# Portrait Segmentation and Background Stylization

**Ye Zhang, Xiangyu Zeng, Shuo Wang, Deyang Yin**
Department of Computer Science
Columbia University
{yz3060, xz2571, sw3135, dy2331}@columbia.edu

## Abstract

Image segmentation and stylization are two image processing technologies developed for a long time as separate processes. Powered by machine learning, both fields have been significantly improved, and the combination of the two leads to practical applications and business values. This paper analyzes one application that mixes the two processes - portrait segmentation and background stylization - by (1) evaluating traditional and latest deep learning methods in image segmentation, (2) analyzing a convolutional neural network in stylization, and (3) applying both for a complete picture transformation. The transformation results are satisfactory in terms of quality and performance.

## 1 Introduction

With the improvement of computing power and increasing popularity of smartphone portraits, image segmentation and stylization technologies have attracted tremendous scientific and business interests. Popular mobile communication and photo editing applications, such as Snapchat, Meitu and Faceu, have added image segmentation and stylization functionality to enhance photo quality and entertainment, and achieved promising business values. However, image segmentation and image stylization are still largely treated as two separate processes, and the mixture of both still has a wide range of applications to explore.

In this paper, we explore one of the mixed applications by combining the two processes to generate portraits with stylized backgrounds. This mixture has its prominent value in terms of removing imperfections in the background, hiding unwanted information, and enhancing interests and creativity. We perform a comparative analysis of image segmentation, stylization, and their combination. In particular, we compare the performance and quality of a traditional machine learning method (K-Means) and cutting-edge deep neural network (CNN + ResNet, FCN8s, and DeepLab-FLOV) image segmentation methods, and feed the segmented portraits into a convolutional neural network for background stylization to produce the final pictures.

## 2 Portrait Segmentation

### 2.1 K-Means

Segmenting a person out of the original portrait can be defined a classification problem. Specifically, to map the original portrait to its perfect segmentation, it is necessary to distinguish foreground from background. Based on human experience, the color of the foreground and the background are usually different in a photo. If one segment has a dark color, the other segment may have a light color. With such an assumption, it is possible that the dark pixels and the light pixels can be classified in different segments.

K-Means is the one of the simplest unsupervised clustering algorithm that divides a collection of objects into K groups. It divides an image into a number of discrete segments such that the pixels

have high similarity in each region and high difference between segments. The algorithm is briefly as follows:

- Compute the mean of each cluster.
- Compute the distance of each pixel from each cluster by computing its distance from the corresponding cluster mean and then assign each pixel to the nearest cluster.
- Iterate over the above two steps until it satisfies the tolerance or error value.

The initial assignment of pixels to clusters can be done randomly. In the process of the iterations, the model attempts to minimize the sum of distances from each pixel to its cluster centroid, over all clusters[1].

Convergence is reached when the objective function cannot be optimized any more. The segments generated are such that they are geometrically as compact as possible around their respective means. Using the set of feature images, a feature vector is constructed corresponding to each pixel ($e_1(a, b)$, $e_2(a, b)$, ..., $e_d(a, b)$), where $d$ is the number of feature images used for the segmentation process. K-Means can then be used to segment the image into two clusters corresponding to the foreground and the background respectively. Here, each feature can be assigned a different weight, which is calculated based on the feature importance. The distance between two vectors is computed using Equation $d^2(v, u) = w_i^2 \sum_{i=1}^{n} (v_i - u_i)^T (v_i - u_i)$.

Once the portrait has been segmented using the K-Means algorithm, the clustering can be improved by assuming that neighboring pixels have a high probability of falling into the same segment. Thus, even if a pixel has been wrongly clustered, it can be corrected by looking at the neighboring pixels.

Although K-means method has the reputation for simple implementation, it has drawbacks. First, the quality of the final clustering results depends on the random selection of initial centers. So, if the initial centroid is randomly chosen, it will produce different results for different initial centers. Therefore, the choice of the initial centers will influence whether we can have desired segmentation. Another problem is that the performance of K-Means depends significantly on the quality of the photo. It can classify foreground and background only if these two segments have obvious color difference. For those photos with high foreground and background chromatic similarity, it is possible that K-Means divides them into one segment.

