Exemplar-Based Image and Video Stylization Using Fully Convolutional Semantic Features

Feida Zhu, Zhicheng Yan, *Member, IEEE*, Jiajun Bu, *Member, IEEE*, and Yizhou Yu, *Senior Member, IEEE*

***Abstract*— Color and tone stylization in images and videos strives to enhance unique themes with artistic color and tone adjustments. It has a broad range of applications from professional image postprocessing to photo sharing over social networks. Mainstream photo enhancement softwares, such as Adobe Lightroom and Instagram, provide users with predefined styles, which are often hand-crafted through a trial-and-error process. Such photo adjustment tools lack a semantic understanding of image contents and the resulting global color transform limits the range of artistic styles it can represent. On the other hand, stylistic enhancement needs to apply distinct adjustments to various semantic regions. Such an ability enables a broader range of visual styles. In this paper, we first propose a novel deep learning architecture for exemplar-based image stylization, which learns local enhancement styles from image pairs. Our deep learning architecture consists of fully convolutional networks for automatic semantics-aware feature extraction and fully connected neural layers for adjustment prediction. Image stylization can be efficiently accomplished with a single forward pass through our deep network. To extend our deep network from image stylization to video stylization, we exploit temporal superpixels to facilitate the transfer of artistic styles from image exemplars to videos. Experiments on a number of data sets for image stylization as well as a diverse set of video clips demonstrate the effectiveness of our deep learning architecture.**

***Index Terms*— Image stylization, fully convolutional networks, color transform.**

I. INTRODUCTION

STYLISTIC enhancement adjusts an image or video for enhancing artistic styles that convey unique themes. Unlike conventional image enhancement focusing on fixing photographic artifacts (under/over exposure, insufficient contrast, etc.), stylistic enhancement involves dramatic color and tone adjustments to achieve distinctive visual effects. For example, the *X-PRO II* filter from mobile photo App Instagram expresses a wistful mood by simulating the cross processing procedure of photographic films. Professional image editing software (such as Adobe Lightroom) and social mobile Apps (such as Instagram) provide users with predefined styles, which are often hand-crafted through a trial-and-error process. An example of learning semantics-aware photo adjustment styles is shown in Figure 1.

Conventional automatic photo adjustment has difficulty in representing complex color transforms between images before and after adjustment. Most of them merely model global color transforms without considering local semantic contexts. Although more sophisticated adjustments introduce spatially varying effects according to local image statistics, they still lack a semantic understanding of image contents. On the contrary, professional photographers often manually enhance images in a semantics-aware manner. For instance, when enhancing photos to create a nostalgic theme, photographers might apply more exaggerated adjustments to a photo of *Broadway* taken in year 1950 than a photo of *Burj Khalifa*, which is the tallest skyscraper built in year 2010, as the former is more fitting with the theme. At a local scale, they employ selection tools to isolate semantic regions (faces, buildings, etc.), which are enhanced with distinct sets of adjustments. For instance, there may exist a demand to apply exaggerated adjustments to foreground objects to help them stand out. The ability of applying distinct adjustments to semantic regions enables a broader range of visual styles.

As stylistic adjustments interact with image semantics and contexts in a complicated manner, it is extremely challenging to manually define the relationships between them. To automatically learn stylistic enhancement from a small set of image exemplars, in this paper, we propose a novel deep learning architecture. Unlike existing work that integrates hand-crafted features with a small-scale multilayer neural network [1], our solution is a large-scale deep network. It consists of fully convolutional networks (FCNs) for automatic feature extraction and fully connected neural layers for adjustment prediction. Recently, fully convolutional networks [2]–[4] have proven to be efficient and powerful deep learning architectures for image processing and visual understanding tasks, such as semantic image segmentation, contour detection and salient object detection, that need to generate high-resolution outputs. In our deep network, feature maps with sufficiently large receptive fields are computed to model contexts. We further employ fully connected neural layers, which predict color transforms according to contexts and pixel features.

Manuscript received June 25, 2016; revised November 21, 2016; accepted January 21, 2017. Date of publication May 10, 2017; date of current version May 26, 2017. This work was supported by the Hong Kong Research Grants Council through General Research Funds under Grant HKU17209714. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Weisheng Dong. *(Corresponding author: Feida Zhu.)*

F. Zhu and Y. Yu are with the Department of Computer Science, The University of Hong Kong, Hong Kong (e-mail: zhufeida@hku.hk).

Z. Yan is with Facebook AI Research Menlo Park, California.

J. Bu is with College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes one PDF file of image results and one MP4 video file of video results. Contact zhufeida@hku.hk for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2703099

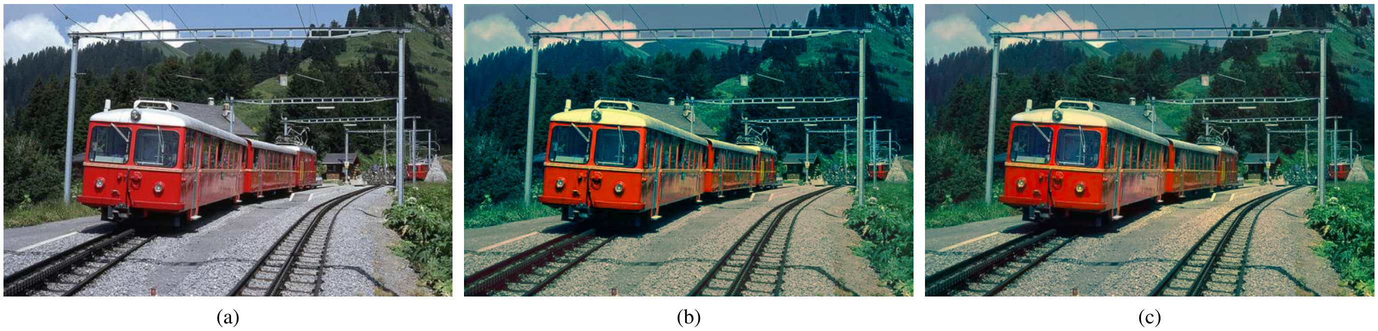


Fig. 1. An example of learning semantics-aware photo adjustment styles. **Left**: Input image. **Middle**: Manually enhanced by photographer. Distinct adjustments are applied to different semantic regions. **Right**: Automatically enhanced by our deep learning model trained from image exemplars. (a) Input image. (b) Ground truth. (c) Our result.

We seamlessly integrate the FCNs with fully connected layers, and an input image can be enhanced with a single forward pass in our deep network.

Furthermore, our deep learning model trained on image exemplars can be readily deployed to enhance videos with artistic styles. Compared with image stylization, video stylization faces extra challenges. Simply enhancing a video in a frame-by-frame manner not only results in an inefficient solution, but also cannot preserve the temporal coherence. Instead, we adopt temporal superpixels (TSPs) [5], which are spatiotemporal primitive regions consistently tracking image regions and object parts across frames. In our video stylization pipeline, a color transform is predicted for each TSP and applied to all pixels within the same TSP. To accelerate computation, features are only extracted for a minimum number of video frames intersecting all TSPs.

In summary, this paper has the following contributions.

- We propose a novel deep learning architecture for stylistic image enhancement. It consists of fully convolutional networks and fully connected neural layers. Our deep network is capable of learning distinct enhancement styles from a small set of training exemplars. Enhancing a novel image only requires a single forward pass through our network.
- Fully convolutional networks in our architecture extracts global features and contextual features. Our novel contextual features have two parts. The first part is a semantics-aware feature extracted from deep layers of a fully convolutional network; the second part consists of a set of color histograms over a small spatial grid.
- We demonstrate that deep neural networks trained with image exemplars can be used to enhance videos as well. We segment a video into temporal superpixels, and apply both temporally coherent and spatially smooth adjustments to them. A greedy frame selection algorithm is developed to reduce the computational cost of feature extraction.

II. Related Work

A. Image Enhancement

On the basis of whether example data is used, image enhancement approaches can be broadly classified into two categories, hand-crafted approaches and data-driven approaches. Hand-crafted filters for image enhancement are commonly seen in image processing softwares and photo management Apps, such as Adobe Lightroom, Google Photos and Instagram. They support a range of adjustments, from exposure correction, contrast enhancement to artistic retouching. Meanwhile, researchers have made an enormous amount of effort to develop fully automatic methods for tone adjustment [6], [7], color management [8], [9], detail manipulation [10]–[12] and image smoothing [13]–[15]. On the other hand, interactive enhancement techniques allow users to perform adjustments at sparse locations, and propagate them to the full image domain [16], [17] while preserving image structures.

In contrast to hand-crafted approaches, which merely achieve a predefined set of effects, data-driven approaches are capable of learning new effects from examples, and thus offer a more flexible set of adjustments. They replace time-consuming manual design with automatic model learning [18]–[20]. Bychkovsky *et al.* [21] predict global tonal adjustments using a Gaussian process regression model built from a large dataset of images. Their regression model only extracts image global features and does not accommodate semantics-aware local adjustments. Kang *et al.* [22] introduce user preference in image global enhancement, and retouch a novel image by finding most similar examples in a database and transferring their tone and color adjustments. Joshi *et al.* [23] retouch imperfect personal photos by leveraging existing high-quality photos of the same person. Lee *et al.* [24] develop an unsupervised technique for learning content-specific style rankings and transfers highly ranked styles from exemplars to an input photo. However, their styles are still limited to global color and tone transforms. Wang *et al.* [25] approximate complex spatially varying tone and color adjustments with piecewise polynomial functions, which rely on low-level image statistics only and are not aware of image semantics. In contrast, our model applies adjustments to local semantic regions using features extracted with a deep convolutional neural network pretrained on thousands of semantic categories. Shih *et al.* [26] synthesize images associated with different times of day by learning locally affine models after locating a matching video within a time-lapse video database. Gatys *et al.* [27] perform image style transfer using convolutional neural networks. A new image is synthesized by matching the coarse structures of a content image and the texture features of a style image.

