When Van Gogh meets Monet: Art Style Transformation through Deep Convolutional Neural Network

Jiawei Wang

COMP 135 Machine Learning

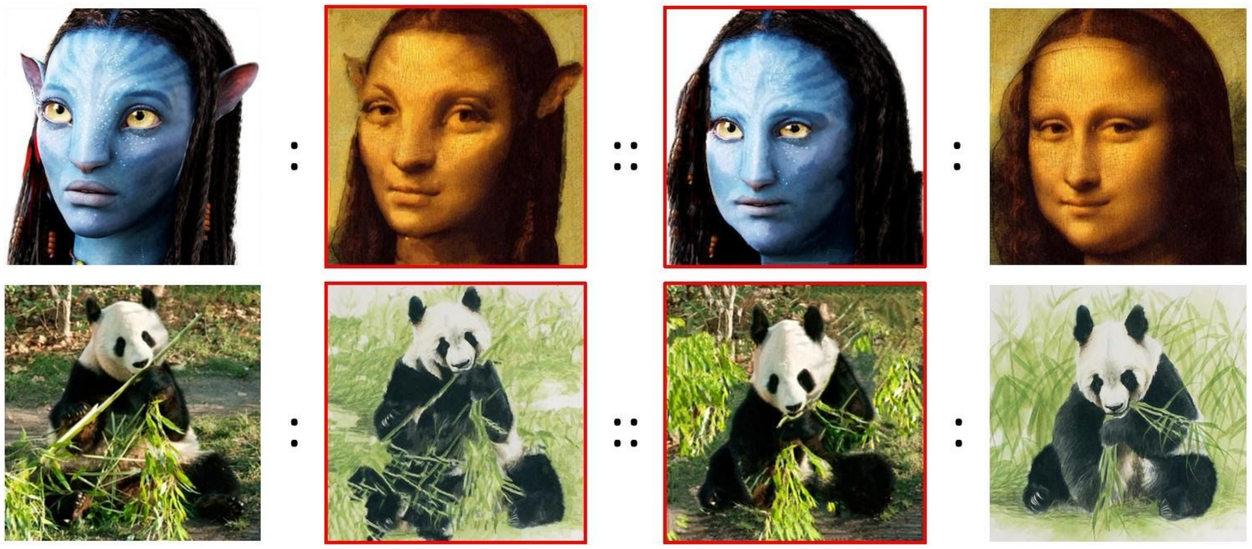


Figure 1 [1]

Deep Image Analogy is a Visual Attribute Transformation software co-developed by a group of researchers from Microsoft Asia Research Institute and students from Shanghai Jiaotong University. The first time I came into it was when browsing randomly on the internet and found a website where it was credited as the best machine learning software for art style transformation of 2017 by a media named MyBridge [2]. Although MyBridge's introduction of it is just one sentence without any highlights, its picture attracted my interest: the famous Mona Lisa, through machine learning, became an Avatar; a photo of a panda, was rendered as a colored pencil drawing (Figure 1). What they have in common is that the layout of the original pictures has not changed, only the style was changed to another form. This easily reminds me of an application that I used to have on my phone a few years ago -- Prisma [3]. In Prisma, the user can upload a picture or photo, with the filters it provides, Prisma can automatically convert the picture into another art style with all the computing done in the server. So naturally, this brings me two questions: 1. What ML algorithm is used by Prisma to modify the picture and 2. Since Prisma has already been launched as a successful commercial software in 2016 [4], what's the innovation of this 2017 Deep Image Analogy.

In order to answer the first question, I searched for related documents. The concept of Convolutional Neural Network (CNN) came into my sight [5]. Convolutional Neural Networks are composed of multiple layers of neurons, and each layer can be regarded as a kind of detection filter, used to detect whether a certain feature exists in the input data. Unlike traditional pre-designed filters, CNNs automatically learn based on data and task goals [6]. Take the human face as an example. The lower layer filters can detect some local and specific features. These features are generally simple, such as lines, or the shape of the eye. The higher layer filters can handle the global and abstract features, these features are more complex, such as facial expressions. The final output layer is responsible for gathering the previously detected features and making the final output [7].

In Prisma, CNNs also adopted the same strategy. The bottom layer is to extract the target image features. Most of the information are local textures or painting details. They form a huge feature map. At this stage, we can already obtain a relatively complete reconstruction image. And the upper layer is mainly used to further abstract the input and adjust the style of the picture.

