How Machines learned Art

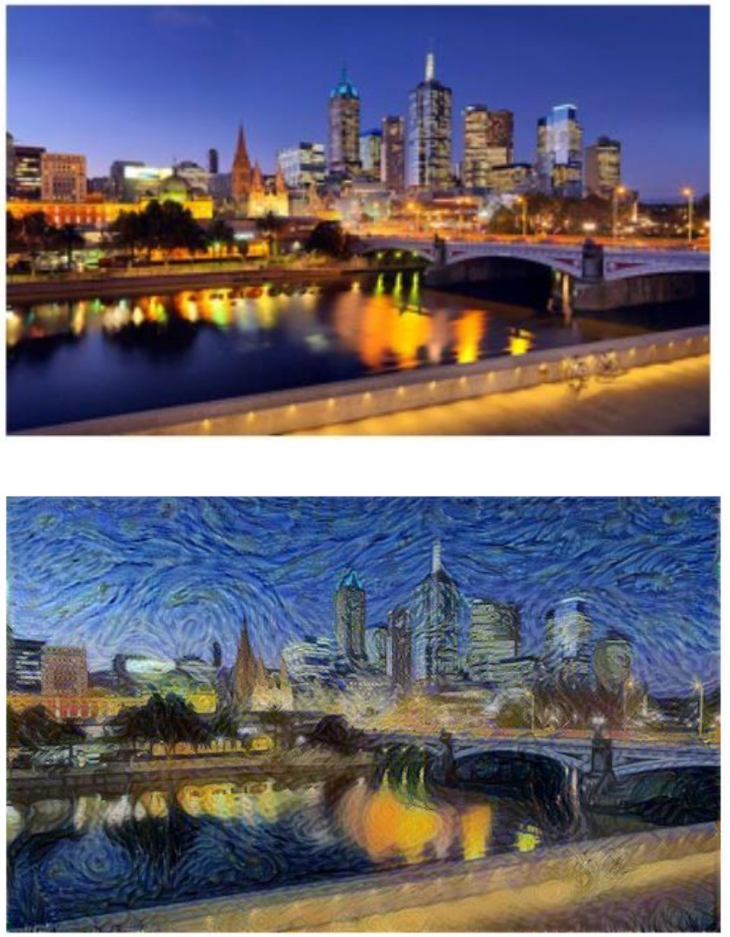
Harsha Vardhini Vasu
hxv190005@utdallas.edu

Motivation

- Neural networks can outperform humans in many tasks like weather prediction, voice recognition, etc... given enough data

- But there a few tasks that are challenging for a machine to learn. One such task is Neural Style Transfer (NST), As it has to do more with creativity than numbers

- Let's see how machines learn to do creative artworks



Picture (a) input Image
Picture (b) Style transferred output

http://www.subsubroutine.com/sub-subroutine/2016/11/12/painting-like-van-gogh-with-convolutional-neural-networks