### 2.2 Deep Neural Network

#### 2.2.1 Convolutional Neural Network

A convolutional neural network (CNN, or ConvNet) is a special kind of neural networks that has been widely applied to a variety of image-related problems.

CNN is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Here, we only focus on two dimensional convolutional (2D Conv) neural networks for image-related problems. Unlike the ordinary neural networks, CNN has a special architecture. The architecture of CNN usually is composed of convolutional layers and downsampling layers. Here, downsampling is also known as pooling.

Convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. A 2D convolutional layer is presented in Figure 1.

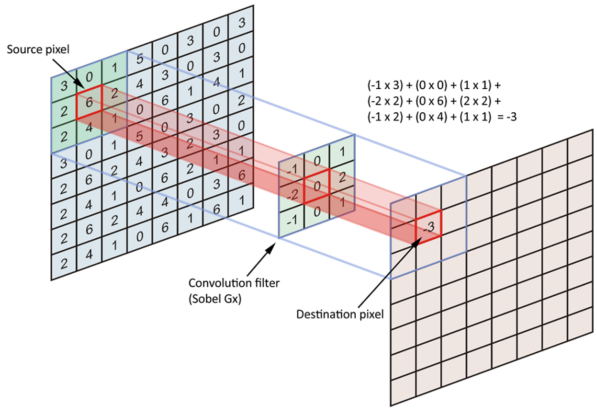


Figure 1: A 2D Convolutional Layer

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which max pooling is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, which is presented in Figure 2.

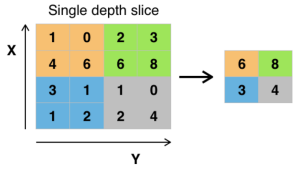


Figure 2: A Max Pooling Layer

VGG16 (Figure 3) is a deep convolutional network for object recognition developed and trained by Oxford's Visual Geometry Group (VGG). It achieved very good performance on the ImageNet dataset, and its pre-trained weights are freely available online. So, we adopt this model (without top dense layers) for both of our image segmentation and background stylization models.

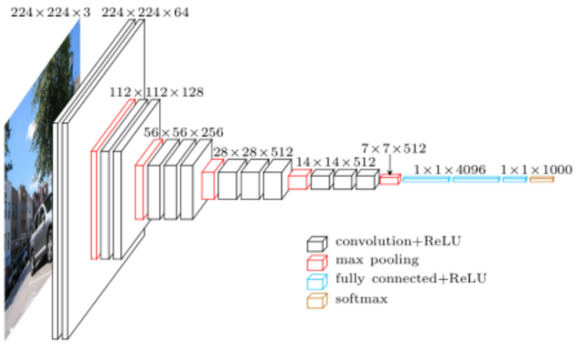


Figure 3: VGG16

### 2.3 Deep Neural Networks for Image Segmentation

In this section, we will introduce three variants of convolutional neural network we use for image segmentation.

#### 2.3.1 CNN + ResNet

First we design a shallow network ourselves (Figure 4). The first three blocks are the same with VGG16, but instead of dense layers, we add a Residual Network Block [2] as top layers to gain as low loss as possible despite the shallow network architecture. Finally, each pixel has only 1 feature, indicating that it is either foreground or background.

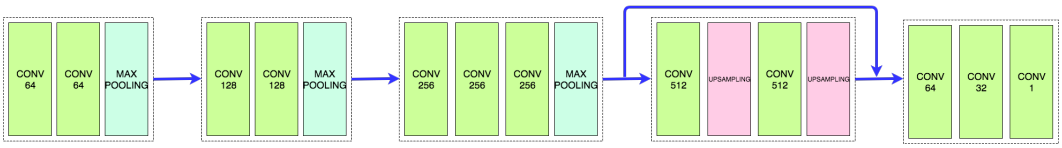


Figure 4: CNN + ResNet

#### 2.3.2 FCN8s

Fully convolutional networks (FCNs) [3] is trained end-to-end, pixel-to-pixel on semantic segmentation. Fully convolutional versions of existing networks predict dense outputs from arbitrary-sized inputs. Both learning and inference are performed whole-image-at-a-time by dense feedforward computation and backpropagation. In-network upsampling layers enable pixel-wise prediction and learning in nets with subsampling.

