CS577 Group Project Final Report

Masked style transfer using Deep Style and U2Net

Baoli Hao **Ivan Gvozdanovic**

1 Introduction

Object detection and recognition using CNN type networks have been a prominent field of study in the last decade. In every branch of this area of research, the content representation is referred to as the feature responses in higher layers of the network [4]. Furthermore, one of those branches studies ways of how to combine features from a reference image to the source image serving as input. Capturing the texture information and extracting the style of the reference image which in turn is applied to the source image is called "style transfer". There are lots of previous studies on powerful non-parametric algorithms that can synthesize photorealistic natural textures by resampling the pixels of a given source texture [3, 2, 7, 13]. The majority of previous texture transfer algorithms rely on these methods for texture synthesis, while employing various approaches to preserve the structure of the target image. For example, Hertzman et al. employ image analogies for transferring texture, taking a pre-stylized image and applying its texture onto a target image [6]. Efros and Freeman propose a correspondence map that incorporates features from the target image, such as image intensity, to constrain the process of texture synthesis [2].

While these algorithms produce impressive results, they share a common fundamental limitation: they rely solely on low-level image features from the target image to guide the texture transfer process. By using the generic feature representations learned by high-performing CNN, Gatys et al. [5] performed a new algorithm to independently process and manipulate the content and the style of natural images (see one example below). This result was generated on the basis of the VGG-Network [12]. The structure of this system consists of two CNN networks. One processes the source image and the other captures the desired style from the *style* image. These two are then combined to get the Figure 1a below.

After obtaining the stylized image, our objective is to detect the object within the stylized image rather than the original content image. Lots of deep salient object detection networks [8, 9] have been proposed in recent years. The method we mainly use in our project is called U2-Net [10]. The reason we choose this powerful method is that it has two advantages: (1) it

(a) "Style-transferred" image

(b) Style image

can capture more contextual information from various scales. (2) It augments the overall depth of the architecture without imposing a substantial rise in computational expenses.

The rest of this report is organized as follows. In Section 2 we give the detailed description of the problem we try to address methodology. Meanwhile, we also introduce the detailed descriptions of methods used (*Neural Algorithm of Artistic Style*[5], *U2-Net* [10] and *Grabcut*[11]). Section 3 introduces the experiments we try and results we obtain. Section 4 discusses a brief summary of the main contributions of the project and the lessons we learn from the project, as well as a list of some potential future work. The github link including our proposal, code and report is also given.

2 Problem Description and Methods

The problem we try to address can be separated into two parts. Firstly, we aim to transfer style from one input image onto another using CNN nets. This produces a composite, stylized image which keeps the 'content' from the content image, but takes the style from the style image. We compare the effect of different parameters on the results These parameters consists of weights associated with 'content' loss, 'style' loss and total variation loss. By perturbing these weights, we can modify the total loss and in turn modify the result of the style transfer. Furthermore, we compare the similarity between content images and stylized images to give clearer quantitative results by using deep learning based approach.

Secondly, we use U2-Net to detect the object and cut it off from the stylized image. We compare the results obtained from using the U2-Net with those from the traditional Grabcut algorithm.

By using the methods above, the stylized object can be extracted in the end. Moreover, we can apply different styles to foreground and background images and combine them in the final output. Such method can provide many interesting patterns by changing the loss weights.