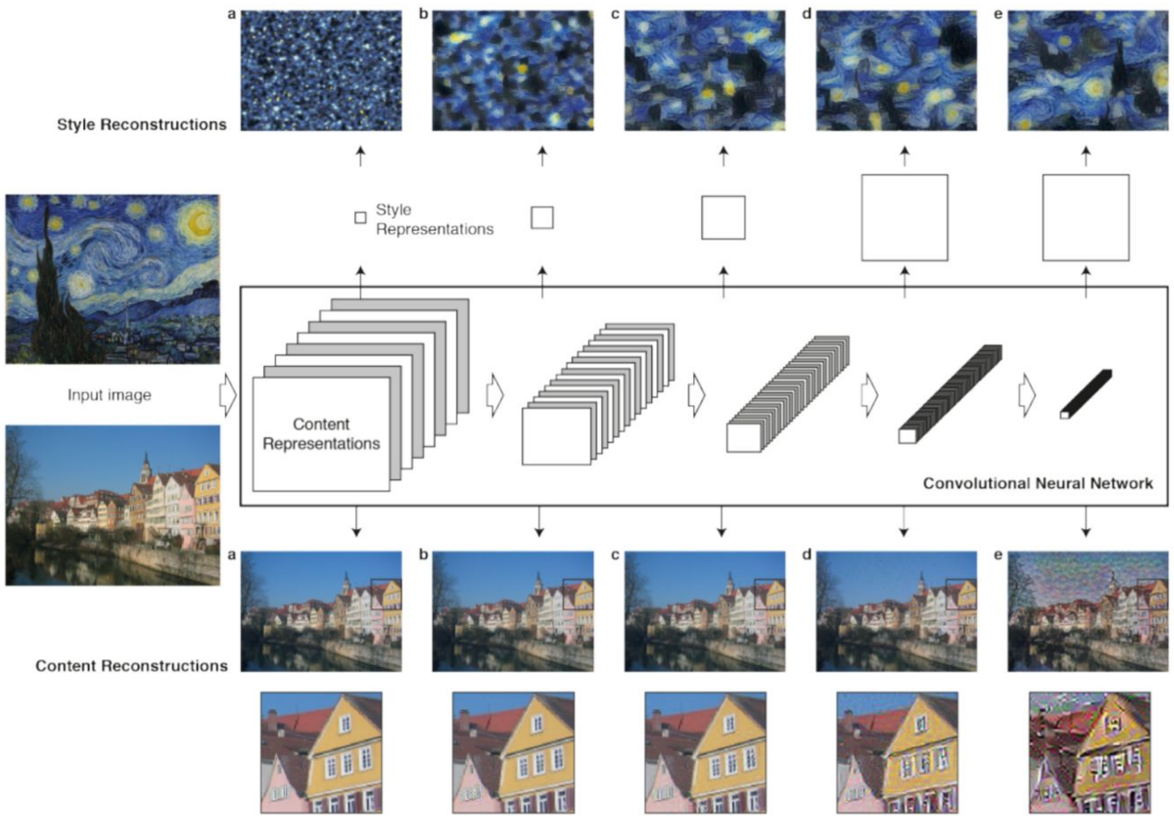


Figure 2 [8]

At the birth of Prisma in 2016, it made a huge impact in the mobile application market. People were eager to download and use it. In addition to the reason that everyone was curious about how their photos can be turned into art masterpieces, the image processing of Prisma was the main innovation: Before Prisma, most applications use preset algorithms. The user takes a picture and runs through these algorithms to get the desired picture. These preset algorithms do not change with the inputs, and Prisma became the first application that can adapt to inputs [9 &10].

Now that we have answered the first question, we need to turn our attention to Deep Image Analogy. What technology does Deep Image Analogy use? How is it different from Prisma? What was it improved upon? These are questions that need to be answered.

Regarding Deep Image Analogy, the best way to understand it is through its paper [11], and the author positioned it as a tool to implement visual attribute transformation across similar images. To accomplish this, it adapts the notion of "image analogy" [12] with features extracted from a Deep Convolutional Neural Network for matching. This technique establishes semantically-meaningful dense correspondences between the input images, which allow effective visual attribute transfer. Not only is it able to match under large visual variations, but it can also handle extreme visual variations, such as matching across an artistic painting and a real photograph. The reason behind this is that it uses high-level features, which makes it different from other common image transfer software.

Let's take a deeper look at how Deep Image Analogy is implemented:

First, we have to define what is visual attribute transfer: visual attribute transfer refers to copying visual attributes of one image such as color, texture, and style, to another image.

