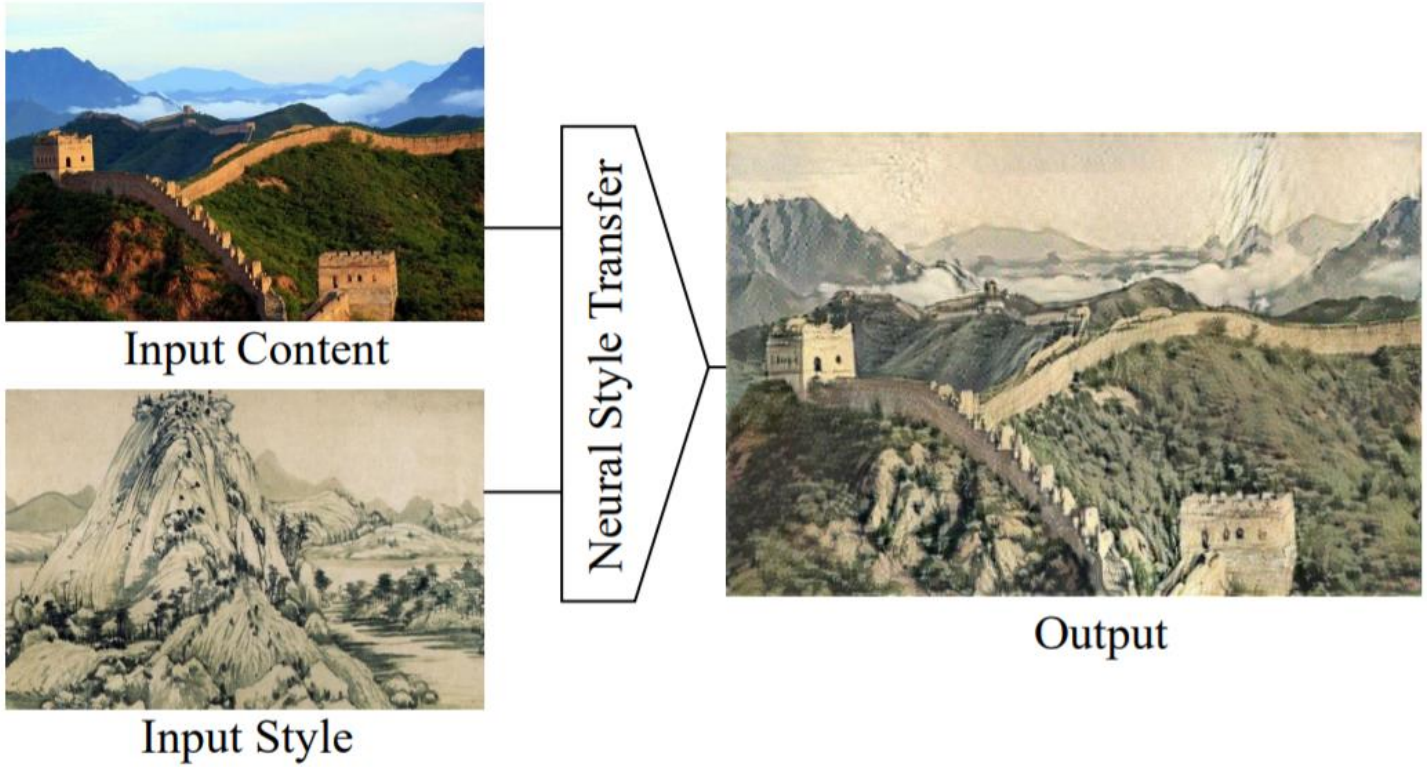
A Tutorial on Neural Style Transfer

Savan Visalpara
sxv180069@utdallas.edu

Motivation

- Humans have the skills to create breathtaking paintings. But what if computers can create such art on their own?
- Or ever wished about changing a photo to something that looks like it's painted by Picasso or van Gogh? **Style transfer** can help.