The work closely related to ours is presented in [1], where local tone and color adjustments are predicted using a combination of image global statistics, contextual semantic features and pixelwise color and spatial features. They rely on existing computationally expensive scene parsing [28] and object detection [29] tools to explicitly generate a semantic label map, from which contextual features are formed by multiscale spatial pooling. These scene parsing and object detection tools have limited accuracy and robustness. Our method in this paper does not rely on explicit scene labeling, instead, perform scene understanding implicitly by extracting contextual semantic features using a fully convolutional network, which offers higher accuracy and improved robustness, and also runs faster than traditional scene understanding tools.

B. Video Enhancement

Traditional professional video editing softwares (Adobe After Effects, Nuke, etc.) offer a suite of predefined operations with tunable parameters that apply common global adjustments (exposure/color correction, white balancing, sharpening, denoising, etc). Local adjustments within specific spatiotemporal regions are usually accomplished with masking layers created with intensive user interaction. Both parameter tuning and masking layer creation are labor intensive processes.

Example-based approaches have been proposed to automatically transfer adjustments from exemplars to novel videos, and alleviate the needs of user interaction. Bonneel *et al.* [30] propose to transfer the color palette of an example video to a novel input video to achieve color grading. However, they rely on users to provide foreground-background segmentation for estimating separate color transforms. In contrast, our approach is fully automatic and implicitly distinguishes semantic regions from each other by using deep features from convolutional neural networks. Xue *et al.* [31] study the relationships between film tags (director, emotion, etc.) and color styles for movie color grading. Their method is solely dependent on low-level image statistics (luminance, hue and saturation), and is not able to support semantics-aware local adjustments. Ruder *et al.* [32] extend [27] to video stylization by proposing a temporal loss term between frames to maintain temporal coherence.

III. Overview

Given a set of exemplar image pairs, each representing a photo before and after pixel-level color and tone adjustments following a particular style, we wish to learn a computational model that can automatically adjust a novel input photo in the same style. We still cast this learning task as a regression problem as in [1]. For completeness, let us first review their problem definition and then present our new deep learning based architecture and solution.

We seek a color transformation function ϕ such that, for every pixel p_i in the exemplar images, the color transform returned by ϕ is $\phi(\theta, x_i)$, which maps the pixel color at p_i before adjustment, $c_i = [L_i \; a_i \; b_i]^T$ (CIELab color space), to its corresponding pixel color y_i after adjustment. Here θ denotes the model parameters and x_i the feature vector at pixel p_i. The quadratic color basis $V(c_i) = [L_i^2 \; a_i^2 \; b_i^2 \; L_ia_i \; L_ib_i \; a_ib_i \; L_i \; a_i \; b_i \; 1]^T$ is used to absorb high-frequency pixelwise color variations. The product of the color transform $\phi(\theta, x_i)$ and the color basis is a prediction of the enhanced color. Since our color space has 3 channels and the quadratic color basis is a 10-dimensional vector, $\phi(\theta, x_i)$ is in fact a 3x10 matrix. The model parameters of θ are learnt by minimizing the following objective function, which measures the squared differences between the predicted and groundtruth enhanced colors.

$$\arg\min_{\theta} \sum_i ||\phi(\theta, x_i)V(c_i) - y_i||^2 \tag{1}$$

Since each color transform is a matrix with 30 elements, solving a distinct color transform at every pixel is an under-constrained problem. To sufficiently constrain every color transform, we group pixels in an image into a predefined number of superpixels, $\{s_v\}_{v=1}^{N_s}$, and let all pixels within a superpixel s_v share a single color transform $\mathcal{F}_v = \phi(\theta, x_v)$.

Thus, the above objective function is revised as follows.

$$\arg\min_{\theta} \sum_v \sum_{j \in s_v} ||\phi(\theta, x_v)V(c_j) - y_j||^2. \tag{2}$$

Refer to [1] for more details.

A. Photo Stylization Using FCNs

In this paper, we model the entire process to produce an enhanced image using deep neural networks. The complete architecture of our deep network is shown in Figure 2. Our deep network makes use of fully convolutional networks to extract global and contextual semantic features for every superpixel in the input image. The global feature of a superpixel is the globally pooled deep CNN feature. And the pixel feature is simply the pixel color at the centroid of the superpixel. As shown in Figure 3, the contextual feature of a superpixel consists of three components. The first two components are deep CNN features extracted over two differently sized receptive fields centered at the centroid of the superpixel. The third component is a set of concatenated color histograms computed over a 3x3 grid also centered at the centroid of the superpixel. Let us now explain the architecture of our deep network and how we compute these features in greater details.

An input image is fed into a fully convolutional network (FCN). Deep features extracted using this FCN pass through a *global pooling* layer and become a single global feature vector x^g. We replicate x^g at the centroids of all superpixels in a *replication* layer to obtain per-superpixel global features $\{x_v^g\}_v$ (Section IV-A). We also upsample the input image, and feed the original and upsampled images into two FCNs to extract two feature maps. The receptive fields of these two feature maps respectively have the same size as the two (blue) windows in Figure 3. These feature maps represent semantic contexts at two different scales, and are sampled at the centroids of superpixels in a *sampling* layer to obtain contextual semantic features $\{x_v^{s1}\}_v$ and $\{x_v^{s2}\}_v$ (Section IV-B). We compute contextual color histogram features $\{x_v^h\}_v$ over two-scale pooling regions (red grid in Figure 3) in a *color*

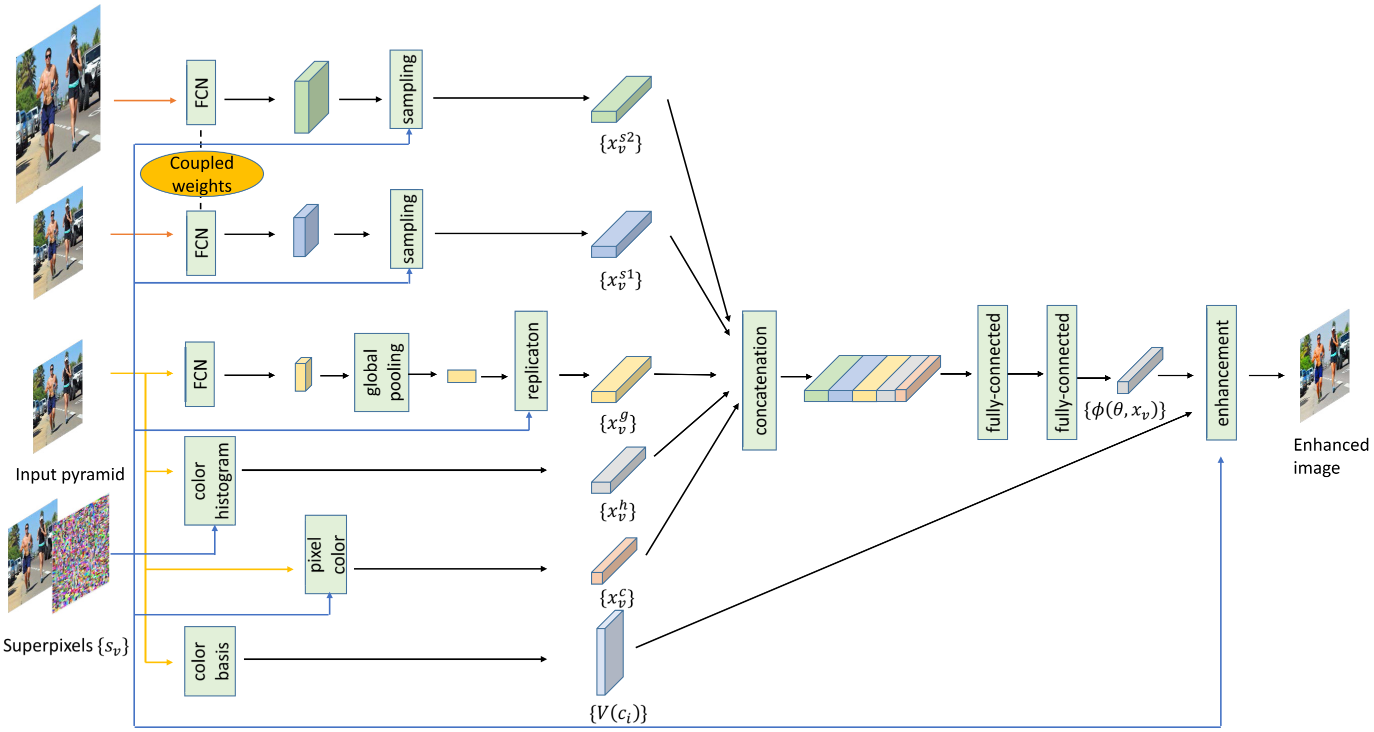


Fig. 2. The architecture of our deep neural network for stylistic image enhancement.

