

Automatic Photo Adjustment Using Deep Neural Networks

Zhicheng Yan

University of Illinois at Urbana Champaign

Hao Zhang[†]

Carnegie Mellon University

Baoyuan Wang

Microsoft Research

Sylvain Paris

Adobe Research

Yizhou Yu

The University of Hong Kong and University of Illinois at Urbana Champaign

Photo retouching enables photographers to invoke dramatic visual impressions by artistically enhancing their photos through stylistic color and tone adjustments. However, it is also a time-consuming and challenging task that requires advanced skills beyond the abilities of casual photographers. Using an automated algorithm is an appealing alternative to manual work but such an algorithm faces many hurdles. Many photographic styles rely on subtle adjustments that depend on the image content and even its semantics. Further, these adjustments are often spatially varying. Because of these characteristics, existing automatic algorithms are still limited and cover only a subset of these challenges. Recently, deep machine learning has shown unique abilities to address hard problems that resisted machine algorithms for long. This motivated us to explore the use of deep learning in the context of photo editing. In this paper, we explain how to formulate the automatic photo adjustment problem in a way suitable for this approach. We also introduce an image descriptor that accounts for the local semantics of an image. Our experiments demonstrate that our deep learning formulation applied using these descriptors successfully capture sophisticated photographic styles. In particular and unlike previous techniques, it can model local adjustments that depend on the image semantics. We show on several examples that this yields results that are qualitatively and quantitatively better than previous work.

Categories and Subject Descriptors: I.4.3 [Image Processing and Computer Vision]: Enhancement; I.4.10 [Image Processing and Computer Vision]: Representation—*Statistical*

Authors' email addresses: zyan3@illinois.edu, hao@cs.cmu.edu, baoyuanw@microsoft.com, sparis@adobe.com, yizhouy@acm.org.

[†]This work was conducted when Hao Zhang was an intern at Microsoft Research.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 0730-0301/YYYY/15-ARTXXX \$10.00

DOI 10.1145/XXXXXX.YYYYYYYY

<http://doi.acm.org/10.1145/XXXXXX.YYYYYYYY>

Additional Key Words and Phrases: Color Transforms, Feature Descriptors, Neural Networks, Photo Enhancement

1. INTRODUCTION

With the prevalence of digital imaging devices and social networking, sharing photos through social media has become quite popular. A common practice in this type of photo sharing is artistic enhancement of photos by various Apps such as Instagram. In general, such photo enhancement is artistic because it not only tries to correct photographic defects (under/exposure, poor contrast, etc.) but also aims to invoke dramatic visual impressions by stylistic or even exaggerated color and tone adjustments. Traditionally, high-quality enhancement is usually hand-crafted by a well-trained artist through extensive labor.

In this work, we study the problem of learning artistic photo enhancement styles from image exemplars. Specifically, given a set of image pairs, each representing a photo before and after pixel-level tone and color enhancement following a particular style, we wish to learn a computational model so that for a novel input photo we can apply the learned model to automatically enhance the photo following the same style.

Learning a high-quality artistic photo enhancement style is challenging for several reasons. First, photo adjustment is often a highly empirical and perceptual process that relates the pixel colors in an enhanced image to the information embedded in the original image in a complicated manner. Learning an enhancement style needs to extract an accurate quantitative relationship underlying this process. This quantitative relationship is likely to be complex and highly nonlinear especially when the enhancement style requires spatially varying local adjustments. It is nontrivial to learn a computational model capable of representing such a complicated relationship accurately, and large-scale training data is likely to be necessary. Therefore, we seek a learning model scalable with respect to both the feature dimension and data size and efficiently computable with high-dimensional, large-scale data.

Second, an artistic enhancement is typically semantics-aware. An artist does not see individual pixels; instead he/she sees semantically meaningful objects (humans, cars, animals, etc.) and determines the type of adjustments to improve the appearance of the objects. For example, it is likely that an artist pays more attention to improve the appearance of a human figure than a region of sky in the same



Fig. 1. An example of our semantics-aware photo enhancement style, which extends the “cross processing” effect in a local manner. **Left:** input image; **Middle:** enhanced image by our deep learning based automatic approach; **Right:** groundtruth image manually enhanced by a photographer, who applied different adjustment parameters in different semantic regions. See more results of such effects in Section 7.1 and the supplemental materials.

photo. We would like to incorporate this semantics-awareness in our learning problem. One challenge is the representation of semantic information in learning so that the learned model can perform image adjustments according to the specific content as human artists do.

We present an automatic photo enhancement method based on deep machine learning. This approach has recently accumulated impressive successes in domains such as computer vision and speech analysis for which the semantics of the data plays a major role, e.g., [Vincent et al. 2008; Krizhevsky et al. 2012]. This motivated us to explore the use of this class of techniques in our context. To address the challenges mentioned above, we cast exemplar-based photo adjustment as a regression problem, and use a Deep Neural Network (DNN) with multiple hidden layers to represent the highly nonlinear and spatially varying color mapping between input and enhanced images. A deep neural network (DNN) is a universal approximator that can represent arbitrarily complex continuous functions [Hornik et al. 1989]. It is also a compact model which is readily scalable with respect to high-dimensional, large-scale data.

Feature design is a key issue that can significantly affect the effectiveness of DNN. To make sure the learned color mapping responds to complex color and semantic information, we design informative yet discriminative feature descriptors that serve as the input to the DNN. For each input image pixel, its feature descriptor consists of three components, which reflect respectively the statistical or semantic information at the pixel, contextual, and global levels. The global feature descriptor is based on global image statistics, whereas the context feature descriptor is based on semantic information extracted from a large neighborhood around the pixel. Understanding image semantics has been made possible with recent advances in scene understanding and object detection. We use existing algorithms to annotate all input image pixels and the semantics information from the annotated images are incorporated into a novel context feature descriptor.

Contributions. In summary, our proposed photo enhancement technique has the following contributions.

—It introduces the first automatic photo adjustment framework based on deep neural networks. A variety of normal and artistic photo enhancement styles can be achieved by training a distinct model for each enhancement style. The quality of our results is superior to that of existing methods.

—Our framework adopts informative yet discriminative image feature descriptors at the pixel, contextual and global levels. Our context descriptor exploits semantic analysis over multiscale spa-

tial pooling regions. It has achieved improved performance over a single pooling region.

—Our method also includes an effective algorithm for choosing a representative subset of photos from a large collection so that a photo enhancement model trained over the chosen subset can still produce high-quality results on novel testing images.

While a contribution of our work is the application of deep machine learning in a new context, we use a standard learning procedure and do not claim any contribution in the design of the learning algorithm itself. Similarly, while we propose a possible design for semantic context descriptor, and demonstrate its effectiveness, a comprehensive exploration of the design space for such descriptors is beyond the scope of this paper.

Complete source codes and datasets used by our system are publicly available on Github¹.

2. RELATED WORK

Traditional image enhancement rules are primarily determined empirically. There are many software tools to perform fully automatic color correction and tone adjustment, such as Adobe Photoshop, Google Auto Awesome, and Microsoft Office Picture Manager. In addition to these tools, there exists much research on either interactive [Lischinski et al. 2006; An and Pellacini 2008] or automatic [Bae et al. 2006; Cohen-Or et al. 2006] color and tone adjustment. Automatic methods typically operate on the entire image in a global manner without taking image content into consideration. To address this issue, Kaufman et al. [2012] introduces an automatic method that first detects semantic content, including faces, sky as well as shadowed salient regions, and then applies a sequence of empirically determined steps for saturation, contrast as well as exposure adjustment. However, the limit of this approach is that output style is hard-coded in the algorithm and cannot be easily tuned to achieve a desired style. In comparison and as we shall see, our data-driven approach can easily be trained to produce a variety of styles. Further, these techniques rely on a fixed pipeline that is inherently limited in its ability to achieve user-preferred artistic enhancement effects, especially the exaggerated and dramatic ones. In practice, a fixed-pipeline technique works well for a certain class of adjustments and only produces approximate results for effects outside this class. For instance, Bae et al. [2006] do well with tonal global transforms but

¹<https://github.com/stephenyan1984/dl-image-enhance>
https://github.com/stephenyan1984/cuda_convnet_plus

do not model local edits, and Kaufman et al. [2012] perform well on a predetermined set of semantic categories but does not handle elements outside this set. In comparison, deep learning provides a universal approximator that is trained on a per-style basis, which is key to the success of our approach.