Figure 5 shows the architecture of FCN8s. As we can see, the first half is convolutional layers with downsampling layers, which are used to extract semantic features of images. For pixelwise prediction, we need to connect these coarse outputs back to the pixels. The second half is convolutional layers with upsampling layers, which are used to recover the image size.

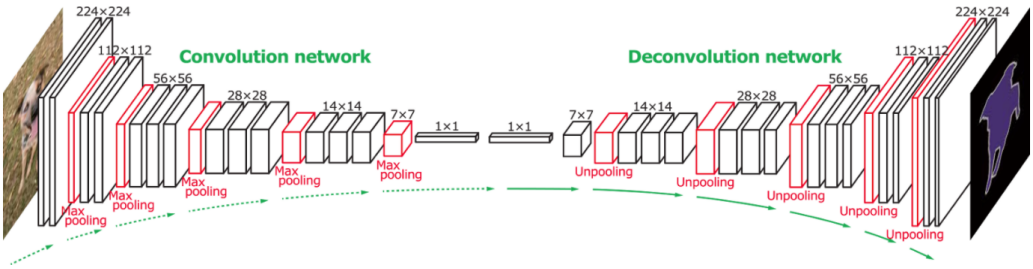


Figure 5: FCN8s

#### 2.3.3 DeepLab-LFOV

In particular there are two challenges in the application of CNNs to semantic image segmentation:

1. reduced feature resolution
2. existence of objects at multiple scales

DeepLab proposed a network architecture DeepLab-LFOV[4] to overcome these challenges.

The first challenge is caused by the repeated combination of max-pooling and downsampling performed at consecutive convolutional layers of CNNs originally designed for image classification. In order to overcome this hurdle and efficiently produce denser feature maps, DeepLab remove the downsampling operator from the last few max pooling layers of CNNs and instead upsample

the filters in subsequent convolutional layers (atrous convolution layers), resulting in feature maps computed at a higher sampling rate.

The second challenge is caused by the existence of objects at multiple scales. DeepLab propose a computationally efficient scheme of resampling a given feature layer at multiple rates prior to convolution. Rather than actually resampling features, DeepLab efficiently implement this mapping using multiple parallel atrous convolutional layers with different sampling rates.

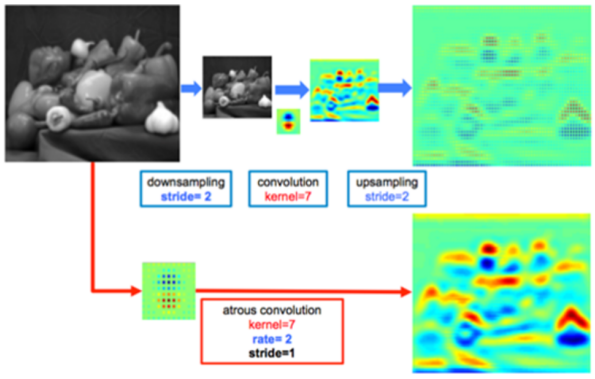


Figure 6: Atrous Convolution Layers

As we can see, too much downsampling and upsampling causes the reduction of signal resolution, and the atrous layers solve this problem.


Figure 7: The difference between VGG16 and our net


Figure 8: The difference of image size between VGG16 and our net

Our network is a simplified version of DeepLab-LFOV[3]. It is also adopted from VGG16. There are mainly two difference between our network and VGG16:

1. VGG16 is classification network and its output is scalar/vector. But what we need to do here is to do a pixel-level classification. So we need to recover the input picture size. We replace the VGG16's top dense layers to convolutional layers so that we get a fully convolutional network.

2. The second difference is that we use atrous convolution layers instead of traditional convolutional layers in the 5th and 6th block, same as the style of DeepLab-LFOV.