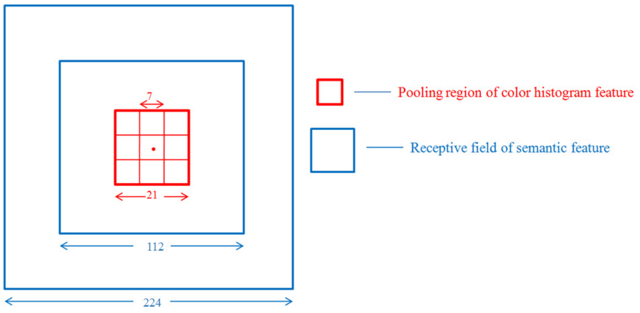

7
21
112
224
Pooling region of color histogram feature
Receptive field of semantic feature

Fig. 3. Contextual feature descriptors. There are both semantic features and color histogram features in our contextual feature description. Two-scale contextual semantic features are extracted from the two large blue windows using a fully convolution network while color histogram features are extracted from 9 + 1=10 pooling regions over a 21 × 21 window.

histogram layer (Section IV-B). The global feature, contextual semantic features, contextual color histogram feature as well as the pixel feature for every superpixel are concatenated and passed through two fully connected neural layers to produce the set of per-superpixel color transforms $\{\phi(\theta, x_v)\}_v$. Each of these color transforms is applied to the per-pixel color basis vectors $\{V(c_i)\}_i$ in the same superpixel in an *enhancement* layer to produce the final enhanced image.

1) Global Pooling Layer: Given an incoming feature map $\{f_{x,y,c}\}$ of size $H \times W \times C$, a global pooling layer performs average pooling over the full spatial domain to compute a global feature vector $x^g = \frac{\sum_x \sum_y f_{x,y,c}}{H \times W}$.

2) Replication Layer: This layer replicates the input feature vector x^g with C dimensions at the centroid of every superpixel and produces a feature map $\{x_v^g\}$ of size $N_s \times C$.

3) Sampling Layer: Given a feature map of size $H \times W \times C$, this layer samples a feature vector at the centroid of every superpixel and assembles them into a feature map of size $N_s \times C$.

4) Color Histogram Layer: This layer computes a contextual color histogram feature with D dimensions at the centroid of every superpixel. These color histogram features are assembled into a feature map of size $N_s \times D$. The details of contextual color histogram computation are elaborated in Section IV-B.

5) Enhancement Layer: This layer predicts the enhanced colors at all pixels within every superpixel s_v. For each pixel p_j in s_v, if the original color at p_j is c_j, the enhanced color is computed as the product of the predicted color transform $\phi(\theta, x_v)$ and the quadratic color basis vector $V(c_j)$.

6) Discussion: Inspired by the design of feature descriptors in [1], we use the combination of the global feature, contextual features and the pixel feature to predict the color transform. However, there exist significant differences between our features and those in [1]. While various types of low-level image statistics (e.g. *intensity distritbuion, scene brightness, equalization curves*) are used as image global features in [1], our global feature is computed with a global pooling layer which spatially averages deep features from the last convolutional layer of a fully convolutional network. The resulting global feature is shown to be discriminative with respect to the semantics of image contents (e.g. an indoor portrait vs. an outdoor landscape) [33]. We use t-SNE [34] to compute a two-dimensional embedding of global features. This embedding arranges images with similar global features close to each other in a plane. A visualization of the resulting

Fig. 4. Visualization of global feature where images are displayed exactly at their embedded location.

spatial layout is shown in Figure 4. To form a contextual feature descriptor for representing the local surrounding of a superpixel, Yan *et al.* [1] rely on a traditional scene parser [28] to label background image regions, and a set of cascaded object detectors [29] to detect and classify foreground objects into a small set of predefined categories. By merging scene parsing results and object detections into a semantic label map, they compute label histograms using a multiscale spatial pooling scheme. However, both the scene parser and the object detectors they use have been substantially outperformed by recent deep CNN models [2], [3], [35], [36]. Furthermore, in contrast to the continuous deep CNN features used in our contextual features, their category-oriented discrete label map is more prone to quantization errors, which can lead to severe artifacts in the final results.

Moreover, compared to our fully convolutional network with an efficient GPU implementation, the scene parser and object detectors they use are at least one order of magnitude slower, and cannot be seamlessly integrated with their multilayer Perceptron network. In contrast, our entire deep network performs photo stylization in an end-to-end manner without any external dependency.

B. Video Stylization

To make our deep network trained on image exemplars perform video stylization, we choose to represent the video with temporal superpixels (TSPs) [5] (Section VI-A). TSPs pay particular attention to the temporal dimension, and explicitly model the motion flow between frames in a probabilistic generative framework. As a result, they can track image regions and object parts over a period of time.

To exploit the tracking ability of TSPs in maintaining the temporal coherence in the enhanced video, we associate each TSP with a single color transform (Section VI-B). To reuse our trained deep network for predicting per-TSP color transforms, we can project a TSP onto video frames to obtain a sequence of temporally adjacent superpixels. As described in Section III-A, we could predict color transforms associated with those superpixels and aggregate them to obtain the per-TSP color transform. Since each TSP only needs one color transform, it is not necessary to predict a color transform for every projected superpixel. To minimize the number of frames used for feature extraction, we propose a greedy algorithm to choose a small set of representative frames. In Section VII, we empirically demonstrate that this technique clearly reduces the computational cost without compromising the visual quality.

IV. Feature Description

An image is decomposed into a set of superpixels with the graph-based segmentation algorithm in [37], and we seek to predict a single color transform $\phi(\theta, x_v)$ for every superpixel s_v using the feature vector x_v described in this section. This feature vector consists of three components, namely the global feature, the contextual feature and the pixel feature.

A. Global Features

The overall image content affects how professional photographers adjust the image. We employ a fully convolutional network to extract image global features. FCN is introduced in [2] for semantic image segmentation. It can be set up by transforming a convolutional neural network (CNN). The CNN can be pretrained on a large-scale dataset for image classification, such as ImageNet [38], to learn discriminative feature representations, which are generically useful for a set of related tasks [39]–[41]. We take the VGG-16 network [42] pretrained on ImageNet images from 1000 object categories. This network consists of 5 groups of convolutional layers (*conv1-conv5*), 5 pooling layers (*pool1-pool5*) and 2 fully connected layers (*fc6* and *fc7*). We replace layers *fc6* and *fc7* with new convolutional layers *conv6* and *conv7* while kernel parameters in *conv6* and *conv7* are copied from *fc6* and *fc7*, respectively. An input image of size $H \times W \times 3$ is sent into this FCN and a feature map of size $H^g \times W^g \times 4096$ is extracted from the deepest layer *conv7* which has a sufficiently large receptive field of size 224×224. The spatial dimensions of the feature map is reduced by a factor of 32 due to the 5 pooling layers. Thus $H^g = \frac{H}{32}$, and $W^g = \frac{W}{32}$. We perform global average pooling to obtain a 4096D global feature, whose dimensionality is further reduced to 200 using PCA to prevent overfitting during the network training stage. The resulting 200D feature x_v^g is replicated at the centroids of all superpixels.

B. Contextual Features

As photographers apply spatially varying adjustments to various regions in a photo according to the local contents and their appearances in these regions, we introduce semantic features over two different scales and color histogram features to differentiate among local contexts.

1) Semantic Features: Apart from the FCN taking the input image in its original size to extract the global feature, two more FCNs with identical weights are used to extract contextual semantic features. On the basis of the FCN used for global feature extraction, we make two crucial modifications to the two FCNs used here. First, we reduce the stride of pooling

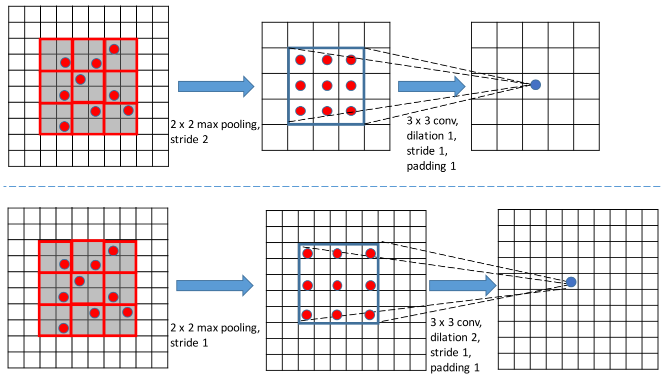


Fig. 5. Dilated convolution. **Top**: The spatial resolution of feature maps is reduced by half after a max-pooling operation with stride 2, and the subsequent 3 × 3 convolution is not dilated (i.e. dilation = 1). **Bottom**: The stride of max-pooling is reduced to 1. The spatial resolution is preserved and the convolution is dilated to have input stride 2 (i.e. dilation = 2).

kernels in layers *pool4* and *pool5* from 2 to 1, and thus the spatial resolution of feature maps is not reduced across these layers. As a result, the overall spatial resolution is merely reduced by a factor of 8. Feature maps with high spatial resolutions have been shown to be vital in attaining superior performance in image processing tasks [2], [43]. Changing the kernel stride at a pooling layer alone makes the following convolutional layer have a different receptive field and extract meaningless features. We use *dilated convolution* [44] to increase the input stride of feature maps for layers *conv5* and *conv6* by factors of 2 and 4 respectively to compensate the change of pooling kernels in *pool4* and *pool5* (Figure 5). We further fine-tune this transformed VGG16 network on the Places2 scene classification dataset [45], which makes deep features produced by *conv7* more discriminative across different semantic regions.