2.1 Style Transfer

We used NST algorithm to realize our style transfer. Fig 2 shows the details. Initially, the content and style features are extracted and stored. The style image vector $\overrightarrow{a}$ undergoes the network, calculating and storing its style representation A^l across all included layers (illustrated on the left side). Simultaneously, the content image vector $\overrightarrow{p}$ traverses the network, storing the content representation P^l within a single layer (depicted on the right). Subsequently, a random white noise image vector $\overrightarrow{x}$ journeys through the network, computing its style features G^l and content features F^l. Across each layer encompassed in the style representation, the style loss L_{style} is obtained by calculating the element-wise mean squared difference between G^l and A^l (seen on the left). Additionally, the content loss $L_{content}$ is computed by determining the mean squared difference between F^l and P^l (demonstrated on the right). The total loss L_{total} emerges as a linear combination of the content and style losses. Its derivative concerning the pixel values is determined through error back-propagation (displayed in the middle). This gradient is then utilized iteratively to adjust the image vector $\overrightarrow{x}$ until it converges, harmonizing both the style features of the style image vector $\overrightarrow{a}$ and the content features of the content image $\overrightarrow{p}$ (depicted in the middle and bottom sections).

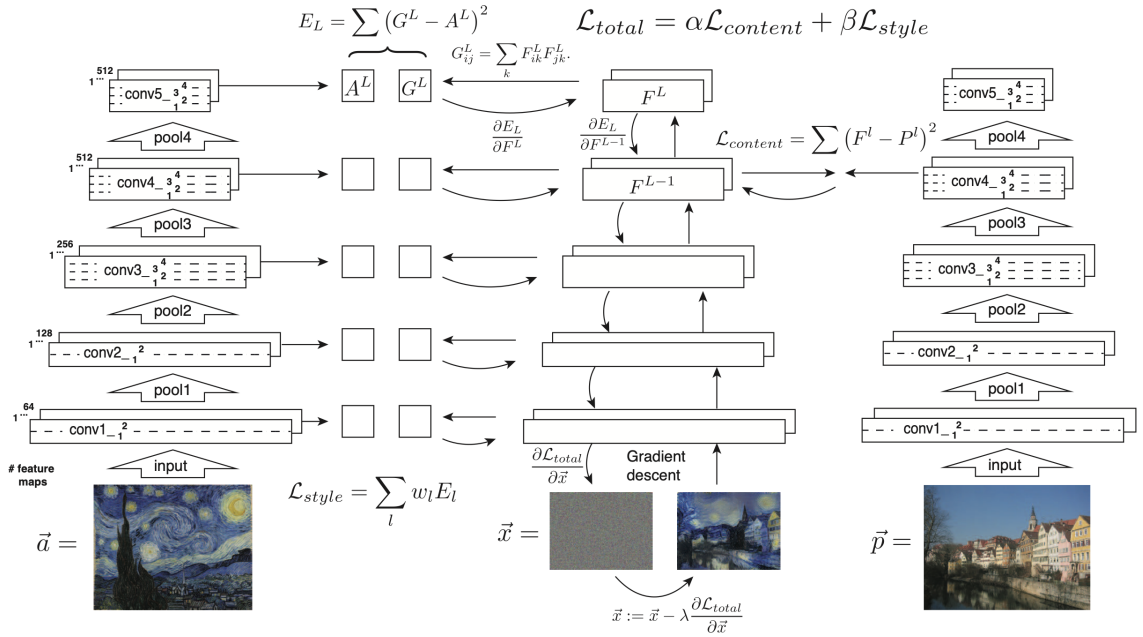


Figure 2: Style transfer algorithm

In this algorithm, there are style weights, content weights and tv loss weight. Different weights give us different results as seen in Fig 5.

2.2 Grabcut Algorithm and U2-Net

Next, what we want is to cut off the figure from the stylized image to prepare for "photo wake up". The algorithm we used is U2-Net [10]. Compared with traditional grabcut algorithm, U2-Net is much more powerful.

The grabcut algorithm works in the following way. The algorithm Iteratively preforms the following steps. In step 1, it estimates the color distribution of the foreground and background via a Gaussian Mixture Model. In step 2, it constructs a Markov random field over the pixels labels. Finally, in step 3 it applies a graph cut optimization to arrive at the final segmentation. Gaussian Mixture Models are used because it has been extensively documented that such probabilistic models preform well in the task of learning the underlying probability distribution. Markov random field is a set of random variables X_v, indexed by the vertices of a undirected graph $G = (V, E)$ which satisfy:

Pairwise Markov property: $X_u \perp\!\!\!\perp X_v | X_{V/\{u,v\}}$

Local Markov property: $X_u \perp\!\!\!\perp X_{V/N[v]} | X_{N(v)}$

Global Markov property: $X_A \perp\!\!\!\perp X_B | X_S$ for every path from A to B that passes thorugh S.