Figure 9: Some segmentation results

## 3 Background Stylization

### 3.1 Neural Style Transfer

Neural Style Transfer algorithm[5] is increasingly popular these days. This algorithm does well in landscape images. Here are some examples:

Figure 10

The Neural Style Transfer performs well in those landscape pictures:


Figure 11

### 3.2 Neural Network Structure

A pre-trained VGG16 network is used as a feature extractor. Two different features can be gained when the style image and the content image go through the VGG16 network separately. It is worth noticing that we extract features from five different layers, lower to higher, for stylization, and use only one higher layer as content feature extractor. The extractor can be seen in Figure 12:

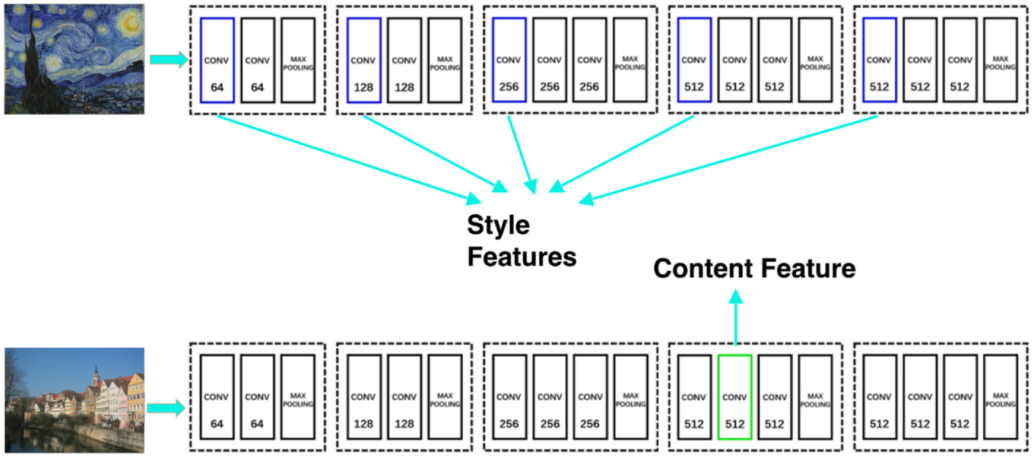


Figure 12

### 3.3 Feature Extraction

#### 3.3.1 Style Reconstruction

On top of the original CNN activations we use a feature space that captures the texture information of an input image. The style representation computes correlations between the different features in different layers of the CNN. We reconstruct the style of the input image from a style representation built on different subsets of CNN layers((a) 'conv1_1', (b) 'conv1_1' + 'conv2_1', (c) 'conv1_1' + 'conv2_1' + 'conv3_1', (d) 'conv1_1' + 'conv2_1' + 'conv3_1' + 'conv4_1', (e) 'conv1_1' + 'conv2_1' + 'conv3_1' + 'conv4_1' + 'conv5_1'). This creates images that match the style of a given image on an increasing scale.

#### 3.3.2 Content Reconstruction

Visualization of the information at different processing stages in the CNNs can be gained by reconstructing the input image from the network's responses in a particular layer. By reconstructing the input image from layers 'conv1_2', 'conv2_2', 'conv3_2', 'conv4_2' and 'conv5_2' of the original VGG16 network, it is obvious that reconstruction from lower layers is almost perfect which extractions contain almost all kinds of information. In contrast, for higher layers of the network, the network responses show that the detailed pixel information is lost while the high-level content of the image is preserved.