Second, besides extracting semantic features at the original image resolution, we also upsample the input image by a factor of 2 and feed the upsampled image into one of the aforementioned FCNs to extract features from the *conv7* layer, the size of whose receptive field is reduced from 224 × 224 to 112 × 112. This two-scale scheme is better suited for context modeling. To ensure meaningful features are extracted at image boundaries, we apply 112-pixel wide reflection padding to the upsampled images. Finally, if an input image of size $H^u \times W^u \times 3$ is sent into any of these modified FCNs, a feature map of size $\frac{H^u}{8} \times \frac{W^u}{8} \times 4096$ is extracted from layer *conv7*. To prevent overfitting, the 4096D feature at each pixel of this feature map is reduced to a 800D feature using PCA. Thus, at the centroid of every superpixel, there are two concatenated contextual features, which are respectively sampled from the nearest pixels in layer *conv7* of these two FCNs. Thus, every superpixel has a 1600D contextual semantic feature.

2) Color Histogram Features: To represent the appearances of local contents, we also compute a color histogram feature by performing two-scale spatial pooling over a grid of cells, as illustrated in Figure 3. Inspired by spatial pyramid pooling [46], we place a 3 × 3 grid of cells at the centroid of every superpixel. For each channel in the CIELab color space, we compute a histogram over each of the 9 cells, each of which has 7 × 7 pixels, as well as over their bounding box, which has 21 × 21 pixels. Such spatial pooling makes our local color histograms more robust to spatial deformations. We use 32 bins in each histogram and concatenate all histograms into a 960D feature vector.

Due to the relatively large spatial extent of the grid used for histogram computation, color histograms for a superpixel near an object or region boundary may be heavily influenced by pixels on the other side of the boundary. Such color histograms can mislead our deep network to apply incorrect adjustments to the superpixel and further give rise to halo artifacts, especially when pixel colors on both sides of the boundary have significant differences. We mitigate this problem by altering pixel weights during histogram binning. Specifically, before computing color histograms at the centroid of a superpixel, the weight associated with a pixel in the aforementioned grid is set to the following bilateral coefficient,

$$w_i = e^{\frac{|c_i - c_v|^2}{\sigma_c}} e^{\frac{|p_i - p_v|^2}{\sigma_p}},$$

where c_v and p_v respectively denote the color and location of the centroid, c_i and p_i respectively denote the color and location of the pixel, and σ_c and σ_p represent the estimated standard deviations of colors and distances respectively. We further normalize all pixel weights and use such weights during histogram binning afterwards.

C. Pixel Features

Pixel colors represent high-resolution spatial variations. We represent pixel colors in the CIELab color space. The pixel feature of a superpixel is simply the 3D pixel color at the centroid of the superpixel.

V. Experimental Results on Image Stylization

A. Experimental Setup

As shown in Figure 2, we employ FCNs to extract image global features and contextual semantic features. All the features of a superpixel are concatenated and fed into a deep neural network with one input layer, two fully connected hidden layers, and one output layer. The number of neurons in the hidden layers are set empirically to 256, and the number of neurons in the output layer are set equal to the number of coefficients in the predicted color transform. Since we use quadratic color transforms, there are 30 neurons in the output layer, 10 for each of the three color channels.

At the training stage, only the weights associated with the fully connected layers are updated with the classic error back-propagation algorithm. In practice, each image is segmented into around 7000 superpixels, from each of which 10 pixels are sampled. Therefore, even if we only have 50 example image pairs for learning one specific style, the total number of training samples is still as large as 3.5 million. Such a size of the training set can largely eliminate the risk of overfitting. It typically takes a few hours to finish training the deep neural network on a training dataset with hundreds of image pairs. At the testing stage, we still segment each image into around 7000 superpixels and the features extracted for each superpixel at its centroid are shared among all pixels within the same

TABLE I

MEAN PER-PIXEL L^2 DISTANCES BETWEEN INPUT IMAGES AND GROUNDTRUTH ADJUSTED IMAGES AND MEAN PER-PIXEL L^2 TESTING ERRORS BETWEEN AUTOMATICALLY ADJUSTED RESULTS AND GROUNDTRUTH RESULTS

Style	Groundtruth L^2 distance	Global Transform	Yan et al. [1]	Our Method
Foreground Pop-out	13.86	6.60	7.08	6.39
Local Xpro	19.71	6.31	7.43	5.55
Watercolor	15.30	7.23	7.20	6.44

superpixel. The enhanced result of a novel testing image can be computed in just one forward pass through our deep network. It takes around 25 seconds to enhance an image 512-pixel wide.

B. Results on the Uniform Dataset

Here we report image stylization results from our deep network on the Uniform dataset introduced in [1]. This dataset includes 115 images and three local enhancement styles. In the following, we briefly review these three styles.

The first style, "**Foreground Pop-out**", increases the contrast and color saturation of foreground objects while decreasing the color saturation of background. In general, this style makes the foreground objects more salient and colorful while deemphasizing the background. The second style, "**Local Xpro**", is more complex compared to the previous effect. It generalizes the popular "cross processing" effect in a local manner. When creating this style, a photographer first defined multiple profiles in Photoshop, each of which is specifically tailored for a subset of semantic categories. All the profiles share a common series of operations, such as hue/saturation adjustment and brightness/contrast manipulation. Nonetheless, each profile defines a distinct set of adjustment parameters tailored for its corresponding semantic categories. Although the profiles roughly follow the "cross processing" style, the choice of local profiles and additional minor image editing operations were heavily influenced by the photographer's personal taste which can be naturally learned through exemplars. The third style, "**Watercolor**", gives viewers artistic impressions. It creates "brush" effects over the input image. All pixels inside the region covered by a single brush stroke share the same color. This mimics the "watercolor" painting style. This style also gives rise to complex spatially varying color adjustments.

We have trained our deep network to learn all three styles from this dataset. As in [1], for each style, 70 images are used for training and the remaining 45 images are used for testing. Fig 6 shows some testing results in these three styles. We have calculated the mean per-pixel L^2 distance in the CIELab color space between our enhanced images and the groundtruth results as well as between the input images and the groundtruth results. They are shown in Table I. We also add a baseline which is the L^2 distance between the ground truth and the result obtained with per-image best global quadratic color transform. It provides the theoretical lower error bound for image adjustment using a global quadratic color transform. Our method can achieve even lower numerical errors, which indicates the importance of spatially-varying local color transforms.

In this paper, we primarily compare our method against the one proposed by Yan *et al.* [1] under the same experimental setting. This is because the method in [1] is the most recent work capable of performing local semantics-aware color and tone stylization. It has been demonstrated in [1] that their method outperforms all other earlier relevant techniques. Comparison of numerical errors on testing images are shown in the fourth and fifth columns of Table I. Our method achieves lower numerical errors on all the three local enhancement styles.

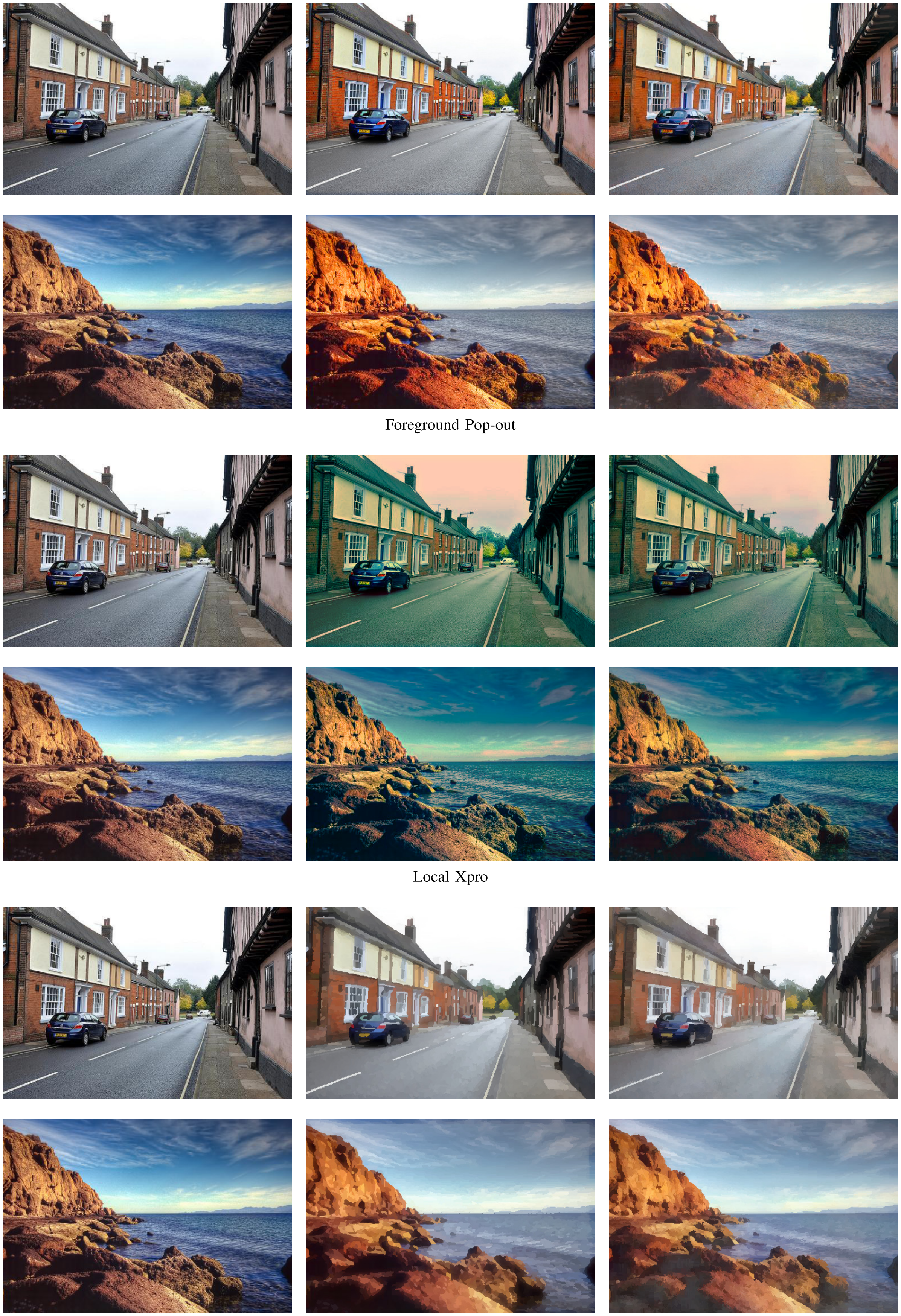
In addition to lower numerical errors, our method achieves clearly higher visual quality for the following reasons. First, their contextual feature is based on a label map generated from a scene parser [28] and object detectors [29], which tend to make unreliable discrete labeling decisions while our method avoids such discrete decisions by using continuous deep features. Therefore, their enhanced results are more likely to have artifacts due to incorrect labeling. One such example from the Foreground Popout style is shown in Fig 7, where 'sea' is mislabeled as 'mountain' with their method and the saturation of this mislabeled region is incorrectly increased. Second, their method requires the definition of a set of semantic categories. When this set is limited (20 categories used in [1]), their contextual feature would not work well on testing images containing object categories beyond this predefined set. In addition, the limited number of categories also limits the effectiveness of their feature. Another example from the Foreground Popout style is shown in Fig 8, where the saturation of 'building' is not increased with their method even though it is labeled correctly. In contrast, our FCN based architecture extracts contextual features without using a category set and is more robust. The color of the 'building' region is correctly adjusted with our method.