Another line of research for photo adjustment is primarily data-driven. Learning based image enhancement [Kang et al. 2010; Joshi et al. 2010; Caicedo et al. 2011; Bychkovsky et al. 2011] and image restoration [Dale et al. 2009] have shown promising results and therefore received much attention. Kang et al. [2010] found that image quality assessment is actually very much personalized, which results in an automatic method for learning individual preferences in global photo adjustment. Bychkovsky et al. [2011] introduces a method based on Gaussian processes for learning tone mappings according to global image statistics. Since these methods were designed for global image adjustment, they do not consider local image contexts and cannot produce spatially varying local enhancements. Wang et al. [2011] proposes a method based on piecewise approximation for learning color mapping functions from exemplars. It does not consider semantic or contextual information either. In addition, it is not fully automatic, and relies on interactive soft segmentation. It is infeasible for this technique to automatically enhance a collection of images. In comparison, this paper proposes a scalable framework for learning user-defined complex enhancement effects from exemplars. It explicitly performs generic image semantic analysis, and its image enhancement models are trained using feature descriptors constructed from semantic analysis results.

Hwang et al. [2012] proposes a context-aware local image enhancement technique. This technique first searches for the most similar images and then the most similar pixels within them, and finally apply a combination of the enhancement parameters at the most similar pixels to the considered pixel in the new test image. With a sufficiently large image database, this method works well. But in practice, nearest-neighbor search requires a fairly large training set that is challenging to create and slow to search, thereby limiting the scalability of this approach. Another difference with our approach is that, to locate the most similar pixels, this method uses low- and mid-level features (i.e., color and SIFT) whereas we also consider high-level semantics. We shall see in the result section that these differences have a significant impact on the adjustment quality in several cases.

3. A DEEP LEARNING MODEL

Let us now discuss how we cast exemplar-based photo adjustment as a regression problem, and how we set up a DNN to solve this regression problem. A photo enhancement style is represented by a set of exemplar image pairs $\Lambda = \{I^k, J^k\}_{k=1}^m$, where I^k and J^k are respectively the images before and after enhancement. Our premise is that there exists an intrinsic color mapping function \mathcal{F} that maps each pixel's color in I^k to its corresponding pixel's color in J^k for every k . Our goal is to train an approximate function $\tilde{\mathcal{F}}$ using Λ so that $\tilde{\mathcal{F}}$ may be applied to new images to enhance the same style there. For a pixel p_i in image I^k , the value of $\tilde{\mathcal{F}}$ is simply the color of image J^k at pixel p_i , whereas the input of $\tilde{\mathcal{F}}$ is more complex because $\tilde{\mathcal{F}}$ depends on not only the color of p_i in I^k but also additional local and global information extracted from I^k , thus we formulate $\tilde{\mathcal{F}}$ as a parametric function $\tilde{\mathcal{F}}(\Theta, x_i)$, where Θ represents the parameters and x_i represents the feature vector at p_i that encompasses the color of p_i in I^k as well as additional local and global information. With this formulation, training the function

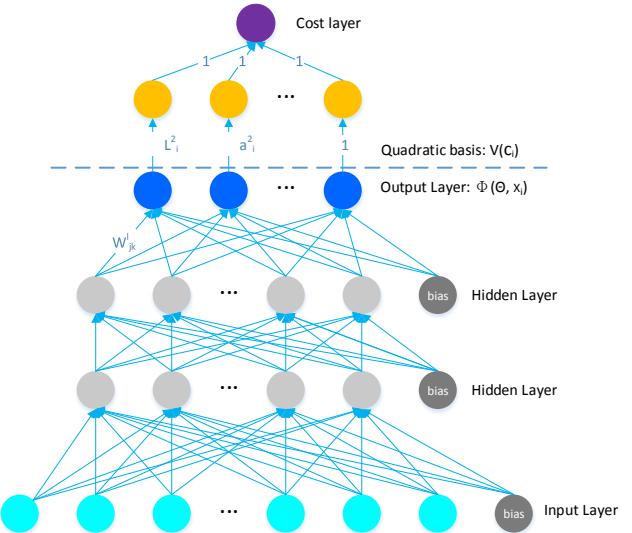


Fig. 2. The architecture of our DNN. The neurons above the dash line indicate how we compute the cost function in (3). Note that the weights for the connections between the blue neurons and the yellow neurons are just the elements of the quadratic color basis, and the activation function in the yellow and purple neurons is the identity function. During training, error backpropagation starts from the output layer, as the connection weights above the dash line have already been fixed.

$\tilde{\mathcal{F}}$ using Λ becomes computing the parameters Θ from training data Λ through nonlinear regression.

High-frequency pixelwise color variations are difficult to model because they force us to choose a mapping function which is sensitive to high-frequency details. Such a mapping function often leads to noisy results in relatively smooth regions. To tackle this problem we use a color basis vector $V(c_i)$ at pixel p_i to rewrite $\tilde{\mathcal{F}}$ as $\tilde{\mathcal{F}} = \Phi(\Theta, x_i)V(c_i)$, which expresses the mapped color, $\tilde{\mathcal{F}}$, as the result of applying the color transform matrix $\Phi(\Theta, x_i)$ to the color basis vector $V(c_i)$. $V(c_i)$ is a vector function taking different forms when it works with different types of color transforms. In this paper we work in the CIE Lab color space, and the color at p_i is $c_i = [L_i a_i b_i]^T$ and $V(c_i) = [L_i a_i b_i 1]^T$ if we use 3x3 affine color transforms. If we use 3x10 quadratic color transforms, then $V(c_i) = [L_i^2 a_i^2 b_i^2 L_i a_i L_i b_i a_i b_i L_i a_i b_i 1]$. Since the per-pixel color basis vector $V(c_i)$ varies at similar frequencies as pixel colors, it can absorb much high-frequency color variation. By factorizing out the color variation associated with $V(c_i)$, we can let $\Phi(\Theta, x_i)$ focus on modeling the spatially smooth but otherwise highly nonlinear part of $\tilde{\mathcal{F}}$.

We learn $\Phi(\Theta, x_i)$ by solving the following least squares minimization problem defined over all training pixels sampled from Λ :

$$\arg \min_{\Phi \in \mathcal{H}} \sum_i^n \| \Phi(\Theta, x_i)V(c_i) - y_i \|^2, \quad (1)$$

where \mathcal{H} represents the function space of $\Phi(\Theta, x_i)$ and n is the total number of training pixels. In this paper, we represent $\Phi(\Theta, x_i)$ as a DNN with multiple hidden layers.



Fig. 3. (Left) Input image, and (Right) visualization of its per-pixel quadratic color transforms, $\Phi(\Theta, x_i)$, each of which is a 3×10 matrix. Each image on the right visualizes one coefficient in this matrix at all pixel locations. Coefficients are linearly mapped to [0,1] in each visualization image for better contrast. This visualization illustrates two properties of the quadratic color transforms: 1) they are spatially varying and 2) they are smooth with much high-frequency content suppressed.

3.1 Neural Network Architecture and Training

Our neural network follows a standard architecture that we describe below for the sake of completeness.