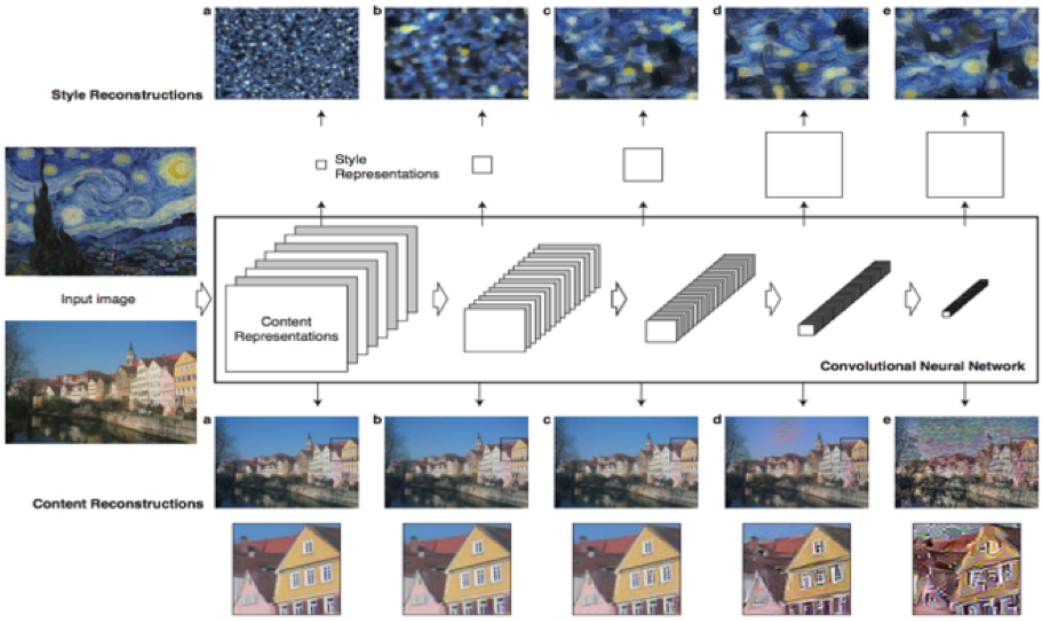


Figure 13

Based on the result of visualization, low level layers preserve more detailed information while high level layers give more abstract features. VGG16 is originally used for image classification. Therefore, in the high level layer, the information of objects will be highlighted. More specifically, the content information is saved while the texture, color and other information is dropped. Because of this property, it is easy to extract content features when processing the content image. As for style images, people usually need texture and color information, and thus it is necessary to extract features from each convolutional block to calculate the final loss. In our experiments, we choose layer 'conv5_2' for content representation and layers 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1' for style representation.

### 3.4 Loss Function

We use the feature space provided by a normalized version of the 13 convolutional and 5 pooling layers of the 16-layer VGG network. We normalize the network by scaling the weights such that the mean activation of each convolutional filter over images and positions is equal to one.

#### 3.4.1 Content Loss

The loss function for content information can be calculated as follows:

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, \vec{l}) = \frac{1}{2} \sum_{i,j} (F_{ij}^{l} - P_{ij}^{l})^2 \tag{1}$$

where $\vec{p}$ represents the original image vector, and $\vec{x}$ stands for generated image vector, $l$ is the layer symbol. $F^l$ is the feature representation of the generated image in layer $l$, and $P$ stands for content image feature representation.

Higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not contain too much exact pixel information values for reconstruction.

#### 3.4.2 Style Loss

We choose the feature responses in higher layers of the network as the content representation.

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l} \tag{2}$$

It's a inner production between the vectorized feature maps $i$ and $j$ in layer $l$.

To obtain the representation of the style of an input image, we use a feature space designed to capture texture information. This feature space can be built on top of the filter responses in any layer of the network. We use Gram matrix to represent the correlations between the different filter responses.

By including the feature correlations of multiple layers, we obtain a stationary, multi-scale representation of the input image, which captures its texture information but not the global arrangement. Again, we can visualize the information captured by these style feature spaces built on different layers of the network by constructing an image that matches the style representation of a given input image. This is done by using gradient descent from a white noise image to minimize the mean-squared distance between the entries of the Gram matrices from the original image and the Gram matrices of the image to be generated.

The contribution of layer l to the total loss is as follows:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \tag{3}$$

The total style loss is as follows:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \tag{4}$$

where $w_l$ represent different weighting factors of the contribution of each layer to the total loss.