Previous Work

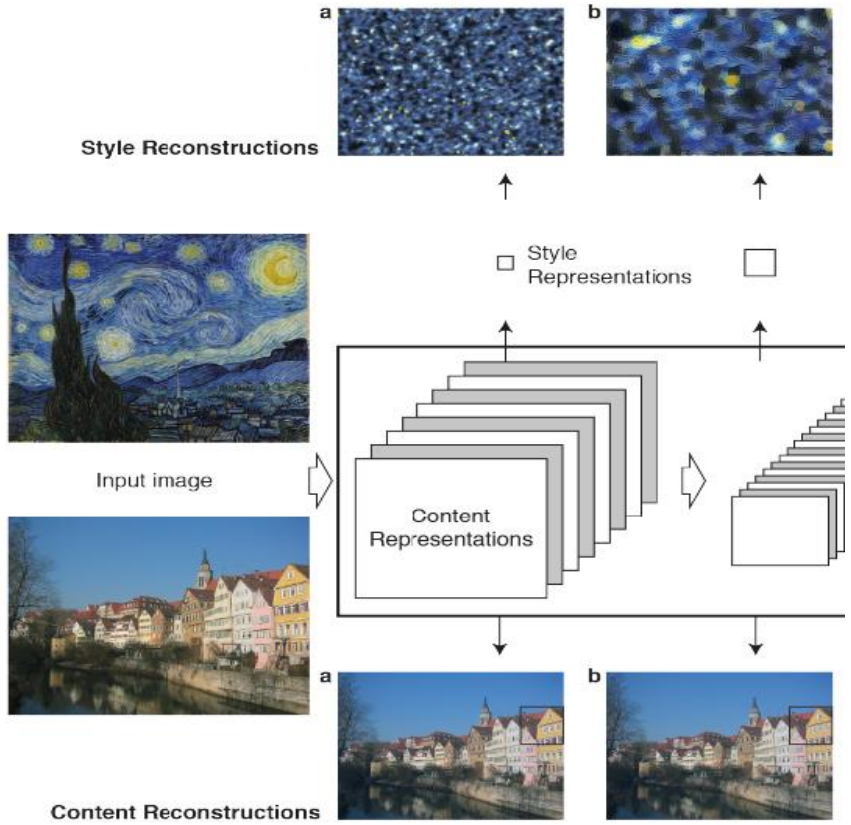
- The idea of style transfer initially came from the concept of texture synthesis.
- **Texture synthesis** is a process of growing a new image from initial seed to arbitrarily large size. Initially, it was done one pixel at a time[1], then the concept was extended to one patch at a time.
- **Texture Transfer** was implemented using the idea of texture synthesis. It is a process of rendering an object with the texture of a different object[2]. Later its performance was improved by using EM Algorithm
- **Style Transfer** was initially achieved by local texture transfer and global color transfer. This idea was improved later by making sure that the synthesized image is close to the content image features extracted by the VGG-19 classifier

Texture synthesis on surfaces

https://graphics.stanford.edu/~mdfisher/TextureSynthesis.html

Content & Style Representation

- Content Representation
  - Each layer in CNN can be seen as a filter, and its output can be seen as a filtered version of the input image
  - On this process of filtering with multiple layers, the input image is transformed into a representation that cares more about the actual content of the image
- Style Representation
  - Correlation between different filter responses over the spatial dimension of the output of each layer in CNN captures the style information or the texture of the image



Picture (a) on top is the style extracted from the artwork by the first layer in the CNN
Picture (b) in the bottom is the content extracted from the input image by the first layer in the CNN

Reference [3]

Per-Pixel loss & Its improvement

- The Euclidian distance between output and target image. It can be used only if the dimensions of both images match.
- Commonly used for image transformation tasks

- Problem with this approach is that it cannot recognize the similarity between an image and the same image shifted one pixel, as there is a lot of difference in the pixel level although the content of the image is exactly the same

- SOLUTION: Perceptual Loss

- Apart from solving this issue, using perceptual loss also generates high-quality images and runs three times faster

Picture (a) Style transfer using per-pixel loss
Picture (b) style transfer using perceptual loss

Reference [4]

Perceptual loss

- Unlike per-pixel loss, Perceptual measures the high-level perceptual difference between images
- They make use of loss network pre-trained for image classification task

- For generating visually pleasing results in style transfer, the semantic information in the input image must be reflected in the output image despite the drastic change in color and texture
- Using perceptual loss allows the transfer of semantic information from the loss network to the style transfer model

- Style transfer model uses 2 perceptual loss functions
  - Content loss function
  - Style loss function

Content Loss

- Content loss is computed as the Euclidean between the high-level features extracted by the pre-trained CNN classifier

$$\mathcal{L}_c(p) = \sum_{j \in \mathcal{C}} \frac{1}{U_j} \mid\mid \phi_j(p) - \phi_j(c) \mid\mid_2^2$$

- Φj(p) – output of activation function of layer j for input image p
- Φj(c) – output of activation function of layer j for generated image c
- Uj – Total number of units in layer j