Multi-layer deep neural networks have proven to be able to represent arbitrarily complex continuous functions [Hornik et al. 1989]. Each network is an acyclic graph, each node of which is a neuron. Neurons are organized in a number of layers, including an input layer, one or more hidden layers, and an output layer. The input layer directly maps to the input feature vector, i.e. x_i in our problem. The output layer maps to the elements of the color transform, $\Phi(\Theta, x_\nu)$. Each neuron within a hidden layer or the output layer takes as input the responses from all the neurons in the preceding layer. Each connection between a pair of neurons is associated with a weight. Let us denote v_j^l as the output of the j -th neuron in the l -th layer. Then v_j^l is expressed as follows:

$$v_j^l = g \left(w_{j0}^l + \sum_{k>0} w_{jk}^l v_k^{l-1} \right) \quad (2)$$

where w_{jk}^l is the weight associated with the connection between the j -th neuron in the l -layer and the k -th neuron in the $(l-1)$ -th layer, and $g(z)$ is an activation function which is typically nonlinear. We choose the rectified linear unit (ReLU) [Krizhevsky et al. 2012], $g(z) = \max(0, z)$, as the activation function in our networks. Compared with other widely used activation functions, such as the hyperbolic tangent, $g(z) = \tanh(z) = 2/(1 + e^{-2z}) - 1$, or the sigmoid, $h(x) = (1 + e^{-x})^{-1}$, ReLU has a few advantages, including inducing sparsity in the hidden units and accelerating the convergence of the training process. Note that there is no nonlinear activation function for neurons in the output layer. The output of a neuron in the output layer is only a linear combination of its inputs from the preceding layer. Figure 2 shows the overall architecture, which has two extra layers (yellow and purple neurons) above the output layer for computing the product between the color transform and the color basis vector. Given a neural network architecture for color mapping, \mathcal{H} in (1) should be the function space spanned by all neural networks with the same architecture but different weight parameters Θ .

Once the network architecture has been fixed, given a training dataset, we use the classic error backpropagation algorithm to train the weights. In addition, we apply the *Dropout* training strategy [Krizhevsky et al. 2012; Hinton et al. 2012], which has been shown very useful for improving the generalization capability.

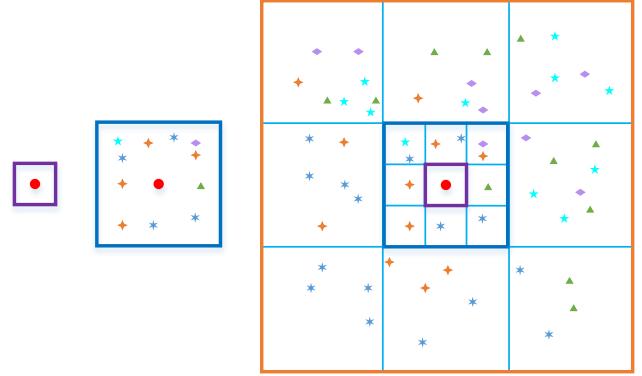


Fig. 4. Our multiscale spatial pooling schema. In each pooling region, we compute a histogram of semantic categories. The shown three-scale scheme has $9 \times 2 + 1 = 19$ pooling regions. In our experiments, we use a four-scale scheme with 28 pooling regions.

We set the output of each neuron in the hidden layers to zero with probability 0.5. Those neurons that have been “dropped out” in this way do not contribute to the forward pass and do not participate in error backpropagation. Our experiments show that adding *Dropout* during training typically reduces the relative prediction error on testing data by 2.1%, which actually makes a significant difference in the visual quality of the enhanced results.

Figure 3 visualizes the per-pixel quadratic color transforms, $\Phi(\Theta, x_i)$, generated by a trained DNN for one example image. We can see that the learned color mappings are smooth in most of the local regions.

4. FEATURE DESCRIPTORS

Our feature descriptor (x_i) at a sample pixel p_i serves as the input layer in the neural network. It has three components, $x_i = (x_i^p, x_i^c, x_i^g)$, where x_i^p represents pixelwise features, x_i^c represents contextual features computed for a local region surrounding p_i , and x_i^g represents global features computed for the entire image where p_i belongs. The details about these three components follow.

4.1 Pixelwise Features

Pixelwise features reflect high-resolution pixel-level image variations, and are indispensable for learning spatially varying photo

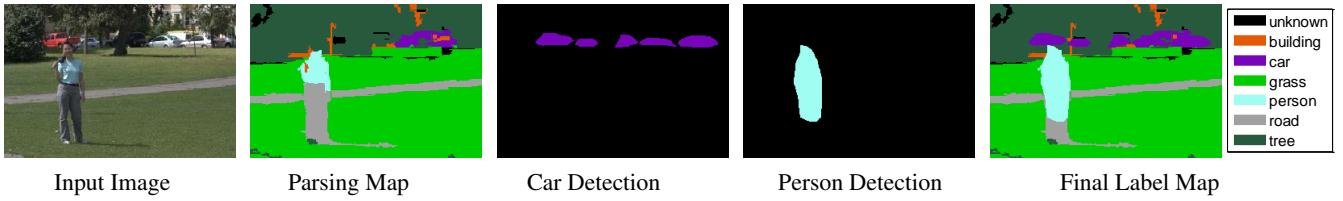


Fig. 5. Pipeline for constructing our semantic label map. Starting with an input image, we first perform scene parsing to obtain a parsing map, then we run object detectors to obtain a detection map for each object category (i.e. car, person), finally we superpose the detection maps onto the parsing map to obtain the final semantic label map (the **rightmost** image).

enhancement models. They are defined as $x_i^p = (c_i, p_i)$, where c_i represents the average color in the CIELab color space within the 3×3 neighborhood, and $p_i = (x_i, y_i)$ denotes the normalized sample position within the image.

4.2 Global Features

In photographic practice, global attributes and overall impressions, such as the average intensity of an image, at least have partial influence on artists when they decide how to enhance an image. We therefore incorporate global image features in our feature representation. Specifically, we adopt six types of global features proposed in [Bychkovsky et al. 2011], including *intensity distribution*, *scene brightness*, *equalization curves*, *detail-weighted equalization curves*, *highlight clipping*, and *spatial distribution*, which altogether give rise to a 207-dimensional vector.

4.3 Contextual Features

Our contextual features try to characterize the distribution of semantic categories, such as sky, building, car, person, and tree, in an image. Such features are extracted from semantic analysis results within a local region surrounding the sample pixel. Typical image semantic analysis algorithms include scene parsing [Tighe and Lazebnik 2010; Liu et al. 2011] and object detection [Viola and Jones 2001; Felzenszwalb et al. 2008; Wang et al. 2013]. Scene parsing tries to label every pixel in an image with its semantic category. Object detection on the other hand trains one highly specialized detector for every category of objects (such as dogs). Scene parsing is good at labeling categories (such as grass, roads, and sky) that have no characteristic shape but relatively consistent texture. These categories have a large scale, and typically form the background of an image. Object detectors are better at locating categories (such as persons and cars), which are better characterized by their overall shape than local appearance. These categories have a smaller scale, and typically occupy the foreground of an image. Because these two types of techniques are complementary to each other, we perform semantic analysis using a combination of scene parsing and object detection algorithms. Figure 5 illustrates one fusion example of the scene parsing and detection results.

We use existing algorithms to automatically annotate all input image pixels and the semantics information from the annotated images are gathered into a novel context feature descriptor. During pixel annotation, we perform scene parsing using the state-of-the-art algorithm in [Tighe and Lazebnik 2010]. The set of semantic categories, \mathcal{S}^p , during scene parsing include such object types as sky, road, river, field and grass. . After the scene parsing step, we obtain a parsing map, denoted as \mathcal{I}^p , each pixel of which receives one category label from \mathcal{S}^p , indicating that with a high probability, the corresponding pixel in the input image is covered by a semantic instance in that category. We further apply the state-of-the-art object

detector in [Wang et al. 2013] to detect the pixels covered by a predefined set of foreground object types, \mathcal{O}^d , which include person, train, bus and building. After the detection step, we obtain one confidence map for each predefined type. We fuse all confidence maps into one by choosing, at every pixel, the object label that has the highest confidence value. This fused detection map is denoted as \mathcal{I}^d . We further merge \mathcal{I}^d with \mathcal{I}^p so that those pixel labels from \mathcal{I}^d with confidence larger than a predefined threshold are used to overwrite the corresponding labels from \mathcal{I}^p . Since scene parsing and object detection results tend to be noisy, we rely on voting and automatic image segmentation to perform label cleanup in the merged label map. Within each image segment, we reset the label at every pixel to the one that appears most frequently in the segment. In our experiments, we adopt the image segmentation algorithm in [Arbelaez et al. 2011]. This cleaned map becomes our final semantic label map, \mathcal{I}^{label} .