The Markov random field is constructed using the Gaussian mixture model and in combination with an energy function that prefers connected regions having the same label, an optimization scheme is employed in order to infer pixel values.

However, in the tests we preformed, we noticed that grabcut algorithm struggles to approximate the foreground and background if there are multiple objects in the foreground which overlap to some degree. We suspect that this happens due to the Gaussian mixture model wrongly approximating the color distribution. In the most extreme case, if we have multiple overlapping figures with colors that are similar to the background, the grabcut preforms poorly.

On the other hand, U2-Net takes a different approach to solving such a problem. The U2-Net represents a sophisticated two-level nested U-structure architecture crafted specifically for salient object detection (SOD). This design empowers the network to delve deeper, achieving enhanced resolution while efficiently managing memory and computational costs. The key lies in its nested U-structure: the lower level integrates a pioneering ReSidual U-block (RSU) module, extracting intra-stage multi-scale features while maintaining the integrity of the feature map resolution. At the upper level, a U-Net styled structure is employed, where each stage is enriched by an RSU block, further optimizing the network's performance. Fig 6 shows the results.

Salient object detection (SOD) is an important computer vision task which seeks to precisely detects and segments visually distinctive image regions from the perspective of the human visual system. SOD models aim to accomplish the goal of assigning high probability values to salient elements in a scene while producing a saliency maps. As mentioned above, in the U-net case,

this is done through the ReSidual U-blocks which have the following structure

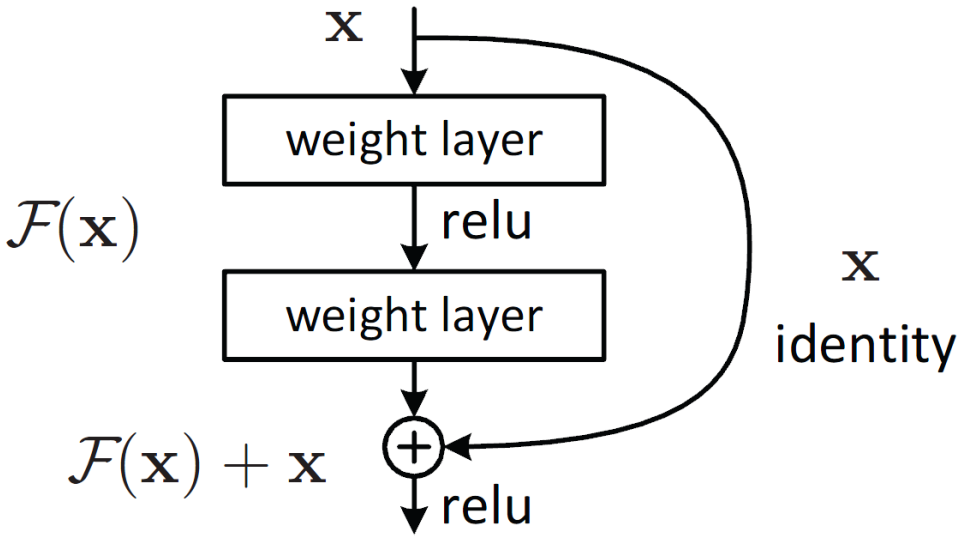


Figure 3: ReSidual Block

In the figure 3 above, x is the input, and $\mathcal{F}$ is the output defined as

$$\mathcal{F}(x) = W_2\sigma(W_1x + b_1) + b_1$$

The block is constructed using two layers of weights, a ReLU activation function in between them and finally, the output is added to the initial input to get $\mathcal{F} + x$. Such structure resembles the additive self-attention that we find in the encoder and decoder blocks. Furthermore, the subsequent blocks of the U2-Net decreases in dimension, the reason being the efficiency and speed of computation. Although the dimension of the output is shrinking, the network keeps record of all the SODs coming from each ReSidual block, and therefore, the resolution of the salient map remains high. Finally, all salient maps are stacked onto each other to produce the final result. The schematic of the U2-Net can be seen in Figure 6.