### 3.4.3 Total Loss

To transfer the style of an artwork onto a photo, we synthesize a new image that simultaneously matches the content representation of the photo and the style representation of the artwork. We can jointly minimize the distance of the feature representations of a white noise image from the content representation of the photograph in one layer and the style representation of the painting defined on a number of layers of the Convolutional Neural Network. The loss function we minimize is as follows:

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x}) \tag{5}$$

where $\alpha$ and $\beta$ are the weighting factors for content and style reconstruction, respectively. The whole structure of style transformation is depicted as follows.

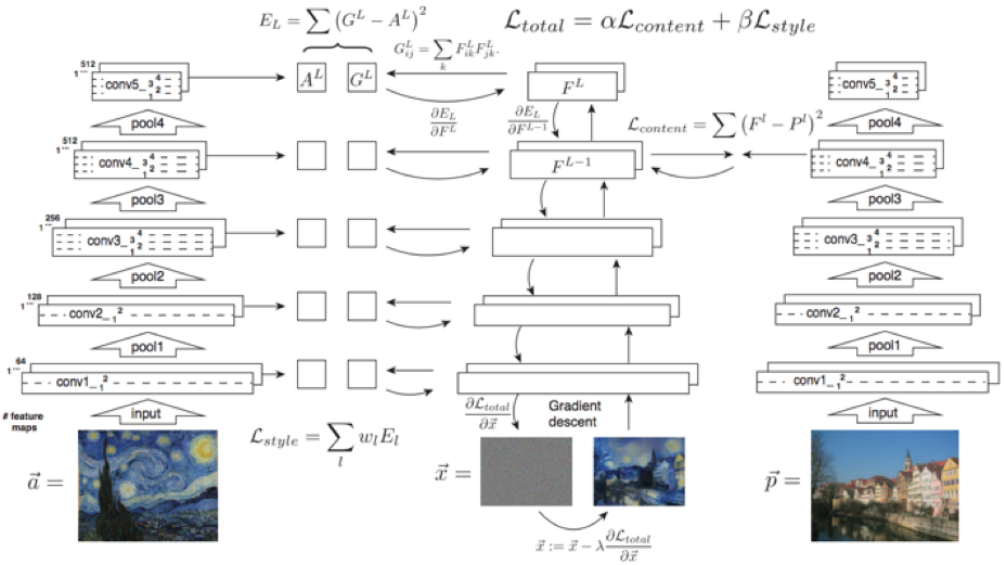


Figure 14

### 3.5 Style Transfer Algorithm

1. Content and style features are extracted and stored. The style image is passed through the network and its style representation on all layers included is computed and stored (see the left part). The content image is passed through the network and the content representation in one layer is stored (see the right part of Figure 14).

2. A random white noise image is passed through the network and its style features and content features are computed. On each layer included in the style representation, the element-wise mean squared difference between the white noise image and the style image is computed to give the style loss(see the left part). Also the mean squared difference between white noise image and the style image is computed to give the content loss (see the right part of Figure 14).

3. The total loss is a linear combination between the content and the style loss.

4. The derivative of the total loss with respect to the pixel values can be computed using error back-propagation (see the middle part). This gradient is used to iteratively update the input white-noise image until it simultaneously matches the style features of the style image and the content features of the content image.

## 4 Combination of Portrait Segmentation and Background Stylization

As stated above, standard neural transfer can perform well in pure landscape photos. When it comes to a situation where a person or other objects are inside, we want them clearly showed instead of showing the following result:

Figure 15

So, we want to perform style transfer without ruining the face when images have objects inside. The goal is to combine the content of one image with the style of another image (usually a famous artworks) to get a result image.

However, when we want to transfer a picture with people in it, we might get a horrible face because the style transformation might cause the image to lose its original focus, making the figure more ”chaotic” as part of the image.

Most of the time, we may only want to get a styled background while keep ourselves look nice. So we combine the image segmentation with the neural style transfer to solve that.