Given the final semantic label map for the entire input image, we construct a contextual feature descriptor for each sample pixel to represent multiscale object distributions in its surroundings. For a sample point p_i , we first define a series of nested square regions, $\{R_0, R_1, \dots, R_\tau\}$, all centered at p_i . The edge length of these regions follows a geometric series, i.e. $\lambda_k = 3\lambda_{k-1}$ ($k = 1, \dots, \tau$), making our feature representation more sensitive to the semantic contents at nearby locations than those farther away. We further subdivide the ring between every two consecutive squares, $R_{k+1} - R_k$, into eight rectangles, as shown in Figure 4. Thus, we end up with a total of $9\tau + 1$ regions, including both the original regions in the series as well as regions generated by subdivision. For each of these regions, we compute a semantic label histogram, where the number of bins is equal to the total number of semantic categories, $N = |\mathcal{S}^p \cup \mathcal{O}^d|$. Note that the histogram for R_k is the sum of the histograms for the nine smaller regions within R_k . Such spatial pooling can make our feature representation more robust and better tolerate local geometric deformations. The final contextual feature descriptor at p_i is defined to be the concatenation of all these semantic label histograms. Our multiscale context descriptor is partially inspired by shape contexts [Belongie et al. 2002]. However, unlike the shape context descriptor, our regions and subregions are either rectangles or squares, which facilitate fast histogram computation based on integral images (originally called summed area tables) [Viola and Jones 2001]. In practice, we pre-compute N integral images, one for each semantic category. Then the value of each histogram bin can be calculated within constant time, which is extremely fast compared with the computation of shape contexts. To the best of our knowledge, our method is the first one that explicitly constructs semantically meaningful contextual descriptors for learning complex image enhancement models.

It is important to verify whether the complexity of our contextual features is necessary in learning complex spatially varying local adjustment effects. We have compared our results against those

obtained without contextual features as well as those obtained from simpler contextual features based on just one pooling region (vs. our 28 multiscale regions) at the same size as our largest region. From Figure 11, we can see that our contextual features are able to produce local adjustment results closest to the ground truth.

Discussion. The addition of this semantic component into our feature vectors is a major difference with previous work. As shown in Figure 11 and in the result section, the design that we propose for this component is effective and produces a significant improvement in practice. That said, we acknowledge that other options may be possible and we believe that exploring the design space of semantic descriptors is an exciting avenue for future work.

5. TRAINING DATA SAMPLING AND SELECTION

5.1 Superpixel Based Sampling

When training a mapping function using a set of images, we prefer not to make use of all the pixels as such a dense sampling would result in unbalanced training data. For example, we could have too many pixels from large “sky” regions while relatively few from smaller “person” regions, which could eventually result in a serious bias in the trained mapping function. In addition, an overly dense sampling unnecessarily increases the training cost, as we need to handle millions of pixel samples. Therefore, we apply a superpixel based method to collect training samples. For each training image I , we first apply the graph-based segmentation [Felzenszwalb and Huttenlocher 2004] to divide the image into small homogeneous yet irregularly shaped patches, each of which is called a superpixel. Note that a superpixel in a smooth region may be larger than one in a region with more high-frequency details. We require that the color transform returned by our mapping function at the centroid of a superpixel be used for predicting with sufficient accuracy the adjusted color of all pixels within the same superpixel. To avoid bias, we randomly sample a fixed number of pixels from every superpixel. Let ν be any superpixel from the original images (before adjustment) in Λ , and S_ν be the set of pixels sampled from ν . We revise the cost function in (1) as follows to reflect our superpixel-based sampling and local smoothness requirement.

$$\sum_{\nu} \sum_{j \in S_\nu} \| \Phi(\Theta, x_\nu) V(c_j) - y_j \|^2, \quad (3)$$

where Θ represents the set of trained weights in the neural network, x_ν is the feature vector constructed at the pixel closest to the centroid of ν , $V(c_j)$ denotes the color basis vector of a sample pixel within ν , and y_j denotes the adjusted color of the same sample within ν .

5.2 Cross-Entropy Based Image Selection

In example-based photo enhancement, example images that demonstrate a certain enhancement style often need to be manually prepared by human artists. It is a labor intensive task to adjust many images as each image has multiple attributes and regions that can be adjusted. Therefore, it is much desired to pre-select a small number of representative training images to reduce the amount of human work required. On the other hand, to make a learned model achieve a strong prediction capability, it is necessary for the selected training images to have a reasonable coverage of the feature space.

In this section, we introduce a cross-entropy based scheme for selecting a subset of representative training images from a large collection. We first learn a codebook of feature descriptors with $K = 400$ codewords by running K-means clustering on feature descriptors collected from all training images. Then every original

Algorithm 1: Small Training Set Selection

Input: A large image collection, Ω^I ; The desired number of representative images, m_d
Output: A subset Ω with m_d images selected from Ω^I

```

1 Initialize  $\Omega \leftarrow \emptyset$ 
2 for  $i = 1$  to  $m_d$  do
3    $I^* = \arg \max_{I \in \Omega^I - \Omega} - \sum_j H^{\Omega'}(j) \log H^{\Omega'}(j),$ 
4   where  $\Omega' = \Omega \cup \{I\};$ 
5    $\Omega = \Omega \cup \{I^*\}$ 
6 end

```

feature descriptor can find its closest codeword in the codebook via vector quantization, and each image can be viewed as “a bag of” codewords by quantizing all the feature descriptors in the image. We further build a histogram for every image using the codewords in the codebook as histogram bins. The value in a histogram bin is equal to the number of times the corresponding codeword appears in the image. Let H^k be the histogram for image I^k . For any subset of images Ω from an initial image collection Ω^I , we compute the accumulated histogram H^Ω by simply performing elementwise summation over the individual histograms of the images in Ω . We further evaluate the representative power of Ω using the cross entropy of H^Ω . That is, $\text{Entropy}(H^\Omega) = - \sum_j H^\Omega(j) \log H^\Omega(j)$, where $H^\Omega(j)$ denotes the j -th element of H^Ω . A large cross entropy implies that the codewords corresponding to the histogram bins are evenly distributed in the images in Ω and vice versa. Thus, to encourage an even coverage of the feature space, the set of selected images essentially need to be the solution of the following expensive combinatorial optimization,

$$\Omega = \arg \max_{\Omega \subseteq \Omega^I} - \sum_j H^\Omega(j) \log H^\Omega(j). \quad (4)$$

In practice, we seek an approximate solution by progressively adding one image to Ω every time until we have a desired number of images in the subset. Every time the added image maximizes the cross entropy of the expanded subset. This process is illustrated in Algorithm 1.

6. OVERVIEW OF EXPERIMENTS

Our proposed method is well suited for learning complex and highly nonlinear photo enhancement styles, especially when the style requires challenging spatially varying local enhancements. Successful local enhancement may not only rely on the content in a specific local region, but also contents in its surrounding areas. In that sense, such operations could easily result in complex effects that require stylistic or even exaggerated color transforms, making previous global methods (e.g., [Bychkovsky et al. 2011]) and local empirical methods (e.g., [Kaufman et al. 2012]) inapplicable. In contrast, our method was designed to address such challenges with the help of powerful contextual features and the strong regression capability of deep neural networks.

To fully evaluate our method, we hired one professional photographer who carefully retouched three different stylistic local effects using hundreds of photos. Section 7 reports experiments we have conducted to evaluate the performance of our method. Although our technique was designed to learn complex local effects, it can be readily applied to global image adjustments without any difficulty. Experiments in Section 8 and the supplemental materials show that our technique achieves superior performance both visually and nu-

merically when compared with other state-of-the-art methods on the MIT-Adobe Fivek dataset. To objectively evaluate the effectiveness of our method, we have further conducted two user studies (Section 8.3) and obtained very positive results.