Style Loss

- Style loss is computed as the Forbenius norm between the Gram matrices computed from the lower-level features extracted by the pre-trained CNN

$$\mathcal{L}_s(p) = \sum_{i \in \mathcal{S}} \frac{1}{U_i} \mid\mid G(\phi_i(p)) - G(\phi_i(s)) \mid\mid_F^2$$

- G(Φi(p)) – Gram matrix associated with output of activation function of layer i for input image p
- G(Φi(c)) – Gram matrix associated with output of activation function of layer i for generated image c
- Ui – Total number of units in layer i

Loss Function

- The loss function minimized at each step of gradient descend contains components of both content and style

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

- Alpha and beta in this equation is the weight of content loss and style loss respectively
- P is the input image
- A is the input artwork
- X is the generated image

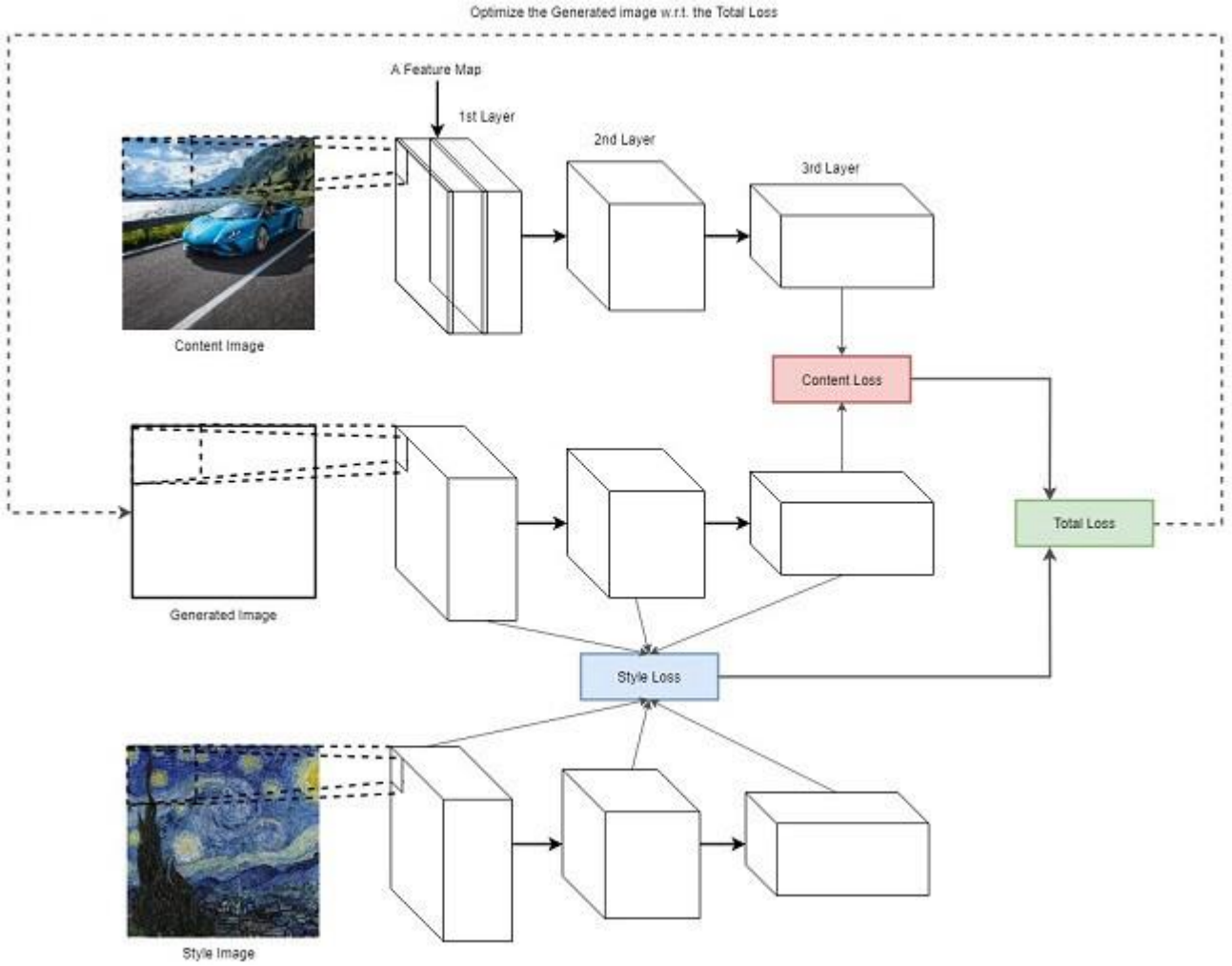
Trade off

- Output image is generated by taking global arrangement of objects / higher-level features of the input image and the color, texture, and local structure like the fine details and strokes from the painting.
- Giving more weightage to content loss will result in an output image that is very similar to the content of the input image but doesn't reflect the style of the painting.
- Giving more weightage to the style component will result in an output image that matches the appearance of the artwork. But, the content from the input image will not be obviously visible.
- TRADE OFF IS IMPORTANT

Picture (a) shows the output of giving more weight to style. Picture (b) shows the result of giving correct weights for both content and style components

Neural Style Transfer Model

- The algorithm starts with a white noise image
- In every iteration of gradient descent:
  - it makes the content of the image more similar to the input image, trying to reduce the content loss
  - It makes the style of the image more similar to the input artwork trying to reduce the style loss



If you copy a figure from somewhere else add a link here to the original source

Problem with single purpose nature of NST

- Style transfer algorithms are widely used in the mobile application. If we have to train a model for each style of painting/artwork. It will consume a large amount of memory
- Building separate model ignores the common visual features (like strokes) in different painting and relearns it from scratch for each painting