Then let us consider the input and output. In general, we only have two pictures, A and B ′. If we take A as the main base, we want to get a new picture and keep it consistent with A in content, but some image attributes (such as color, texture, etc.) are similar to B′. We refer to this picture as A′. B can also be defined in this way. We define all inputs and outputs as: A: A′ :: B: B′, where A′ and B are unknown. This representation has the following two constraints:

1. A and A' are perfectly aligned, as are B and B'.
2. A and B are similar in image attributes.

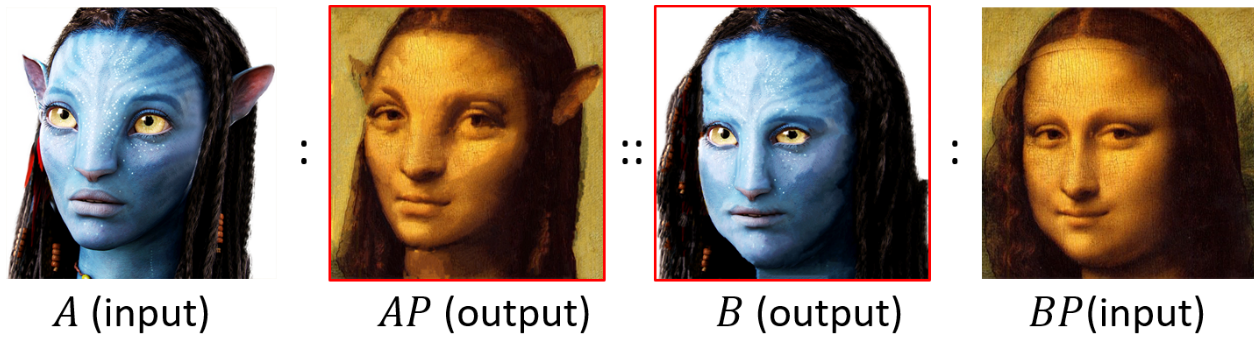


Figure 3 [13]

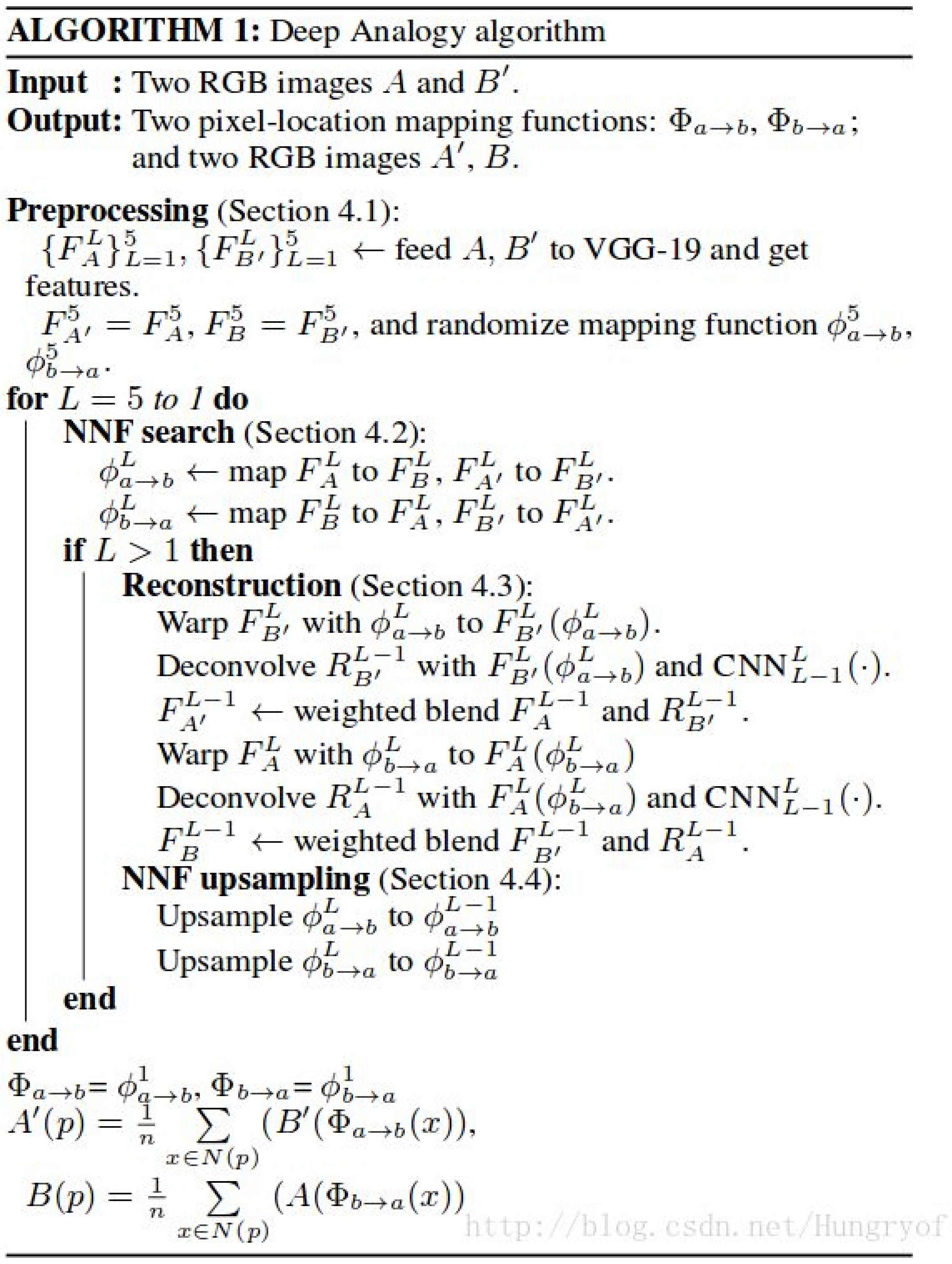
For example, in Figure 3, A' has the same outline as A, with content in almost the same place. However, the attributes of the entire image (such as color and texture) have become the same as B′, that is, it has become dark yellow with some textures of the face being erased.