Another comparison has been conducted against the method proposed by Wang *et al.* [25], which tries to approximate complex spatially varying tone and color adjustments with piecewise polynomial functions based on low-level image statistics. Our method relies on more powerful features, which not only include low-level image statistics such as color histograms but also high-level contextual semantic information. An example is shown in Fig 9, where we can see that our enhanced result is much closer to the ground truth.

C. Effectiveness of Semantic and Color Histogram Features

Our contextual feature consists of two parts: semantic feature and color histogram feature. In this subsection, we demonstrate the importance of these individual features as well as their integration in our method.

We conduct experiments using four different contextual feature combinations, including concatenated semantic and color histogram features, semantic feature alone, color histogram feature alone and no contextual feature at all. The mean per-pixel L^2 testing errors in the CIELab color space are summarized in Table II. Without using any contextual features, the testing errors are 10.10, 9.76 and 8.45 on the Foreground Pop-out, Local Xpro and Watercolor styles respectively. These errors drop to 8.75, 8.45 and 7.72 respectively after adding

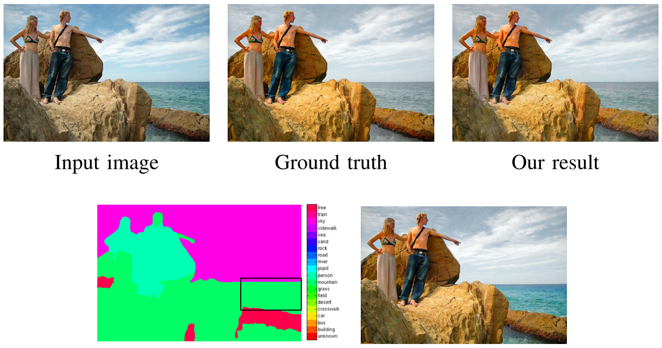


Foreground Pop-out

Local Xpro

Watercolor

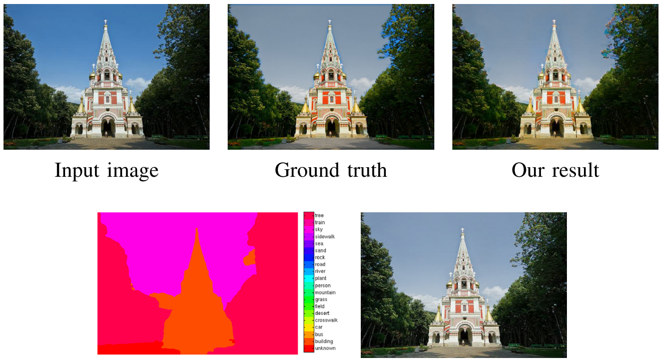
Fig. 6. Examples of local photo adjustment styles. **First column:** Input image. **Second column:** Ground truth. **Third column:** Our result.



Input image Ground truth Our result

Scene labels from [1] Result from [1]

Fig. 7. Comparison with the method in [1] on an example from the Foreground Popout style. Their method mislabels 'sea' as 'mountain' and the saturation of this mislabeled region is incorrectly increased. In contrast, our method produces much more robust visual results by using deep contextual and global features extracted using fully convolutional networks.



Input image Ground truth Our result

Scene labels from [1] Result from [1]

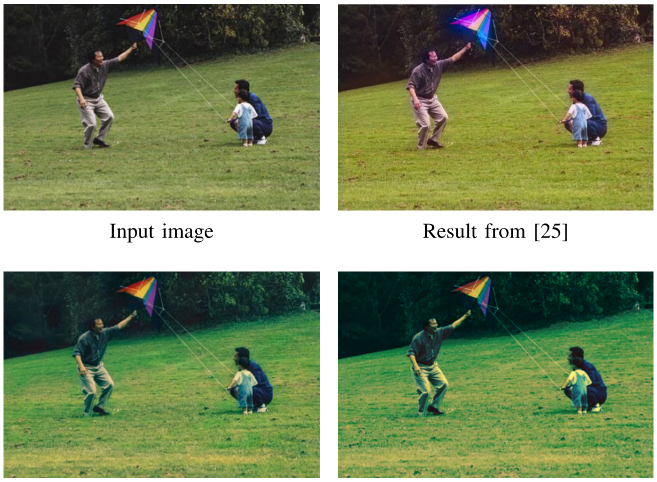
Fig. 8. Comparison with the method in [1] on another example from the Foreground Popout style. Although their method labels the 'building' region correctly, it still adjusts its color incorrectly, which reveals the limitation of their feature description. In contrast, our features give rise to an enhanced image closer to the ground truth.

semantic features and drop to 8.68, 8.60 and 7.66 respectively after adding color histogram features. These results numerically indicates the importance of these individual features. Moreover, the testing errors can be further reduced to 7.14, 7.19 and 6.78 after integrating these two features together, which indicates the necessity of using both features in our method.

We also provide an example of visual comparison in Figure 10. This example is from the Foreground Popout style. In this example, 'person' is classified as foreground and 'sky' and 'sea' are classified as background. We can see that the result obtained using both semantic and color histogram features is visually closer to the groundtruth result than those obtained using one type of contextual features only.

Bilateral Weights in Color Histogram Features:

Here we verify the effectiveness of bilateral weights in color histogram features. A comparison of enhanced results with and without using bilateral weights are shown in Fig 11. We can see that there are obvious halo artifacts around heads and shoulders when bilateral weights are not used, and such artifacts are suppressed when bilateral weights are used.



Input image Result from [25]

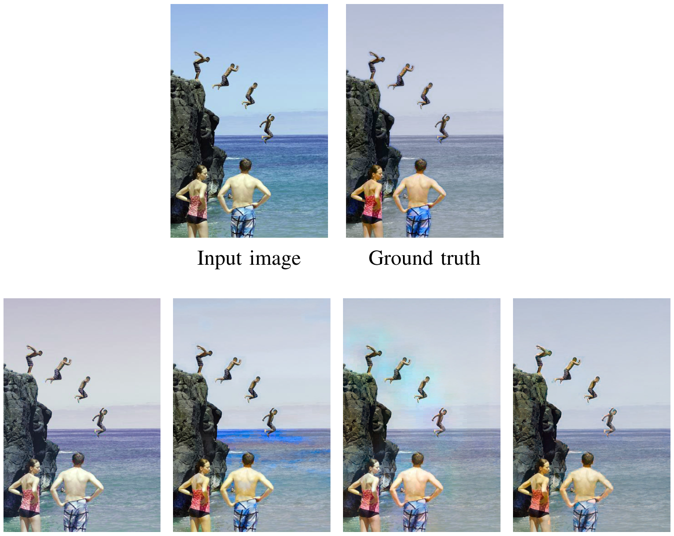
Our result Ground truth

Fig. 9. Comparison with the method in [25] on an example in the Local Xpro style. Our enhanced result is visually closer to the manually enhanced ground truth.

TABLE II

COMPARISON OF MEAN PER-PIXEL L^2 TESTING ERRORS ACHIEVED WITH DIFFERENT COMBINATIONS OF CONTEXTUAL SEMANTIC AND COLOR HISTOGRAM FEATURES

Style	Both features	Semantic features only	Color features only	Without context features
Foreground Pop-out	6.39	8.75	8.68	10.10
Local Xpro	5.55	8.45	8.60	9.76
Watercolor	6.44	7.72	7.66	8.45

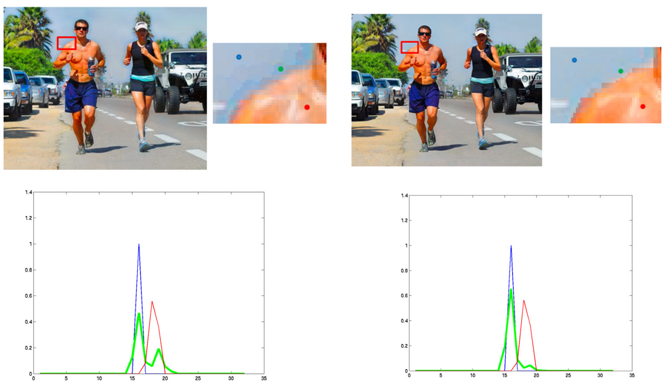


Input image Ground truth

Without context Color histogram Semantic feature With both features

Fig. 10. Comparison of visual results produced with different combinations of contextual semantic and color histogram features.