6.1 Experimental Setup

Neural Network Setup. Throughout all the experiments in this paper, we use a fixed DNN with one input layer, two hidden layers, and one output layer (Figure 2). The number of neurons in the hidden layers were set empirically to 192, and the number of neurons in the output layer were set equal to the number of coefficients in the predicted color transform. Our experiments have confirmed that quadratic color transforms can more faithfully reproduce the colors in adjusted images than affine color transforms. Therefore, there are 30 neurons in the output layer, 10 for each of the three color channels.

Data Sampling. Since we learn pixel-level color mappings, every pixel within the image is a potential training sample. In practice, we segment each image into around 7,000 superpixels, from each of which we randomly select 10 pixels. Therefore, for example, even if we only have 70 example image pairs for learning one specific local effect, the number of training samples can be as large as 4.9 million. Such a large-scale training set can largely eliminate the risk of overfitting. It typically takes a few hours to finish training the neural network on a medium size training dataset with hundreds of images. Nevertheless, a trained neural network only needs 0.4 second to enhance a 512-pixel wide test image.

Image Enhancement with Learned Color Mappings. Once we have learned the parameters (weights) of the neural network, during the image enhancement stage, we apply the same feature extraction pipeline to an input image as in the training stage. That is, we first perform scene parsing and object detection, and then apply graph-based segmentation to obtain superpixels. Likewise, we also extract a feature vector at the centroid of every superpixel, and apply the color transform returned by the neural network to every pixel within the superpixel. Specifically, the adjusted color at pixel p_i is computed as $y_i = \Phi(\Theta, x_{\nu_i})V(c_i)$, where ν_i is the superpixel that covers p_i .

7. LEARNING LOCAL ADJUSTMENTS

7.1 Three Stylistic Local Effects

We manually downloaded 115 images from *Flickr* and resized them such that their larger dimension has 512 pixels. 70 images were chosen for training and the remaining 45 images for testing. A professional photographer used Photoshop to retouch these 115 images and produce the datasets for three different stylistic local effects. She could perform a wide range of operations to adjust the images, including selecting local objects/areas with the region selection tool, creating layers with layer masks, blending different layers using various modes, just to name a few. To reduce subjective variation during retouching, she used the “actions” tool, which records a sequence of operations, which can be repeatedly applied to selected image regions.

The first local effect “**Foreground Pop-Out**” was created by increasing both the contrast and color saturation of foreground salient objects/regions, while decreasing the color saturation of the background. Before performing these operations, foreground salient regions need to be interactively segmented out using region selection tools in Photoshop. Such segmented regions were only used for dataset production, and they are not used in our enhancement

pipeline. This local effect makes foreground objects more visually vivid while making the background less distractive. Figure 6 (b) and (c) show three examples of our automatically enhanced results and groundtruth results from the photographer. Refer to the supplemental materials for the training data as well as our enhanced testing photos.

Our second effect “**Local Xpro**” was created by generalizing the popular “cross processing” effect in a local manner. Within Photoshop, the photographer first predefined multiple “Profiles”, each of which is specifically tailored for one of the semantic categories used in scene parsing and object detection in section 4.3. All the profiles share a common series of operations, such as the adjustment of individual color channels, color blending across color channels, hue/saturation adjustment as well as brightness/contrast manipulation, just to name a few. Nonetheless, each profile defines a distinct set of adjustment parameters tailored for its corresponding category. When retouching a photo, the photographer used region selection tools to isolate image regions and then applied one suitable profile to each image region according to the specific semantic content within that region. To avoid artifacts along region boundaries, she could also slightly adjust the color/tone of local regions after the application of profiles. Although the profiles roughly follow the “cross processing” style, the choice of local profiles and additional minor image editing were heavily influenced by the photographer’s personal taste which can be naturally learned through exemplars. Figure 6 (d)&(e) show three examples in this effect, and compare our enhanced results against groundtruth results. Figure 1 shows another example of this effect.

To further increase diversity and complexity, we asked the photographer to create a third local effect “**Watercolor**”, which tries to mimic certain aspects of the “watercolor” painting style. For example, watercolors tend to be brighter with lower saturation. Within a single brush region, the color variation also tends to be limited. The photographer first applied similar operations as in the Foreground Pop-Out effect to the input images, including increasing both contrast and saturation of foreground regions as well as decreasing those of background regions. In addition, the brightness of both foreground and background regions are increased by different amounts. She further created two layers of brush effects from the same brightened image, using larger “brushes” on one layer and a smaller one on the other. On the first layer, the brush size for the foreground and the background are also different. Finally, these two layers are composited together using the ‘Lighten’ mode in Photoshop. Overall, this effect results in highly complex and spatially varying color transforms, which force the neural network to heavily rely on local contextual features during regression.

Figure 6 (f)&(g) show the enhanced results of three testing examples and their corresponding groundtruth results. To simulate brush strokes, after applying the same color transform to all pixels in a superpixel, we calculate the average color within the superpixel and fill the superpixel with it. See another example of Watercolor effect as well as visualized superpixels in Fig 7. Our automatic results look visually similar to the ones produced by the photographer. Refer to the supplemental materials for more examples enhanced with this effect. Note that our intention here is not rigorously simulating watercolors, but experimentally validating that our technique is able to accurately learn such complex local adjustments.

To successfully learn an enhancement effect, it is important to make the adjustments on individual training images consistent. In practice, we have found the following strategies are helpful in increasing such consistency across an image set. First, as artistic adjustment of an image involves the personal taste of the photographer, the result could be quite different from different photographers.



Fig. 6. Examples of three stylistic local enhancement effects. **Row (a)**: input images. **Row (b)&(c)**: our enhanced results and the groundtruth for the Foreground Pop-Out effect. **Row (d)&(e)**: our enhanced results and the groundtruth for the Local Xpro effect. **Row (f)&(g)**: our enhanced results and the groundtruth for the Watercolor effect.

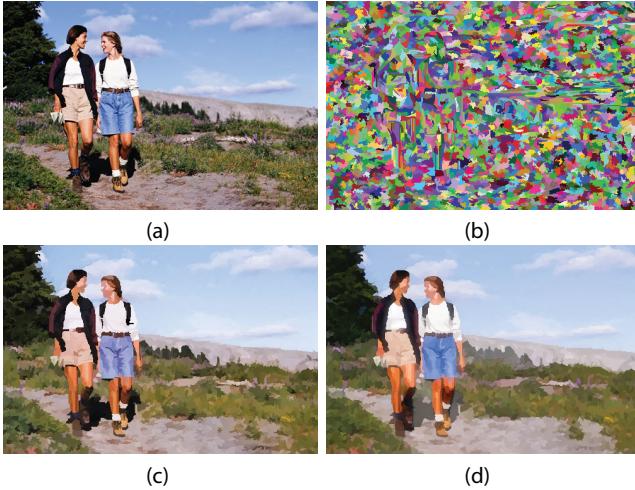


Fig. 7. An example of Watercolor local effect. (a): input image. (b): a visualization of superpixels used for simulating brush strokes. Each superpixel is filled with a random color. (c): our enhanced result. (d): the ground truth.

Therefore, we always define a retouching style using photos adjusted by the same photographer. That means, even for the same input content, retouched results by different photographers are always defined as different styles. Second, we inform the photographer the semantic object categories that our scene parsing and object detection algorithms are aware of. Consequently, she can apply similar adjustments to visual objects in the same semantic category. Third, we use the "actions" tool in Photoshop to faithfully record the "Profiles" that should be applied to different semantic categories. This improves the consistency of color transforms applied to image regions with similar content and context.