The authors believe that it is difficult for CNNs to directly learn the mapping from A to B′. This is because A and B' have different visual effects, and there is misalignment on pixels. We should decompose this mapping into two parts:

1. Same position mapping of A → A'.
2. Style mapping of A'→ B' (Misalignment exists, so NNF search is needed).

Then we start to construct A′ and B: Since we already have A and B′, if we have A′ and B as well, then we can get the mapping of ΦA→B and ΦB→A. An ideal example of A′ has both the content structure of A and the details of B′. Now since A' and A are aligned, only the details are different (local textures, colors, etc.), then the features of A and A' on high-level CNNs must be very similar. So for the highest level features, we can directly assume that A and A' are equal, and then gradually construct A' in the lower layer of CNNs. The method of construction is to fuse the features of A in the current layer with the features of B'. And the same applies to B. By constructing layer by layer backward, the input layer A' and B are finally obtained.

Deep PatchMatch: Input A and B′, then we can get the features of A and B ′ in layer L, which is pre-calculated. Given latent images A′ and B, plus the two input images, we can define the mapping Φa→b and Φb→a to be a Nearest-neighbor Field search. As mentioned above, at the highest level of the network, we can consider A and A′, B′ and B to be the same. If we already have the features of all the 4 pictures of the L-th layer, we can fuse the A features of the L layer with the B′ features of the L layer for A′ on the L−1 layer. Since A and A' are similar in content, we can multiply the feature of A in layer L by a certain weight as part of the feature of A′ in layer L-1. However, features of B′ must also play a role in the construction of A′, after all B′ provides details for A′. As mentioned earlier, B' and A' are not aligned so we need to perform a mapping on B' so that the mapping features are aligned with A', and future fuse them together. So what is actually this mapping to B′? It is the calculated Φa→b of the L layer. The pseudo-code of the whole program is shown in Figure 4.