Let us take a closer look at the color histograms at three sample points, p_b (blue), p_g (green) and p_r (red), in the top images of Fig 11. When bilateral weights are not used (Fig 11(a)), the color histogram at p_g can be seen as a blended version of color histograms at p_b and p_r since p_g is close to a region boundary. However, p_g actually belongs



(a) (b)

Fig. 11. Histograms of channel 'a' in the CIELab color space at three pixels ('blue', 'green' and 'red') in the top image are shown in the bottom with corresponding colors. Note that the histogram at pixel 'green' looks more similar to that of pixel 'blue' after bilateral weights have been incorporated. (a) Histogram w/o bilateral weights. (b) Histogram with bilateral weights.

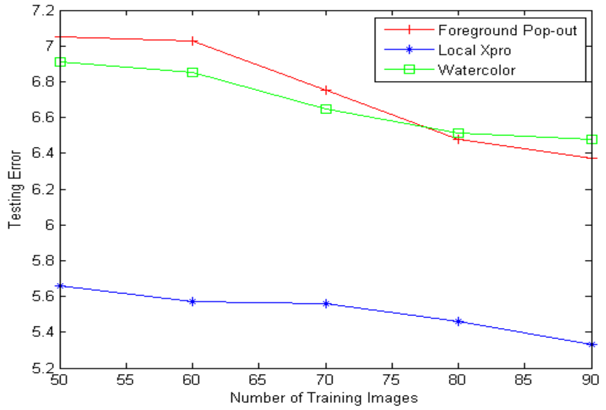
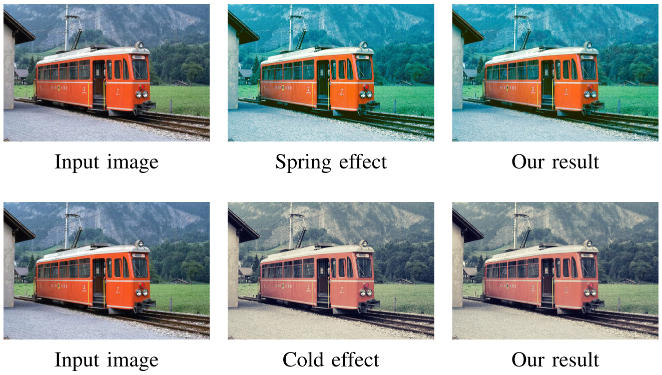


Fig. 12. Average testing error with respect to the number of training images.

to the 'sky' region, and adjustments applied to p_g should be similar to those applied to p_b. After using bilateral weights, we can see that the color histogram at p_g looks more similar to the color histogram at p_b, and it is likely that similar adjustments will be applied to both of them.

D. Number of Training Images

In the previous section, we use a fixed number of training and testing images as in [1] for fair comparison. To further verify the learning ability of our deep network, here we compare the performance of models obtained with different numbers of training images. The Uniform dataset has 115 images, from which 25 images are randomly selected as testing images. Then, we conduct a series of experiments which randomly choose an increasing number (50, 60, 70, 80 and 90) of images from the remaining images as the training set. We repeat each experiment in this series three times each using a different subset of randomly chosen images for training. The average testing error of the three trials with respect to the number of training images is shown in Fig. 12, where we can see that our network can achieve better performance with more training images. This is reasonable since larger training sets exhibit more content diversity and give rise to more accurate predictions of color transforms.



Input image Spring effect Our result

Input image Cold effect Our result

Fig. 13. Examples of global 'Spring' and 'Cold' styles. **Left**: Input image. **Middle**: Ground truth. **Right**: Our result.

E. Global Styles

Our deep network can readily learn global image adjustment styles as well. We asked a photographer to create additional stylistic effects which are saved as "action" files in Photoshop, each of which contains a series of operations such as saturation adjustment, tone adjustment, and curve tuning. An "action" is globally applied to the original images in the Uniform dataset [1] to form a set of image exemplars for a global enhancement style. It is also convenient for us to obtain from the Internet various stylistic "action" files shared by photo retouching enthusiasts. Here we demonstrate model training and testing for two global effects as examples.

The first global effect (called "Spring") gives viewers the feeling of vitality, and the second global effect (called "Cold") tends to make a photo look vintage. Fig 13 shows enhancement examples in these two styles. The mean per-pixel L^2 distance between the original images and their groundtruth enhanced images reaches 16.95 and 18.56 for the "Cold" and "Spring" styles, respectively. This indicates that these two effects make significant color changes to the original images. The testing errors of our trained models for these two styles are 1.88 and 4.13 respectively, which numerically demonstrates the strength of our method in learning such global effects. Comparing the middle column and the right column in Fig 13, we can see that the enhanced results from our model look nearly the same as the ground truth.

VI. Video Stylization

Our deep network trained on image exemplars can be extended to enhance artistic styles in videos as well. Naive video stylization in a frame-by-frame manner ignores the temporal coherence between adjacent frames, and is also inefficient. Therefore, we seek a more compact video representation to facilitate video stylization.

A. Temporal Superpixels

To generalize 2D superpixels used in image stylization for the purpose of video stylization, it is tempting to adopt the graph-based segmentation algorithm for decomposing a video into 3D supervoxels [47]. However, without explicitly modeling motions in a video, supervoxels cannot track image regions and object parts, and temporal coherence is hardly guaranteed. To overcome this difficulty, we represent a video as a set of temporal superpixels (TSPs), which are inferred from a probabilistic generative model explicitly considering motion flows between frames [5]. Moreover, TSPs are more uniform than supervoxels, which make them better suited for tracking image regions in a temporally coherent manner.

B. Frame Selection

A TSP can be projected onto video frames as a sequence of temporally adjacent 2D superpixels. As in image stylization, a single color transform is predicted for every TSP, and applied to all pixels within the TSP during video stylization. As a result, we only need to choose one 2D superpixel from the aforementioned sequence of projected superpixels. Our FCN can extract features at the centroid of this chosen superpixel and predict its color transform by processing the video frame containing this superpixel. The predicted color transform can then be used for enhancing all the pixels within the TSP.

On the basis of the above analysis, to reduce the computational cost, we seek to minimize the total number of video frames processed by the FCN while ensuring at least one of every TSP's projected 2D superpixels is included in the processed frames. We develop an iterative frame selection algorithm to solve the above problem in a greedy manner. At the beginning of our algorithm, all TSPs are unlabeled. During each iteration, we first choose the video frame intersecting the largest number of unlabeled TSPs, and then label these intersected TSPs. This process is repeated until all TSPs have been labeled. The resulting list of chosen frames will be processed by our FCN. In Section VII, we will demonstrate the advantage of our frame selection algorithm in the context of video stylization.

C. Guided Spatial Smoothing

Within each TSP, we only extract features and predict a color transform for one of the projected 2D superpixels, and the resulting color transform is used for enhancing all pixels in the TSP. This strategy works well in terms of maintaining temporal coherence. Nonetheless, for two spatially adjacent TSPs, if their chosen 2D superpixels for feature extraction do not lie on the same video frame, the extracted features as well as the predicted color transforms of these superpixels may not have spatial coherence and blocky artifacts may appear in the enhanced video.

To address this problem, we apply guided bilateral filtering [13] to predicted color transform coefficients. As Figure 15 shows, to obtain smoothed color transform coefficients for a superpixel p_1 in a video frame f, we first identify its immediately neighboring superpixels in the frame. The smoothed color transform coefficients ϕ_1^f for superpixel p_1 in frame f are computed with the following equation, $\phi_1^f = \sum_i w_{1i}\phi_i$, where ϕ_i denotes the predicted color transform coefficients at the i-th neighboring TSP,

$$w_{1i} = \exp\frac{|c_i - c_1|^2}{\sigma_c}\exp\frac{|p_i - p_1|^2}{\sigma_p}$$

represents the bilateral weight computed using the original input image as the guidance, and c_i, p_i are the color and position at the centroid of the i-th neighboring superpixel.