Figure 16

## 5 Evaluation and Comparison

We evaluate our three deep neural networks based on four metrics from common semantic segmentation and scene parsing evaluations that are variations on pixel accuracy and region intersection over union (IU)[3]. Let $n_{ij}$ be the number of pixels of class $i$ predicted to belong to class $j$, where $n_c l$ is the number of classes included in ground truth segmentation, and let $t_i = \sum_j n_{ij}$ be total number of pixels of class $i$ in ground truth segmentation. We compute:

- pixel accuracy: $\frac{\sum_i n_{ii}}{\sum_i t_i}$
- mean accuracy: $\frac{1}{n_c l} \sum_i \frac{n_{ii}}{t_i}$
- mean IU: $\frac{1}{n_c l} \sum_i \frac{n_{ii}}{(t_i + \sum_j n_{ji} - n_{ii})}$
- frequency weighted IU: $(\sum_k t_k)^{-1} \sum_i \frac{n_{ii}}{(t_i + \sum_j n_{ji} - n_{ii})}$

| Accuracy \ Model | FCN + ResNet | DeepLab LFOV | FCN8s |
|---|---|---|---|
| Pixel Accuracy | 86.97 | **97.34** | 96.50 |
| Mean Accuracy | 86.43 | **97.11** | 96.63 |
| Mean IU | 85.72 | 94.70 | **98.04** |
| Frequency Weighted IU | 85.90 | 94.81 | **98.16** |

From the evaluation result, FCN + ResNet has the worst performance. This is reasonable because this model has the shallowest network. DeepLab LFOV and FCN8s have very good performance dealing with the image segmentation problem. In fact, these two models are the state-of-art networks for image segmentation problem.

## 6 Conclusion

In this paper we transform normal photos to portraits with stylized backgrounds and perform analysis and experiments with respect to image segmentation and stylization. For portrait segmentation, we have discovered that, while K-Means is easy to implement and can distinguish light pixels from dark pixels, the algorithm is too simple that the result depends heavily on the relative darkness of different areas in the picture, not reflecting the actual objects if the layout and light distribution of a photo are complex and subtle. Deep neural networks, on the other hand, can clearly separate portraits from backgrounds close to the ground truth. Among the experimented deep neural networks, CNN + ResNet shows acceptable results with a shallower structure, while FCN8s and DeepLab-FLOV networks have much better quality with deeper architectures. For background stylization, we have demonstrated the importance of the design of loss functions for content representation, style representation and the combination of the two for generating results via the gradient descent method.

## 7 Future Work

The process of portrait segmentation and background stylization has room for further investigations and improvements. Despite good results of FCN8s and DeepLab-FLOV networks comparing to the traditional K-Means algorithm in complex and nuanced light and shadow settings, there are still possibility to lose parts of the main object (i.e. human, animals, etc.) and include portrait-irrelevant parts into the extracted results, as shown in our datasets. Adding CRFs(Conditional Random Field) to semantic segmentation may achieve much better results. Also, a deeper analysis and tweaking is required for better quality. For background stylization, the design of style and content representation loss functions and the balance of the two are still subject to improve, and using chain blurred method is one possible solution. Additionally, we could also improve the entire transformation by adding extra image processing steps, such as image matting, light adjustment, color adjustment and blurring, by advanced machine learning methods to produce more natural-looking pictures ready for industrial use.

## References


[1] Dhanachandra, N., Manglem, K., & Chanu, Y. J. (2015). Image segmentation using K-means clustering algorithm and subtractive clustering algorithm. Procedia Computer Science, 54, 764-771.

[2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).

[3] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3431-3440).

[4] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:1606.00915.

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

[6] Li, Y., Wang, N., Liu, J., & Hou, X. (2017). Demystifying Neural Style Transfer. arXiv preprint arXiv:1701.01036.

[7] Johnson, J., Alahi, A., & Fei-Fei, L. (2016, October). Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision (pp. 694-711). Springer International Publishing.

[8] Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2414-2423).

[9] Novak, R., & Nikulin, Y. (2016). Improving the neural algorithm of artistic style. arXiv preprint arXiv:1605.04603.

[10] Gatys, L. A., Bethge, M., Hertzmann, A., & Shechtman, E. (2016). Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897.

[11] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062.

[12] Glorot, X., & Bengio, Y. (2010, May). Understanding the difficulty of training deep feedforward neural networks. In Aistats (Vol. 9, pp. 249-256).