ALGORITHM 1: Deep Analogy algorithm

```
Input  : Two RGB images $A$ and $B'$.
Output: Two pixel-location mapping functions: $\Phi_{a\to b}$, $\Phi_{b\to a}$;
        and two RGB images $A'$, $B$.

Preprocessing (Section 4.1):
  $\{F_A^L\}_{L=1}^5, \{F_{B'}^L\}_{L=1}^5 \leftarrow$ feed $A$, $B'$ to VGG-19 and get
 features.
  $F_{A'}^5 = F_A^5$, $F_B^5 = F_{B'}^5$, and randomize mapping function $\phi_{a\to b}^5$,
 $\phi_{b\to a}^5$.
for $L = 5$ to 1 do
    NNF search (Section 4.2):
        $\phi_{a\to b}^L \leftarrow$ map $F_A^L$ to $F_B^L$, $F_{A'}^L$ to $F_{B'}^L$.
        $\phi_{b\to a}^L \leftarrow$ map $F_B^L$ to $F_A^L$, $F_{B'}^L$ to $F_{A'}^L$.
    if $L > 1$ then
        Reconstruction (Section 4.3):
            Warp $F_{B'}^L$ with $\phi_{a\to b}^L$ to $F_{B'}^L(\phi_{a\to b}^L)$.
            Deconvolve $R_{B'}^{L-1}$ with $F_{B'}^L(\phi_{a\to b}^L)$ and $\mathrm{CNN}_{L-1}^L(\cdot)$.
            $F_{A'}^{L-1} \leftarrow$ weighted blend $F_A^{L-1}$ and $R_{B'}^{L-1}$.
            Warp $F_A^L$ with $\phi_{b\to a}^L$ to $F_A^L(\phi_{b\to a}^L)$
            Deconvolve $R_A^{L-1}$ with $F_A^L(\phi_{b\to a}^L)$ and $\mathrm{CNN}_{L-1}^L(\cdot)$.
            $F_B^{L-1} \leftarrow$ weighted blend $F_{B'}^{L-1}$ and $R_A^{L-1}$.
        NNF upsampling (Section 4.4):
            Upsample $\phi_{a\to b}^L$ to $\phi_{a\to b}^{L-1}$
            Upsample $\phi_{b\to a}^L$ to $\phi_{b\to a}^{L-1}$
    end
end
$\Phi_{a\to b} = \phi_{a\to b}^1$, $\Phi_{b\to a} = \phi_{b\to a}^1$
$A'(p) = \frac{1}{n} \sum_{x\in N(p)} (B'(\Phi_{a\to b}(x))$,
 $B(p) = \frac{1}{n} \sum_{x\in N(p)} (A(\Phi_{b\to a}(x))$
```

Figure 4 [14]

Summary:

Here in this paper, I introduce Deep Image Analogy, which uses machine learning to change the art style of given pictures without destroying the overall structure of the pictures. Specifically, it uses Deep Convolutional Neural Network to implement Visual Attribute Transformation between pictures. In order to realize this, Deep Image Analogy first adopted the matching method, that is, judging whether the two pictures apply to the Similar Principle, and then performs object recognition through pre-trained CNNs, and finally PatchMatch with NNF. This software uses these techniques to achieve matching under large visual variations, and also handle extreme visual variations. But at the same time, it also has some limitations: Deep Image Analogy is designed to work on images with similar content composition. It is not effective for images that are semantically unrelated (eg, a headshot and a countryside photo), and is not designed to handle large geometric variations (including scales and rotations).

References

[1] GitHub. 2020. Msracver/Deep-Image-Analogy. [online] Available at: <https://github.com/msracver/Deep-Image-Analogy>.

[2] Medium. 2020. Learn To Build A Machine Learning Application From Top Articles Of 2017. [online] Available at: <https://medium.mybridge.co/learn-to-build-a-machine-learning-application-from-top-articles-of-2017-cdd5638453fc>.

[3] Prisma Photo Editor - Apps on Google Play. [online] Available at: <https://play.google.com/store/apps/details?id=com.neuralprisma>.

[4] En.wikipedia.org. 2020. Prisma (App). [online] Available at: <https://en.wikipedia.org/wiki/Prisma_(app)>.

[5] www.quora.com. 2020. What is the special algorithm used in the app Prisma? [online] Available at: <https://www.quora.com/What-is-the-special-algorithm-used-in-the-app-Prisma>.

[6] Gatys, L., Ecker, A. and Bethge, M., 2020. Image Style Transfer Using Convolutional Neural Networks.

[7] 揭开Prisma的神秘面纱（一） [online] Available at: <https://zhuanlan.zhihu.com/p/21821372>.

[8] Gatys, L., Ecker, A. and Bethge, M., 2016. A Neural Algorithm of Artistic Style. Journal of Vision, 16(12), p.326.

[9] Gupta, V., Sadana, R. and Moudgil, S., 2019. Image style transfer using convolutional neural networks based on transfer learning. International Journal of Computational Systems Engineering, 5(1), p.53.

[10] github.com. Apache MXNet (incubating) for Deep Learning. [online] Available at: <https://github.com/apache/incubator-mxnet>.

[11] Liao, J., Yao, Y., Yuan, L., Hua, G. and Kang, S., 2017. Visual attribute transfer through deep image analogy. ACM Transactions on Graphics, 36(4), pp.1-15.

[12] Dl.acm.org. Image Analogies | Proceedings Of The 28Th Annual Conference On Computer Graphics And Interactive Techniques. [online] Available at: <https://dl.acm.org/doi/abs/10.1145/383259.383295>.

[13] GitHub. Msracver/Deep-Image-Analogy. [online] Available at: <https://github.com/msracver/Deep-Image-Analogy/blob/master/windows/deep_image_analogy/example/readme/teaser.png>.

[14] Hungryof的专栏 [online] Available at: <https://blog.csdn.net/hungryof/>.