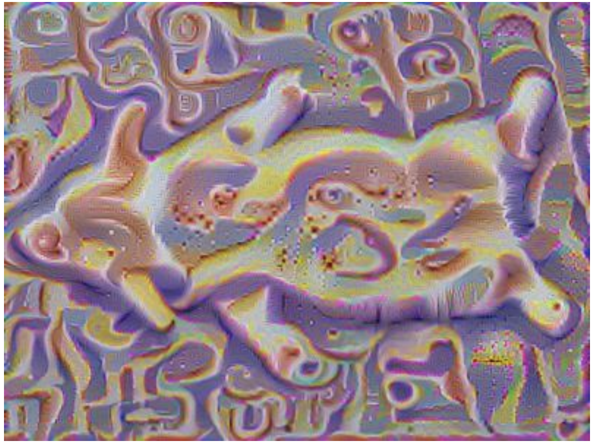
Iteration: 500



Iteration: 1000



Iteration: 2000



Challenges

- One of the challenges we faced was with the learning rate. We tried with multiple number of learning rates.
- After a lot of experiments, we were able to get good stylized image with a learning rate of 0.1.
- If we use 0.01, the image is not getting updated quickly. It is taking a lot of time and we are not able to see much change in the generated image even after 2000 iterations.
- If we use 1, the pixels in the image are getting updated very quickly and we are losing the content in that image.