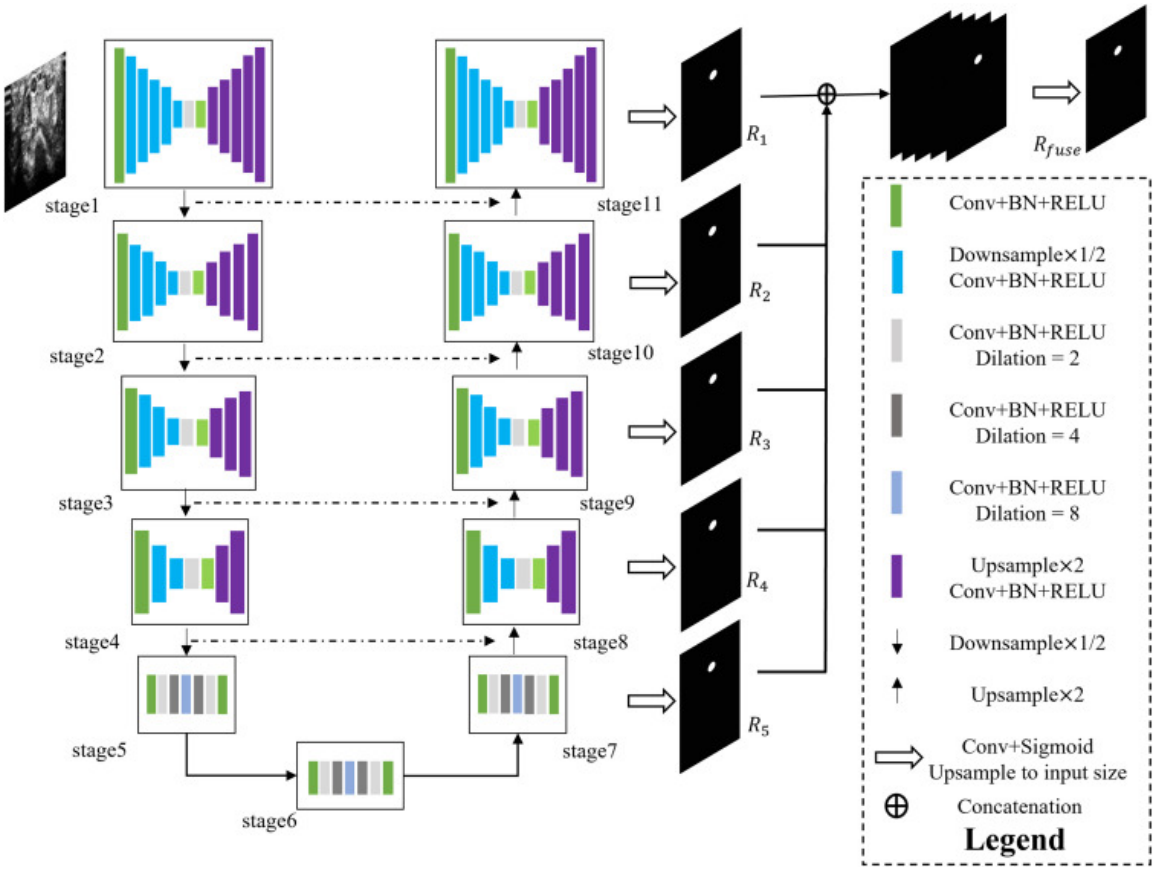


Figure 4: U2Net

2.3 Deep Learning-based Image Similarity Approach

Image similarity can be used in object recognition to match a given object with a known database. The basic idea of deep learning based approach is using pre-trained CNN like ResNet, VGG, and Inception to extract deep features from images [1]. CLIP (Contrastive Language-Image Pre-Training), developed by OpenAI, stands out as a remarkable multimodal zero-shot image classifier. It delivers impressive performance across diverse domains without the need for fine-tuning. By integrating the latest breakthroughs in large-scale transformers, such as those seen in GPT-3, CLIP extends the application of these advancements to the field of computer vision.

In our project, we want to employ this method to quantify our results. By employing CLIP-Based pre-trained model and the *torch*, *open_clip* and *sentence_transformers* libraries, the similarity between images can then be computed based on the cosine similarity or Euclidean distance of these feature vectors.