*Style transfer is a technique of generating an **output image** given an **input image** and a **reference style image** such that output image has the core content from the input image and the style from the reference style image.*

Figure from https://arxiv.org/pdf/1705.04058.pdf

Previous Work

- Style transfer is being studied for over two decades under the field of **non-photorealistic rendering**.
- Earlier methods involved pure computer vision techniques such as placing strokes (i.e brush, tiles) on a photo.
- The first popular machine learning based approach, image analogies, was proposed by Hertzmann et al. [1] in 2001, where they learn mapping from A to A' and apply that learned mapping to B in order to get output image B' as shown below.

Figure from [1] : https://dl.acm.org/doi/pdf/10.1145/383259.383295

Limitation of Previous Approaches

- Training data (pair of input and output (i.e artistic) image) unavailable in practice.
- Learned mapping had more low-level image features which performed poorly at capturing style and content.
- Unsatisfying output images.

Limitation of Previous Approaches

- Training data (pair of input and output (i.e artistic) image) unavailable in practice.
- Learned features were low-level image features which performed poorly at capturing style and content.
  - **Solution**: Use **CNN** as they learn low-level (early layers) and high-level features (deep layers).
- Unsatisfying output images.
  - **Solution**: If above two issues can be solved then this is automatically resolved.

Ok, then how to avoid the requirement of having **pairs**?

Neural Style Transfers

- The Curse of Pairs and the Blessings of CNNs
  - A pair (input image and artistic output image) is required to assess how good the generated image is compared to the given output/target image.
  - To solve this, Gatys et al. [2] reformulated the problem itself by leveraging the feature representational power of CNNs.
  - Instead of solving for input-output mapping, they synthesize output image directly by iterative optimization.
  - To quantify the quality of the output image, they pass these images through a pre-trained CNN network (i.e VGG19) and compare some intermediate feature maps of current output image to respective feature maps of original input image and reference style image. The goal is to make these feature maps similar so that output image has both content from original image and style from reference style image.

[2] : https://arxiv.org/pdf/1508.06576.pdf

Neural Style Transfer

- Algorithm

1. Start with the input image.
2. Iteratively **optimize image itself** (vs mapping from input to output) such that final output has core content from the input image and style from the reference style image. Thus, **no need for pairs**. Compute loss to optimize following below steps 3-5.
3. Pass current output image through a pre-trained CNN (i.e VGG19) and save some intermediate feature maps.
4. Pass original input and reference style image through the same network and save corresponding feature maps.
5. **Compare** feature maps of **current output and original input image**, and **current output and reference style image**. Compute the difference (i.e loss), then run optimization to minimize loss (these differences).

Note that there is **NO TRAINING** involved here. We just use pre-trained CNN.