7.2 Spatially Varying Color Mappings

It is important to point out that the underlying color mappings in the local effect datasets are truly not global. They spatially vary within the image domain. To verify this, we collect pixels from each semantic region of an image. By drawing scatter plots for different semantic regions using pixel color pairs from the input and retouched images, we are able to visualize the spatially varying color transforms. See such an example in Figure 9, which clearly shows that the color transforms differ in the sky, building, grass and road regions. Also, we can see that our method can successfully learn such spatially varying complex color transforms. We further conducted a comparison against [Wang et al. 2011], which adopts a local piecewise approximation approach. However, due to the lack of discriminative contextual features, their learned adjustment parameters tend to be similar across different regions (Figure 8).

7.3 Generalization Capability

Here we verify the generalization capability of the DNN based photo adjustment models we trained using 70 image pairs. As mentioned earlier, the actual number of training samples far exceeds the number of training image pairs because we use thousands of superpixels within each training image pair. As shown in Fig. 10, we apply our trained models to novel testing images with significant visual differences from any images in the training set. The visual objects in these images have either unique appearances or unique spatial configurations. To illustrate this, we show the most similar training



Fig. 8. Comparison with [Wang et al. 2011] on the Local Xpro effect. **Top Left:** Input image; **Top Right:** enhanced image by [Wang et al. 2011]; **Bottom Left:** enhanced image by our approach; **Bottom Right:** enhanced image by photographer. The enhanced image by our approach is closer to the ground truth generated by the photographer.

images, which not only share the largest number of object and region categories with the testing image, but also have a content layout as similar as possible. In Fig 10 top, the mountain in the input image has an appearance and spatial layout that are different from the training images. In Fig 10 bottom, the appearances and spatial configuration of the car and people are also quite different from those of the training images. In despite of these differences, our trained DNN models are still able to adjust the input images in a plausible way.

7.4 Effectiveness of Contextual Features

We demonstrate the importance of contextual features in learning local adjustments in this subsection. First, we calculate the L^2 distance in the 3D CIELab color space between input images and ground truth produced by the photographer for all local effect datasets as shown in the second column of Table I. They numerically reflect the magnitude of adjustments the photographer made to the input images. Second, we numerically compare the testing errors of our enhanced results with and without the contextual feature in the third and fourth columns of Table I. Our experiments show that without contextual features, testing errors of our enhanced results tend to be relatively high. The mean L^2 error in the 3D CIELab color space reaches **9.27**, **9.51** and **9.61** respectively for the Foreground Pop-Out, Local Xpro and Watercolor effects. On the other hand, by including our proposed contextual feature, all errors drop significantly to **7.08**, **7.43** and **7.20**, indicating the necessity of such features.

Table I. Statistics of three local effects and the mean L^2 testing errors. TE=Testing Error.

Effect	ground truth L^2 distance	TE w/o context	TE w/ context
Foreground Pop-Out	13.86	9.27	7.08
Local Xpro	19.71	9.51	7.43
Watercolor	15.30	9.61	7.20

To validate the effectiveness of our multiscale spatial pooling schema in our contextual feature design, we have experimented with

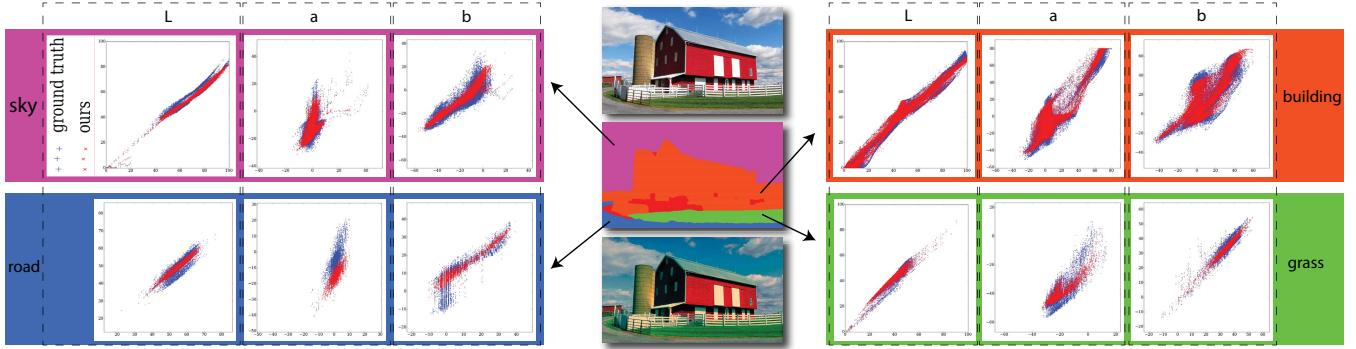


Fig. 9. Scatter plots of color mappings. **Middle (from top to bottom)**: input image, semantic label map and the groundtruth for the Local Xpro effect. **Left and right**: color mapping scatter plots for four semantic regions. Each semantic region has three scatter plots corresponding to its L , a , b color channels. Each scatter plot visualizes two sets of points, which take the original value of a channel as the horizontal coordinate, and respectively the predicted (red) and groundtruth (blue) adjusted values of that channel as the vertical coordinate.



Fig. 11. Effectiveness of our contextual features. **Left:** Input; **Middle Left:** enhanced without context; **Middle:** enhanced with simple contextual features (from a single pooling region); **Middle Right:** enhanced with our contextual features; **Right:** ground truth. It is obvious that among all enhanced results, the one enhanced using our contextual features is the closest to the ground truth.

a simpler yet more intuitive contextual feature descriptor with just one pooling region (vs. our 28 multiscale regions) at the same size as our largest region, and found that such simple contextual features are helpful in reducing the errors but not as effective as ours. Taking the local Watercolor painting effect as an example, we observed the corresponding mean L^2 error is **8.28**, which drops from **9.61**, but still obviously higher than our multiscale features **7.20**. This is because, with multiscale pooling regions, our features can achieve a certain degree of translation and rotation invariance, which is crucial for the histogram based representation. We have also performed visual comparisons. Fig. 11 shows one such example. We can see that without our contextual feature, local regions in the enhanced photo might exhibit severe color deviation from the ground truth.

7.5 Effectiveness of Learning Color Transforms

As shown in Figure 3, the use of color transforms helps absorb high-frequency color variations and enables DNN to regress the spatially smooth but otherwise highly nonlinear part of the color mapping. To highlight the benefits of using color transforms, we train a different DNN to regress the retouched colors directly. The DNN architecture is similar to the one described in section 6.1 except that there are only 3 neurons in the output layer, which represent the enhanced CIELab color. We compare the testing L^2 errors on the Foreground Pop-Out and Local Xpro datasets in Table II. On both datasets, the testing error increases by more than 20% which indicates the use of color transforms is beneficial in our task.

7.6 DNN Architecture

The complexity of our DNN based model is primarily determined by the number of hidden layers and the number of neurons in each layer. Note that the complexity of the DNN architecture should meet the inherent complexity of the learning task. If the DNN did not have the sufficient complexity to handle the given task, the trained model would not even be able to accurately learn all the samples in the training set. On the other hand, if the complexity of the DNN exceeds the inherent complexity of the given task, there exists the risk of overfitting and the trained model would not be able to generalize well on novel testing data even though it could make the training error very small.

The nature of the learning task in this paper is a regression problem. It has been shown that a feedforward neural network with a single hidden layer [Hornik et al. 1989] can be used as a universal regressor and the necessary number of neurons in the hidden layer varies with the inherent complexity of the given regression problem. In practice, however, it is easier to achieve a small training error with a deeper network that has a relatively small number of neurons in the hidden layers. To assess the impact of the design choices of

Table II. Comparison of L^2 testing errors obtained from deep neural networks using and without using quadratic color transforms.