3 Results

Figure 5 shows us the stylized images by using different parameters. Changing style weight gives us less or more style on the final image, assuming you keep the content weight constant. Total variation (TV) is corresponding weight controls the smoothness of the image.

(a) Content Img

(b) $cw : 1e4$ $sw : 1e4$ $tv : 1$

(c) $cw : 1e5$ $sw : 1e2$ $tv : 10$

(d) $cw : 1e5$ $sw : 3e4$ $tv : 1.0$

Figure 5: Stylized Images using Different Weights

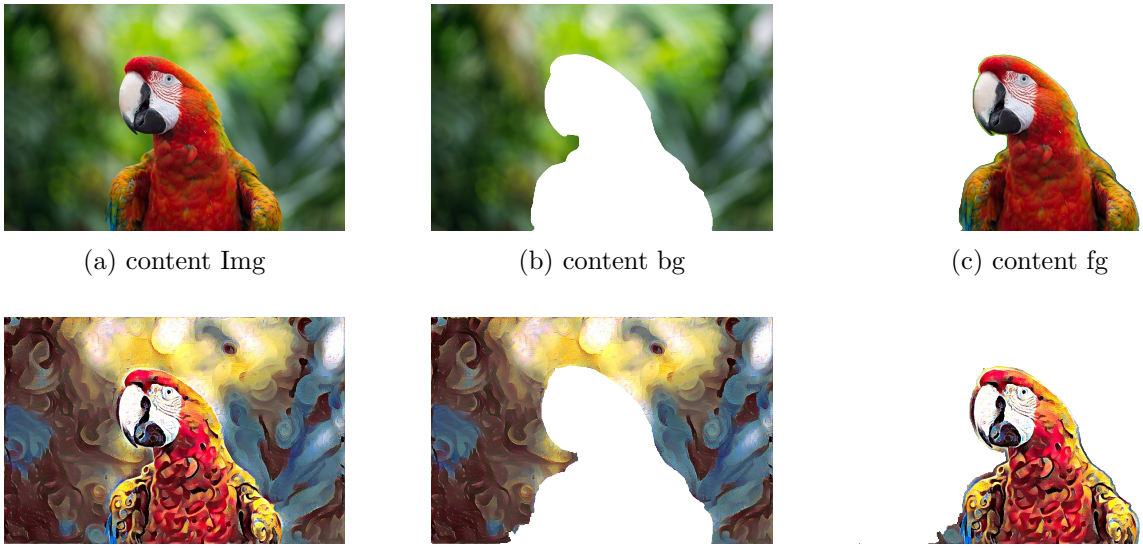
Figure 6 gives the results we get by using U2-Net architecture.

Figure 7 presents the results of Grabcut method and U2-Net architecture. It is obvious that the performance of grabcut method is unstable and particularly relies on the mask we choose.

Table 1 shows the image similarity scores of cut stylized parrot by using U2-Net and Grabcut algorithm. This table gives us a clearer quantitative results. We can see that U2-Net has a better performance than Grabcut since every score of U2-Net is higher than Grabcut algorithm.