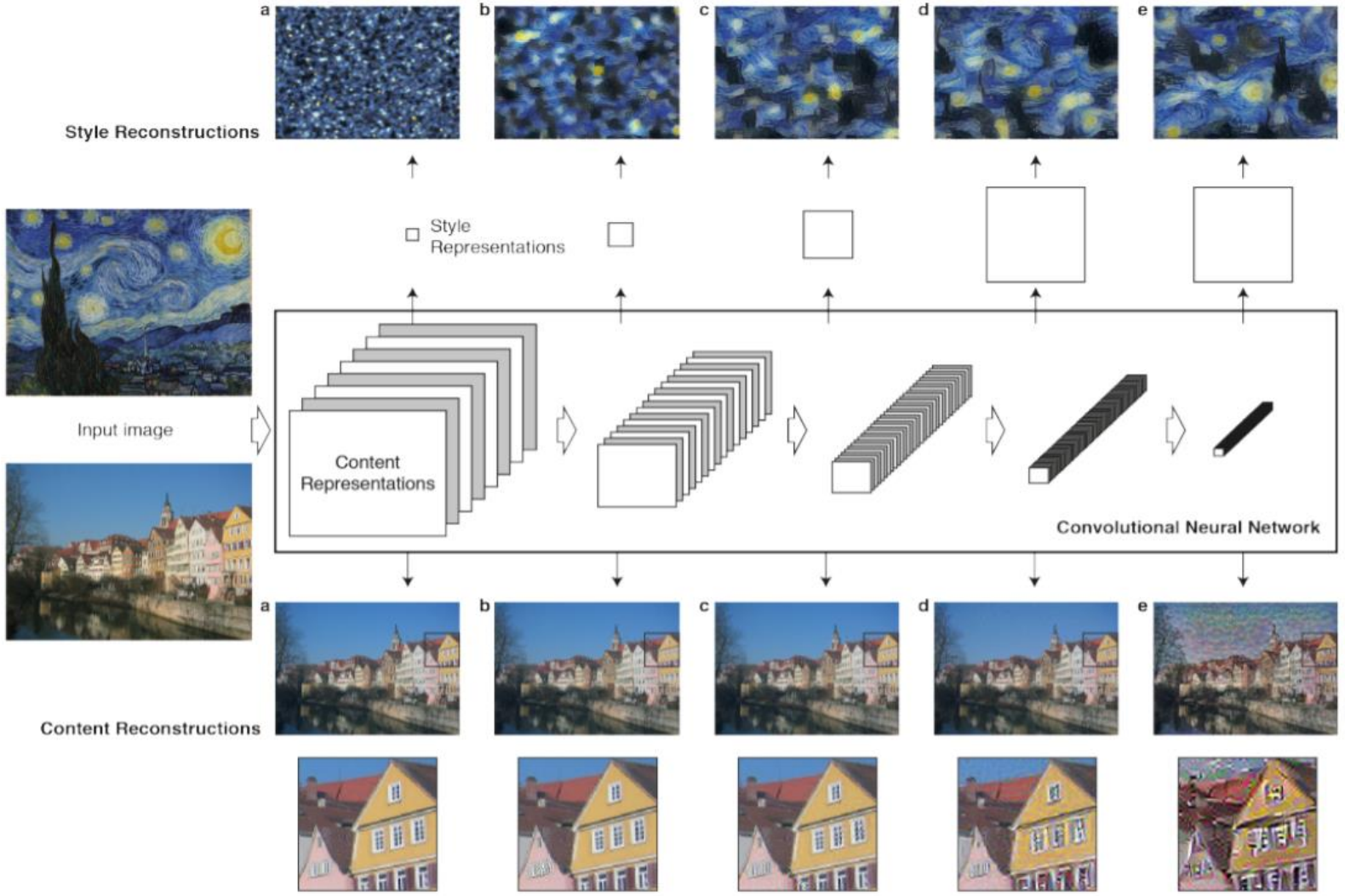
A Neural Algorithm of Artistic Style : https://arxiv.org/pdf/1508.06576.pdf

Neural Style Transfer

- Why intermediate layers and which intermediate layers to compare?
  - It is well-established that CNNs are very good at capturing low-level (early layers) and high-level (deep layers) features.
  - We also know that if two images have **similar content** (i.e objects), **higher-level features will be similar**. Thus, we can use these higher-level feature maps in content loss to evaluate how well the output image contains the original content image.
  - Similarly, **early layers capture low-level features** such as lines that are critical part of a painting. Hence, these can be used in style loss to evaluate how well the output image contains style from a given reference image.
  - In the original paper, authors used **conv4_2** (i.e deep) layer for content comparison and **conv1_1**, **conv2_1**, **conv3_1**, **conv4_1**, and **conv5_1** of VGG19 for the style comparison.

- **IMPORTANT**: This way of computing loss (i.e difference between feature maps) is used in all methods described in this tutorial.

Neural Style Transfer



Notice as we go deeper content reconstruction loses the low-level details but preserves the core content from the original image.

Similarly, early layers gave better low-level style reconstruction.

Neural Style Transfer

- Loss Functions
  - Content loss: $$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$
    - Here, F^l represents the feature map of current output image at layer l
      P^l represents the feature map of original image at layer l
    - **Intuitively** this quantifies how well the current output image contains the core content from the original input image by measuring the difference between high-level feature maps of some intermediate layers.
    - We use higher-level features as these higher-level feature maps represent the core content (i.e objects) from the original image.