Effect	w/o transform	w/ transform
Foreground Pop-Out	8.90	7.08
Local Xpro	9.01	7.43



Fig. 10. Two examples of novel image enhancement. **Top:** an example of the Watercolor effect. **Bottom:** an example of the Local Xpro effect. In each example, (a): input image, (b): our enhanced result, (c): ground truth, (d): training images most similar to the input image. Note that the input images in these examples have significant visual differences from any images in the training set.

the DNN architecture, we evaluate DNNs with a varying number of hidden layers and neurons. We keep a held-out set of 30 images for validation and vary the number of training images from 40 to 85 at a step size of 15 to evaluate the impact of the size of the training set. We repeat the experiments for five times with random training and testing partitions and report the averaged results. The Foreground Pop-Out dataset is used in this study. Fig 12 summarizes our experimental results. Overall, neural networks with a single hidden layer deliver inferior performance than deeper networks. DNNs with 3 hidden layers do not perform as well as those with 2 hidden layers. For a DNN with 2 hidden layers, when the number of training images exceeds 70, the testing error does not significantly improve any more. In summary, DNNs with 2 hidden layers achieve low testing errors and execute faster than those with 3 hidden layers in both training and testing stages. Therefore, we finally use a DNN with 2 hidden layers and 192 neurons each throughout this paper.

7.7 Comparison with Other Regression Methods

Our DNN proves to be effective for regressing spatially varying complex color transforms on the three local effect datasets. It is also of great interest to evaluate the performance of other regressors on our datasets. Specifically, we chose to compare DNN against two popular regression methods, Lasso [Tibshirani 1996] and random forest [Breiman 2001]. Both Lasso and random forest are scalable

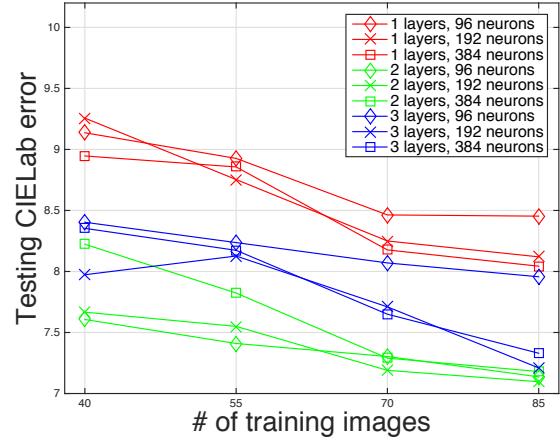


Fig. 12. Testing error vs number of training images for DNNs of various architectures. Error bars are omitted for clarity.

Table III. Comparison of L^2 testing errors obtained from different regressors.

Effect	Lasso	Random Forest	DNN
Foreground Pop-Out	11.44	9.05	8.90
Local Xpro	12.01	7.51	9.01
Watercolor	9.34	11.41	9.22

to the large number of training samples used in DNN training. We use Lasso and a random forest to directly regress target CIELab colors using the same feature set as in DNN training, including pixelwise features, global features and contextual features. The hyperparameters of both Lasso and the random forest are tuned using cross validation. To make a fair comparison, our DNN is also adapted to directly regress the target CIELab colors. A comparison of L^2 errors is summarized in Table III. The DNN significantly outperforms Lasso on the Foreground Pop-Out and Local Xpro datasets, and obtains slightly lower errors on the Watercolor dataset. Compared with the random forest, the DNN obtains lower testing errors on both Foreground Pop-Out and Watercolor datasets. On the Local Xpro dataset, the random forest obtains lower numerical errors than that of the DNN. However, after visual inspection, we found that colors generated by the random forest are not spatially smooth and blocky artifacts are prevalent in the enhanced images, as shown in Figure 13. This is because regression results from a random forest are based on values retrieved from various leaf nodes, and spatial smoothness of these retrieved values cannot be guaranteed. In contrast, our trained DNN generates spatially smooth colors and does not give rise to such visual artifacts.

8. LEARNING GLOBAL ADJUSTMENTS

8.1 MIT-Adobe FiveK Dataset

The MIT-Adobe FiveK dataset [Bychkovsky et al. 2011] contains 5000 raw images, each of which was retouched by five well trained photographers, which results in five groups of global adjustment styles. As we learn pixel-level color mappings, there would be 175 million of training samples in total if half of the images are used for training.



Fig. 13. Visual comparison against Lasso and a random forest. Note an area with blocky artifacts in the result of the random forest is highlighted.

We have compared our method with [Hwang et al. 2012] using the same experimental settings and testing datasets in that work. Two testing datasets were used in [Hwang et al. 2012]. (1) “Random 250”: 250 randomly selected testing images from group C of the MIT-Adobe FiveK dataset (hence 4750 training images) and (2) “High Variance 50”: 50 images selected for testing from group C of the MIT-Adobe FiveK dataset (hence 4950 images for training). Comparison results on numerical errors are shown in the second and third columns of Table IV, from which we can see our method is capable of achieving much better prediction performance in terms of mean L^2 errors on both predefined datasets. Figure 14 further shows the error histograms of our method and [Hwang et al. 2012] on these two testing datasets. The errors produced by our method are mostly concentrated at the lower end of the histograms. Figure 15 shows a visual comparison, from which we can see our enhanced result is closer to the ground truth produced by the photographer. Such performance differences could be explained as follows. The technique in [Hwang et al. 2012] is based on nearest-neighbor search, which requires a fairly large training set that is slow to search. As a result, this technique divides similarity based search into two levels. It first searches for the most similar images and then the most similar pixels within them. While this two-level strategy accelerates the search, a large percentage of similar pixels does not even have the chance to be utilized because the search at the image level leaves out dissimilar images that may still contain many similar pixels. On the other hand, our deep neural network based method is a powerful nonlinear regression technique that considers all the training data simultaneously. Thus our method has a stronger extrapolation capability than the nearest-neighbor based approach in [Hwang et al. 2012], which only exploits a limited number of nearest neighbors. For the same reason, the nearest-neighbor based approach in [Hwang et al. 2012] is also more sensitive to noisy and inconsistent adjustments in the training data. In another comparison with [Bychkovsky et al. 2011], we follow the same setting used in that work, which experimented on 2500 training images from group C and reported the mean error on the L channel (CIELAB color space) only. As shown in the first column of Table IV, we obtained a slightly smaller mean error on the L channel on the remaining 2500 testing images.

To validate the effectiveness of our cross-entropy based training set selection method (Algorithm 1), we have monitored the testing errors by varying the number of training images selected by our

Table IV. Comparison of mean L^2 errors obtained with our method and previous methods on the MIT-Adobe FiveK dataset. The target style is *Expert C*.

Method	2500(L)	Ran. 250(L,a,b)	H.50(L,a,b)
[Bychkovsky et al. 2011]	5.82	N/A	N/A
[Hwang et al. 2012]	N/A	15.01	12.03
Our method	5.68	9.85	8.36

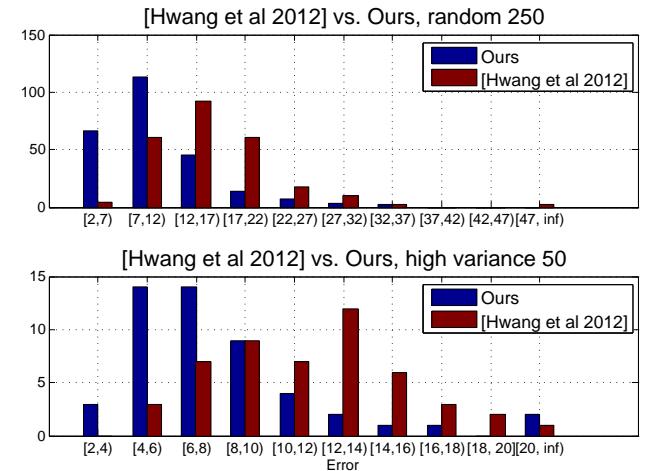


Fig. 14. L^2 error distributions. Note that our method produces smaller errors on both testing datasets.