TABLE III

COMPARISON OF MEAN PER-PIXEL L^2 TESTING ERRORS OF OUR METHOD AND THE FRAME-BASED METHOD

Effect	Groundtruth L^2 distance	Our Method	Frame-based Method
Local Xpro	8.25	3.19	3.23
Foreground Pop-out	4.74	2.68	2.67
Golden	6.05	1.60	1.61
Cold	6.33	0.60	0.56
Spring	6.87	1.51	1.35

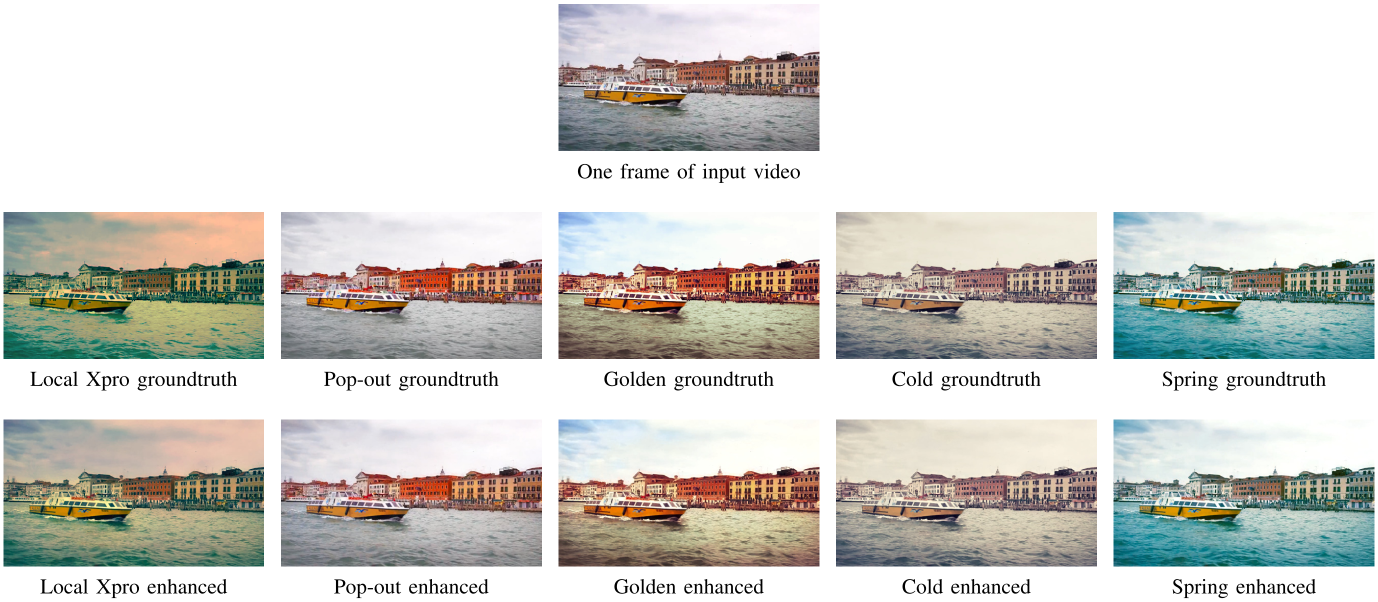
VII. EXPERIMENTAL RESULTS ON VIDEO STYLIZATION

A. Datasets and Statistics

For testing our video stylization algorithm, we train image enhancement models for five styles using our proposed deep network. These five styles include three local styles (Local Xpro, Foreground Pop-out, and a new local style called "Golden") and two global styles (Cold and Spring). The local style **Golden** is created by applying distinct stylistic adjustments to three types of semantic regions, namely natural objects, man-made objects, and sky regions. Natural objects include trees, land, mountains, people, etc. while man-made objects include buildings, cars, etc. Unlike the **Local Xpro** effect, where different adjustment profiles share a common series of operations and only differ in adjustment parameters, the Golden effect has distinct adjustment operations for each type of semantic regions. For example, the operations for natural objects include channel mixer and color curve tuning while the operations for man-made objects include gradient mapping and brightness/contrast manipulation.

We apply each of five enhancement styles to a set of five testing videos named "Fly", "Mountain", "City", "Railway" and "Motor". Each groundtruth stylized video is produced by manually applying local profiles to individual video frames independently. The mean per-pixel L^2 distance in the CIELab color space between original frames and groundtruth enhanced frames over all videos for each style is 8.25, 4.74, 6.05, 6.33 and 6.87, respectively, as shown in Table III. Once our trained models have been applied to the videos, the mean per-pixel L^2 distance between automatically enhanced frames and groundtruth enhanced frames drops to 3.19, 2.68, 1.60, 0.60 and 1.51, respectively. This numerically confirms the effectiveness of our overall video stylization approach based on models trained from images.

Fig 14 shows an example frame chosen from "City". We can see that our enhanced frames in the aforementioned five styles are visually similar to the groundtruth enhanced frames. A real challenge in video stylization is maintaining temporal


One frame of input video

Local Xpro groundtruth | Pop-out groundtruth | Golden groundtruth | Cold groundtruth | Spring groundtruth

Local Xpro enhanced | Pop-out enhanced | Golden enhanced | Cold enhanced | Spring enhanced

Fig. 14. Enhanced Local Xpro, Foreground Pop-out, Golden, Cold and Spring styles at a single frame from the "City" video. Middle row: Ground truth. Bottom row: enhanced result from our video stylization algorithm.

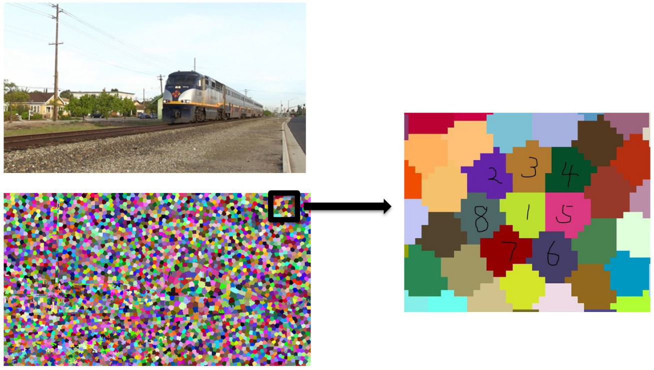


Fig. 15. Neighboring superpixels of a superpixel (labeled as '1') in a video frame. These neighboring superpixels are used in guided bilateral filtering of color transform coefficients within a single video frame.

coherence, which is one of the strengths of our proposed video stylization algorithm. Complete enhanced videos in these five styles can be found in the supplemental materials.

B. Effectiveness of TSP-Based Video Stylization

The most straightforward way to enhance a video using our learned model simply considers the video frames as independent images and enhance these frames individually. This implies that superpixels within each frame are also generated independently. In this paper, this approach is called the frame-based method. To demonstrate the effectiveness of our proposed video stylization algorithm, we compare our method against the frame-based method in terms of both numerical accuracy and visual quality. For numerical accuracy, we compute the mean per-pixel L^2 testing error between enhanced results and the ground truth for the five styles. As shown in the third and fourth columns of Table III, the numerical accuracy of our method only differs slightly from that of the frame-based method. However, as expected, the frame-based method cannot preserve temporal coherence, and the enhanced videos from our method have significantly higher visual quality. A visual comparison between enhanced videos from our method and those from the frame-based method can be found in the supplemental materials.

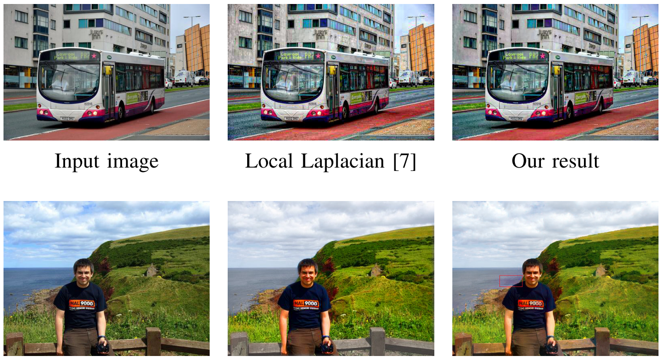
TABLE IV

COMPARISON OF THE NUMBER AND PERCENTAGE (IN PARENTHESES) OF CHOSEN FRAMES AT TWO LEVELS OF TSP GRANULARITY. RUNTIME IS SHOWN IN THE LAST TWO COLUMNS

Video	#frames	1k TSPs	3k TSPs	Runtime 1k TSPs	Runtime 3k TSPs
Fly	129	53(41%)	123(95%)	910s	1200s
City	155	58(37%)	114(74%)	1050s	1350s
Railway	120	82(68%)	98(82%)	990s	1110s
Motor	156	111(71%)	144(92%)	1340s	1500s
Mountain	202	112(55%)	180(89%)	1580s	1920s

We also compare the computational cost between our method and the frame-based method. Taking the video "Fly" as an example which has 129 frames, the frame-based method would need 3225 seconds since each frame takes 25 seconds. The processing time of our video stylization algorithm is significantly shorter because we do not have to extract features at all superpixels in all frames. The feature of a TSP is extracted only once in one of the selected frames and the computed quadratic color transform is shared among all pixels within the same TSP. In addition, we do not need to compute semantic feature maps for those unselected frames. The running times for video stylization are shown in Table IV.

The TSP algorithm [5] has parameters to control the approximate number of superpixels in each frame. To verify the effectiveness and flexibility of our frame selection scheme,



Input image Local Laplacian [7] Our result

Input image Pop-out Our result

Fig. 16. Two failure cases. **Top row:** Local Laplacian filter [7] is used to increase image details exaggeratedly. Our result produces insufficient detail increasement. **Bottom row:** One test result of Foreground Pop-out effect, which has color artifacts marked in the red box.

we produce TSPs at two scales, Seg_1000 and Seg_3000, which indicates there are around 1000 and 3000 superpixels in each frame. Table IV lists the number and percentage of frames chosen by our frame selection algorithm when a video is segmented at each of these two scales as well as the total number of frames in each video. The computational cost of our method decreases when the granularity of superpixels increases since there are less TSPs needed to extract features and less mapped frames needed to compute semantic feature map.