Neural Style Transfer

- Loss Functions
  - Style loss:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l} \qquad E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l}\right)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

- Here, F^l represents the feature map of current output image at layer l
  A^l represents the feature map of original image at layer l
  G^l is a gram matrix which represent the correlation between filters
  w_l represents the weight for the layer l
  N and M represent number of feature maps and feature map size respectively
- Similar intuition applies here but here we care about low-level features (i.e underlying style).

Neural Style Transfer

- Final Loss Function

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

  - Here, α and β are weights for content and style loss.

- Generating Output Image
  - To start, consider the original image as a current output image.
  - Iteratively optimize above loss function using gradient descent such that the loss is minimum.
  - Ideally, when the loss is minimum, the output image contains core content from the original image and style from the reference style image.
  - There **is no training phase**. We optimize output image at test time.

Neural Style Transfer

- Sample Output

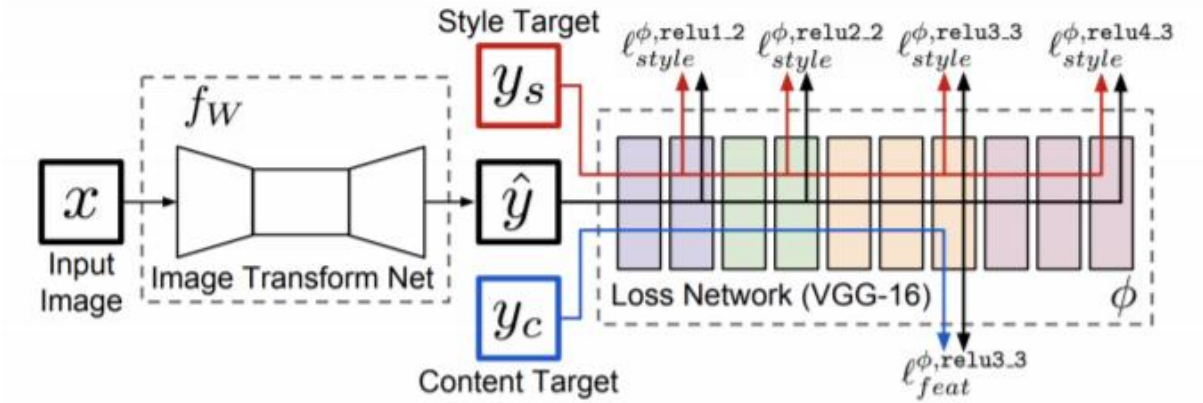
Input Image

Reference Style Image

Output Image

Real-Time Style Transfer

- Previous approach is very inefficient. Takes ~3-4 minutes to generate output image **every time.**
- Johnson et al. [3] proposed training a CNN in supervised learning setting to overcome inefficiency.



- Very similar to previous approach except they train a model (i.e CNN that can reconstruct an image) that generates the output (vs direct optimization). During training, this model's output along with reference style image and content image (i.e input image) are used to compute loss by comparing intermediate feature maps of (i.e) VGG16 just like previous approach.