	Content Img	Style Img	Truth cut Img
U2-Net Stylized Img (Parrot)	75.58	89.52	93.21
Grabcut Stylized Img (Parrot)	57.31	64.66	79.06

Table 1: Comparison of Image Similarity Score between Stylized Parrot by using U2-Net and Grabcut



(a) content Img (b) content bg (c) content fg

(d) Stylized Img (e) Stylized bg (f) Stylized fg

Figure 6: U2-Net Results

(a) content Img (b) Grabcut (c) U2-Net

(d) Stylized Img (e) Grabcut (f) U2-Net

Figure 7: The Comparison of results between Grabcut and U2-Net

4 Conclusions and Future Work

We worked together to present a study of the combination of two networks, assigned for solving two different tasks in computer vision. First we analyze the differences between the grabcut algorithm and U2-net and then we focus on the behaviour of the style transfer net by manipulating the loss function weights.
As expected, the grabcut preforms poorly in situations where there are multiple figures in the foreground of an image. We hypothesize that this is due to the Gaussian mixture model poor approximation of the color distribution, due to the mixing of the color of multiple figures. On the other hand, the U2-Net employs ReSidual blocks in order approximate a salient map. The U structure of the network helps with the dimnesionality problem and speeds up computation. By averaging the salient maps from each layer of network, we get an accurate representation of the foreground and background images.
Furthermore, we test the style transfer network by manipulating the content, style and total variation weights in order to get various results. The magnitude of each weight directly influences the pixel values of the final input. If the content weight is kept high while stlye and total variation weights are low, then the final output will resemble the original image. On the other hand, if the style weight is kept high and other low, the final output will resemble that of the original style image.
The plan for future work is to implement the Photo wake up model along with the U2-Net and style transfer network. By doing this, we can construct simple animations from single isolated foreground image while maintaining the style of the style image.

The code to the project can be found here: https://github.com/ivangvozdanovic/Deep-Learning

References

[1] S. Appalaraju and V. Chaoji, *Image similarity using deep cnn and curriculum learning*, arXiv preprint arXiv:1709.08761, (2017).

[2] A. A. Efros and W. T. Freeman, *Image quilting for texture synthesis and transfer*, in Seminal Graphics Papers: Pushing the Boundaries, Volume 2, 2023, pp. 571–576.

[3] A. A. Efros and T. K. Leung, *Texture synthesis by non-parametric sampling*, in Proceedings of the seventh IEEE international conference on computer vision, vol. 2, IEEE, 1999, pp. 1033–1038.

[4] L. Gatys, A. Ecker, and M. Bethge, *Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks (2015)*, CoRR abs/1505.07376.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge, *A neural algorithm of artistic style*, arXiv preprint arXiv:1508.06576, (2015).

[6] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, *Image analogies*, in Seminal Graphics Papers: Pushing the Boundaries, Volume 2, 2023, pp. 557–570.

[7] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, *Graphcut textures: Image and video synthesis using graph cuts*, Acm transactions on graphics (tog), 22 (2003), pp. 277–286.

[8] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang, *A simple pooling-based design for real-time salient object detection*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 3917–3926.

[9] N. Liu, J. Han, and M.-H. Yang, *Picanet: Learning pixel-wise contextual attention for saliency detection*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3089–3098.

[10] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand, *U2-net: Going deeper with nested u-structure for salient object detection*, Pattern recognition, 106 (2020), p. 107404.

[11] C. Rother, V. Kolmogorov, and A. Blake, *" grabcut" interactive foreground extraction using iterated graph cuts*, ACM transactions on graphics (TOG), 23 (2004), pp. 309–314.

[12] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, (2014).

[13] L.-Y. Wei and M. Levoy, *Fast texture synthesis using tree-structured vector quantization*, in Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, pp. 479–488.