- A model that captures artistic style must be able to export and learn from those patterns
- SOLUTION: Conditional Style Transfer network for N styles

Conditional Style Transfer network for N styles

- The difference between single-purpose style transfer model and N-Style style transfer network is the **Conditional Instance Normalization**

$$z = \gamma_s \left(\frac{x - \mu}{\sigma} \right) + \beta_s$$

- Conditional Instance Normalization transforms a layer's activations x into a normalized activation z specific to a particular painting style s

- According to this approach, all convolutional weights of a style transfer network can be shared across many styles. The only difference between the styles are the parameters of the affine transformation γ and β applied to the activation

Conditional Style Transfer network for N styles

- This model works even with very diverse painting styles, not just similar ones.
- It is also easy to add a new style to the model and much faster than training a single purpose model from scratch for the new style, as all the convolutional weights are shared, the model has to learn only the γ and β values for the new style
- Using only γ and β values to represent a style is like creating an embedding for styles
- So, it is possible to blend painting styles to generate images that have 2 or more styles

Blending styles

- Instead of using single γ and β values to represent style, if combination γ and β values from 2 different images were used, generated images will have 2 styles.

- Convex combination of γ and β values can be used to produce a smooth transition of images from one style to another

$$\gamma = \alpha * \gamma 1 + (1 - \alpha) * \gamma 2$$
$$\beta = \alpha * \beta 1 + (1 - \alpha) * \beta 2$$

Reference [5]

References

- Texture Synthesis by Non-parametric Sampling
  - https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/papers/efros-iccv99.pdf
- Image quilting for texture synthesis and transfer
  - https://www.semanticscholar.org/paper/Image-quilting-for-texture-synthesis-and-transfer-Efros-Freeman/dd7bf950093fc65f3ae6ad79666ce1077f9dfb2e
- A Neural Algorithm of Artistic Style
  - https://arxiv.org/abs/1508.06576
- Perceptual Losses for Real-Time Style Transfer and Super-Resolution
  - https://arxiv.org/abs/1603.08155
- A Learned Representation For Artistic Style
  - https://arxiv.org/abs/1610.07629