[3] : https://arxiv.org/pdf/1603.08155.pdf

Real-Time Style Transfer

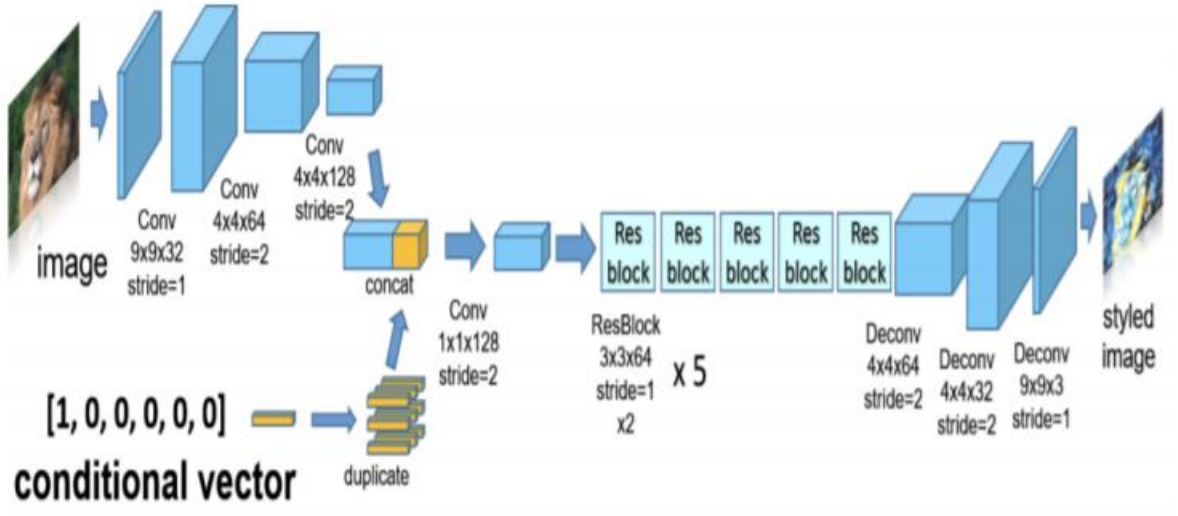
- Generating Output

- Left image produced by Gatys et al. [2]'s method.
- Right image produced by Johnson et al. [3]'s method.
- Notice that the left one has a lot more noise than the right one. This was also a problem with Gatys' method apart from being inefficient.

Figure from [3] : https://arxiv.org/pdf/1603.08155.pdf

Mix Style Transfer

- With previous approach, we need to train one model per style. That's not ideal.
- Yanai et al. [4] proposed a model trained on N styles that can generate N different styles.

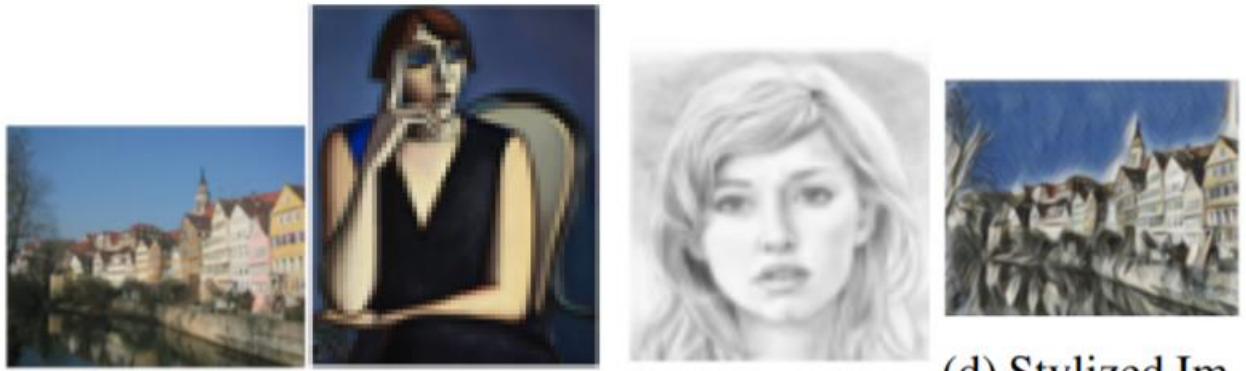


- Very similar to previous approach except instead of passing input image (along with style target) to pre-trained CNN for loss computation, we pass a N-dimensional conditional vector (i.e binary vector) concatenated with input image features. This binary vector tells the network which style to train for.
- For example, [1,0,0,0,0,0] tells it to use the first style. We can also mix multiple styles as follows: [0.2, 0.2, 0.1, 0.1, 0.3, 0.1].

[4] : https://dl.acm.org/doi/abs/10.1145/3078971.3079037

Mix Style Transfer

- Generating Output



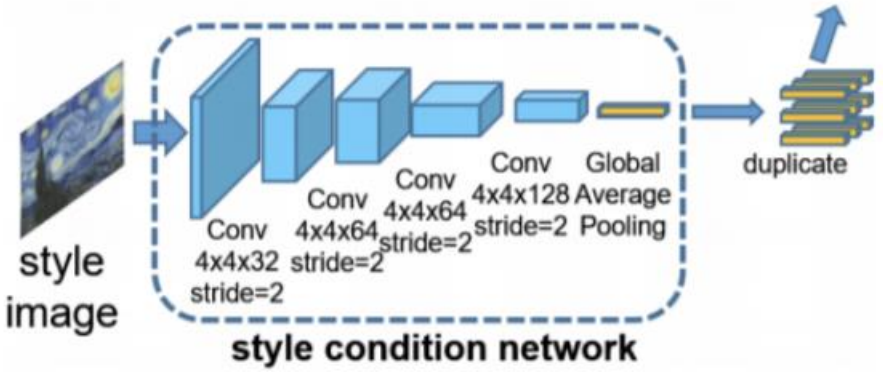
(a) Content Image (b) Style Image1 (c) Style Image2 (d) Stylized Image using Style 1 and 2