We also compare the mean per-pixel L^2 testing error between enhanced videos and the ground truth at these two scales. At 3000 superpixels per frame, the mean per-pixel L^2 error is 3.19, 2.68, 1.60, 0.60 and 1.51 for the Local Xpro, Foreground Popout, Golden, Cold and Spring styles, respectively. The mean per-pixel L^2 error only rises slightly to 3.27, 2.77, 1.65, 0.67 and 1.70 respectively at 1000 superpixels per frame. This comparison indicates that the numerical error will not increase significantly when we increase the granularity of superpixels within a certain range to reduce computational cost.

VIII. Conclusions and Discussion

In this paper, we have presented a novel deep learning architecture for exemplar-based image and video stylization, which learns local enhancement styles from image pairs. Our deep learning architecture consists of fully convolutional networks for automatic semantics-aware feature extraction and fully connected neural layers for adjustment prediction. Image stylization can be efficiently accomplished with a single forward pass through our deep network. To extend our deep network from image stylization to videos, we exploit temporal superpixels to facilitate the transfer of artistic styles from image exemplars to videos. Experiments on a number of datasets for image stylization as well as a diverse set of video clips demonstrate the effectiveness of our deep model.

A. Limitations

Our method has limitations. Detail manipulation is considered one type of image stylization. We use the local Laplacian filter [7] to exaggerate image details in the Uniform dataset. Such exaggerated results are used as training data for our method. The top row of Fig. 16 shows our trained model can only enhance details slightly but cannot achieve results similar to the ground truth. The reason is that our method only applies color transforms to individual pixels independently while detail manipulation needs to adjust local contrast among nearby pixels. The bottom row of Fig. 16 shows that if the predicted color transform at a certain superpixel (highlighted area) is incorrect, all the pixels within the same superpixel will be adjusted incorrectly, which is another limitation of our superpixel-based method.

B. Future Work

An interesting direction to extend this work of image and video stylization is to solve this problem using an end-to-end trainable network. That is we do not perform spatial subsampling using superpixels on the semantic feature maps produced by the fully convolutional networks. The fully connected layers used for predicting color transforms can be replaced with convolutional layers with 1 × 1 kernels. Such a network will be able to make pixel-level predictions. However, since it has a large number of parameters, overfitting can easily occur with insufficient training data.

References

[1] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep neural networks," *ACM Trans. Graph.*, vol. 35, no. 2, p. 11, 2016.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015, pp. 3431–3440.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. (2014). "Semantic image segmentation with deep convolutional nets and fully connected CRFs." [Online]. Available: https://arxiv.org/abs/1412.7062

[4] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 478–487.

[5] J. Chang, D. Wei, and J. W. Fisher, III, "A video representation using temporal superpixels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2051–2058.

[6] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 637–645, 2006.

[7] S. Paris, S. W. Hasinoff, and J. Kautz, "Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid," *ACM Trans. Graph.*, vol. 30, no. 4, p. 68, 2011.

[8] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, "Color harmonization," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 624–630, Jul. 2006.

[9] Y. Baveye, F. Urban, C. Chamaret, V. Demoulin, and P. Hellier, "Saliency-guided consistent color harmonization," in *Computational Color Imaging*. Berlin, Germany: Springer, 2013, pp. 105–118.

[10] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, no. 3, p. 67, 2008.

[11] K. Subr, C. Soler, and F. Durand, "Edge-preserving multiscale image decomposition based on local extrema," *ACM Trans. Graph.*, vol. 28, no. 5, p. 147, 2009.

[12] M. Son, Y. Lee, H. Kang, and S. Lee, "Art-photographic detail enhancement," *Comput. Graph. Forum*, vol. 33, no. 2, pp. 391–400, May 2014.

[13] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 664–672, Aug. 2004.

[14] H. Cho, H. Lee, H. Kang, and S. Lee, "Bilateral texture filtering," *ACM Trans. Graph.*, vol. 33, no. 4, p. 128, 2014.

[15] S. Bi, X. Han, and Y. Yu, "An L_1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition," *ACM Trans. Graph.*, vol. 34, no. 4, p. 78, 2015.

[16] X. An and F. Pellacini, "Appprop: All-pairs appearance-space edit propagation," *ACM Trans. Graph.*, vol. 27, no. 3, p. 40, 2008.

[17] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski, "Interactive local adjustment of tonal values," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 646–653, 2006.

[18] K. Dale, M. K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister, "Image restoration using online photo collections," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 2217–2224.

[19] B. Wang, Y. Yu, T.-T. Wong, C. Chen, and Y.-Q. Xu, "Data-driven image color theme enhancement," *ACM Trans. Graph.*, vol. 29, no. 6, p. 146, 2010.

[20] J. Yan, S. Lin, S. B. Kang, and X. Tang, "A learning-to-rank approach for image color enhancement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 2987–2994.

[21] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input/output image pairs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 97–104.

[22] S. B. Kang, A. Kapoor, and D. Lischinski, "Personalization of image enhancement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 1799–1806.

[23] N. Joshi, W. Matusik, E. H. Adelson, and D. J. Kriegman, "Personal photo enhancement using example images," *ACM Trans. Graph*, vol. 29, no. 2, pp. 1–15, 2010.

[24] J.-Y. Lee, K. Sunkavalli, Z. Lin, X. Shen, and I. S. Kweon, "Automatic content-aware color and tone stylization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2470–2478.

[25] B. Wang, Y. Yu, and Y.-Q. Xu, "Example-based image color and tone style enhancement," *ACM Trans. Graph.*, vol. 30, no. 4, p. 64, 2011.

[26] Y. Shih, S. Paris, F. Durand, and W. T. Freeman, "Data-driven hallucination of different times of day from a single outdoor photo," *ACM Trans. Graph.*, vol. 32, no. 6, p. 200, 2013.

[27] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2414–2423.

[28] J. Tighe and S. Lazebnik, "SuperParsing: Scalable nonparametric image parsing with superpixels," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 352–365.

[29] X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for generic object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 17–24.

[30] N. Bonneel, K. Sunkavalli, S. Paris, and H. Pfister, "Example-based video color grading," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–39, 2013.

[31] S. Xue, A. Agarwala, J. Dorsey, and H. Rushmeier, "Learning and applying color styles from feature films," *Comput. Graph. Forum*, vol. 32, no. 7, pp. 255–264, 2013.

[32] M. Ruder, A. Dosovitskiy, and T. Brox. (2016). "Artistic style transfer for videos." [Online]. Available: https://arxiv.org/abs/1604.08610

[33] W. Liu, A. Rabinovich, and A. C. Berg. (2015). "ParseNet: Looking wider to see better." [Online]. Available: https://arxiv.org/abs/1506.04579

[34] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[36] W. Liu *et al.* (2015). "SSD: Single shot multibox detector." [Online]. Available: https://arxiv.org/abs/1512.02325

[37] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, Sep. 2004.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.

[39] J. Donahue *et al.* (2013). "DeCAF: A deep convolutional activation feature for generic visual recognition." [Online]. Available: https://arxiv.org/abs/1310.1531

[40] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[41] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1–9.

[42] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

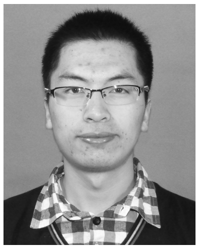
[43] Z. Yan, H. Zhang, Y. Jia, T. Breuel, and Y. Yu. (2016). "Combining the best of convolutional layers and recurrent layers: A hybrid network for semantic segmentation." [Online]. Available: https://arxiv.org/abs/1603.04871

[44] F. Yu and V. Koltun. (2015). "Multi-scale context aggregation by dilated convolutions." [Online]. Available: https://arxiv.org/abs/1511.07122

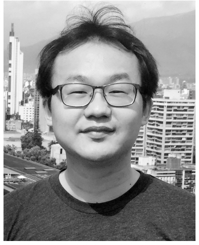
[45] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. (2016). "Places: An image database for deep scene understanding." [Online]. Available: https://arxiv.org/abs/1610.02055

[46] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. 2006, pp. 2169–2178.

[47] C. Xu and J. J. Corso, "Evaluation of super-voxel methods for early video processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 1202–1209.



Feida Zhu received the B.Eng. degree in automation from the University of Science and Technology of China, Hefei, China, in 2014. He is currently pursuing the Ph.D. degree with the Department of Computer Science, The University of Hong Kong, Hong Kong. His research interests include image processing, computer vision, and machine learning.



Zhicheng Yan (M'15) received the B.Eng. degree in software engineering and the M.Eng. degree in computer engineering from Zhejiang University in 2007 and 2010, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign in 2016. He is currently a Research Scientist with the Computer Vision Team, Facebook Inc. His research interests include computer vision, computer graphics, and visualization.

Jiajun Bu (M'06) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, China, in 1995 and 2000, respectively. He is currently a Professor with the College of Computer Science, Zhejiang University. His research interests include multimedia, embedded system, and data mining. He is a member of the ACM.

Yizhou Yu (SM'12) received the Ph.D. degree from the University of California at Berkeley in 2000. He was a Faculty Member with the University of Illinois at Urbana–Champaign for 12 years. He is currently a Professor with The University of Hong Kong. His current research interests include deep learning methods for visual computing, geometric computing, video analytics, and biomedical data analysis. He was a recipient of the 2002 U.S. National Science Foundation CAREER Award and the 2007 NNSF China Overseas Distinguished Young Investigator Award. He has served on the Editorial Board of the IET Computer Vision, the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, *The Visual Computer*, and the *International Journal of Software and Informatics*. He has also served on the Program Committee of many leading international conferences, including SIGGRAPH, SIGGRAPH Asia, and International Conference on Computer Vision.