- At test time, we pass a binary vector to denote which style(s) to use.
- An example of how multiple styles, 2 in this case, can be combined to generate an output is shown above.

Figure from: A Literature Review of Neural Style Transfer https://bit.ly/3h93HCK

Arbitrary Style Transfer

- With previous approach, we can only generate styles that network has been trained on. The network took input image and a vector referring to a style as an input, but not reference style image (it was internally fetched using binary vector for training).
- Yanai [5] proposed an extension of the previous approach which takes input image and reference style image as an input during both training and testing. Hence, removing the constraint of network learning about fixed number of styles.

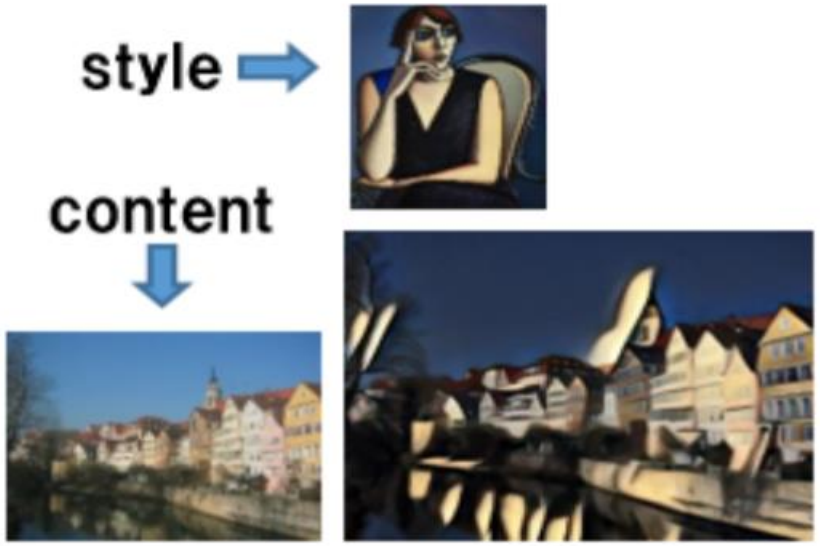


- Approach is very similar to previous one except instead of concatenating binary vector with features of input image, they first pass style image through a CNN to generate a feature vector. Then, this feature vector is concatenated with the input features. This network along with the network in the previous slide is trained together.

[5] : https://openreview.net/pdf?id=H1Y7-1HYg

Arbitrary Style Transfer

- Generating Output



  - At test time, we pass input image and a reference style image to the network which outputs a stylized output image which has content from the original image and style from the reference style image.

Figure from : https://openreview.net/pdf?id=H1Y7-1HYg

References

- [1] Image Analogies
  - https://dl.acm.org/doi/pdf/10.1145/383259.383295
- [2] A Neural Algorithm of Artistic Style
  - https://arxiv.org/pdf/1508.06576.pdf
- [3] Perceptual Losses for Real-Time Style Transfer and Super-Resolution
  - https://arxiv.org/pdf/1603.08155.pdf
- [4] Conditional Fast Style Transfer Network
  - https://dl.acm.org/doi/abs/10.1145/3078971.3079037
- [5] Unseen Style Transfer Based on a Conditional Fast Style Transfer Network
  - https://openreview.net/pdf?id=H1Y7-1HYg
- [6] A Literature Review of Neural Style Transfer
  - https://bit.ly/3o0mmC2
- [7] Neural Style Transfer: A Review
  - https://arxiv.org/pdf/1705.04058.pdf