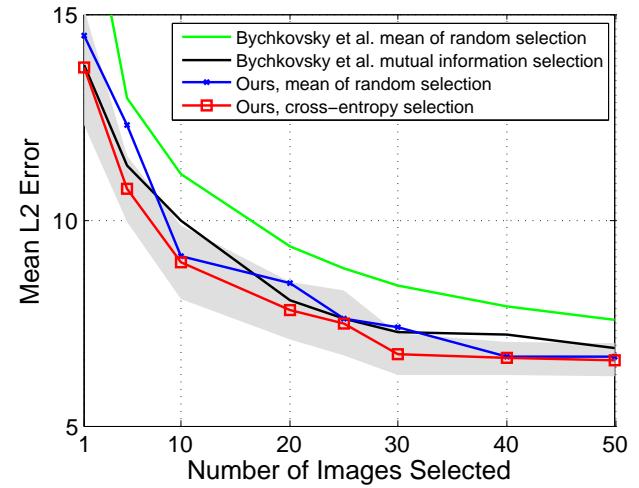


Fig. 16. Comparison of training image selection schemes. When compared with sensor placement based on mutual information, our cross-entropy based method achieves better performance especially when the number of selected images is small. The band shaded in light blue shows the standard deviations of the L^2 errors of our scheme.

method, and compared them with both naive random selection and the sensor placement method used in [Bychkovsky et al. 2011] (Figure 16). Interestingly, when the random selection scheme is used, our neural network based solution achieves significantly better



Fig. 15. Visual comparison with [Hwang et al. 2012]. **Left:** Input image; **Middle Left:** groundtruth enhanced image by expert C; **Middle Right:** enhanced image by our approach; **Right:** enhanced image by [Hwang et al. 2012].



Fig. 17. Comparison with Instagram. **Left:** Input images (from MIT-Adobe FiveK); **Middle:** our results; **Right:** results by Instagram. The top row shows the “EarlyBird” effect, and the bottom row shows the “Nashville” effect. This comparison indicates enhancement results by our trained color mappings are close to the ground truth generated by Instagram.

accuracy than the Gaussian Process based method. This is primarily due to the strong nonlinear regression power exhibited by deep neural networks and the rich contextual feature representation built from semantic analysis. When compared with sensor placement, our cross-entropy based method also achieves better performance especially when the number of selected images is small, which further indicates our method is superior for learning enhancement styles from a small number of training images.

8.2 Instagram Dataset

Instagram has become one of the most popular Apps on mobile phones. In Instagram, hundreds of filters can be applied to achieve different artistic color and tone effects. For example, the frequently used “Lo-Fi” filter boosts contrast and brings out warm tones; the “Rise” filter adds a golden glow while “Hudson” casts a cool light. For each specific effect, we randomly chose 50 images from MIT-Adobe FiveK, and let Instagram enhance each of them. Among the resulting 50 pairs of images, half of them were used for training, and the other half were for testing. We have verified whether images adjusted by the trained color mapping functions are similar to the ground truth produced by Instagram, which has the flavor of a reverse engineering task. Our experiments indicate that Instagram effects are relatively easy to learn using our method. Figure 17 shows the learning results for two popular effects.

8.3 User Studies

To perform a visual comparison between our results and those produced by [Hwang et al. 2012] in an objective way, we collected all the images from the two datasets, “Random 250” and “High vari-

ance 50”, and randomly chose 50, including 10 indoor images and 40 outdoor images, to be used in our user study. For each of these 50 testing images, we also collected the groundtruth images and the enhanced images produced with our method and [Hwang et al. 2012]. Then we invited 33 participants, including 12 females and 21 males, with ages ranging from 21 to 28. These participants had little experience of using any professional photo adjustment tools but did have experience with photo enhancement Apps such as “Instagram”. The experiment was carried out by asking each participant to open a static website using a prepared computer and a 24-inch monitor with a 1920x1080 resolution. For each test image, we first show the input and the groundtruth image pair to let the participants know how the input image was enhanced by the photographer (retoucher C). Then we show two enhanced images automatically generated with our method and Hwang *et al.* in a random left/right layout without disclosing which one was enhanced by our method. The participant was asked to compare them with the ground truth and vote on one of the following three choices: (a) “The left image was enhanced better”, (b) “The right image was enhanced better”, and (c) “Hard to choose”. In this way, we collected $33 \times 50 = 1650$ votes distributed among the three choices. Figure 18 shows a comparison of the voting results, from which we can see that enhanced images produced by our method received most of the votes in both indoor and outdoor categories. This comparison indicates that, from a visual perspective, our method can produce much better enhanced images than [Hwang et al. 2012].

Our second user study tries to verify whether our method has the capability to enhance a target effect in a statistically significant manner. To conduct this study, we chose 30 test images from one of the local effect datasets described in Section 7.1 as our test data. We asked 20 participants from the first study to join our second study. The interface was designed as follows. On top of the screen, we show as the ground truth the enhanced image produced by the photographer we hired, below which we show a pair of images with the left being the original image and the right being the enhanced image produced by our method. Then we asked the participant to assign a score to both the input and enhanced images by considering two criteria at the same time: (1) how closely this image conforms to the impression given by the ground truth, (2) the visual quality of the image. In other words, if the enhanced image looks visually pleasing and closer to the ground truth, it should receive a higher score. For the convenience of the participants, we simply discretized the range of scores into 10 levels. If an image looks extremely close to the ground truth, it should be scored 10. At the end, we collected two sets of scores for the original and enhanced images, respectively. We then conducted the paired T-test on the two sets of scores and found that the two-tail p-value is $p \approx 10^{-10}$, and $t = 1.96$, indicating that

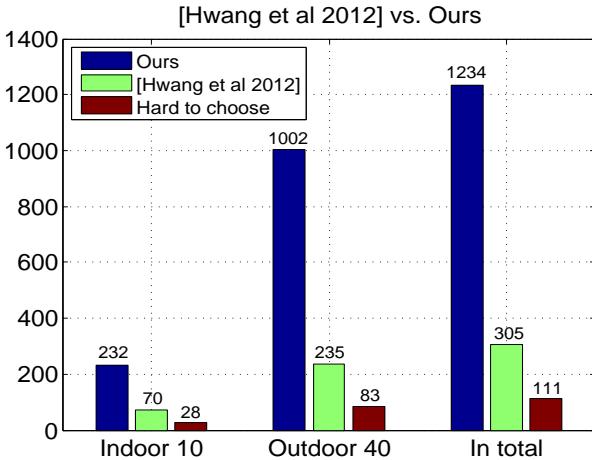


Fig. 18. A comparison of user voting results between our approach and [Hwang et al. 2012].

our approach has significantly enhanced the desired effect from a statistical point of view.

9. CONCLUSIONS AND DISCUSSIONS

In this paper, we have demonstrated the effectiveness of deep learning in automatic photo adjustment. We cast this problem as learning a highly nonlinear mapping function by taking the bundled features as the input layer of a deep neural network. The bundled features include a pixelwise descriptor, a global descriptor, as well as a novel contextual descriptor which is built on top of scene parsing and object detection. We have conducted extensive experiments on a number of effects including both conventional and artistic ones. Our experiments show that the proposed approach is able to effectively learn computational models for automatic spatially-varying photo adjustment.

Limitations. Our approach relies on both scene parsing and object detection to build contextual features. However, in general, these are still challenging problems in computer vision and pattern recognition. Mislabeling in the semantic map can propagate into contextual features and adversely affect photo adjustment. Fig 19(a) shows one such example for the Foreground Pop-Out effect. The ‘sea’ on the right side is mistakenly labeled as ‘mountain’ and its saturation and contrast are incorrectly increased. As both scene parsing and object detection are rapidly developing areas, more accurate techniques are emerging and could be adopted by our system to produce more reliable semantic label maps.

Another failure case is shown in Fig 19(b), where the adjustments in group C of the MIT-Adobe FiveK dataset are learnt. Our method produces insufficient brightness adjustment, which leads to dimmer result than the ground truth. In fact, the L^2 distance between the input image and the ground truth is 38.63, which is significantly higher than the mean distance 17.40 of the dataset. As our DNN is trained using all available training samples, individual adjustments significantly deviating from the average adjustment for a semantic object type are likely to be treated as outliers and cannot be correctly learnt.

Our system employs a deep fully connected neural network to regress spatially varying color transforms. There exist many design choices in the DNN architecture, including the number of hidden layers, the number of neurons in each layer, and the type of neural

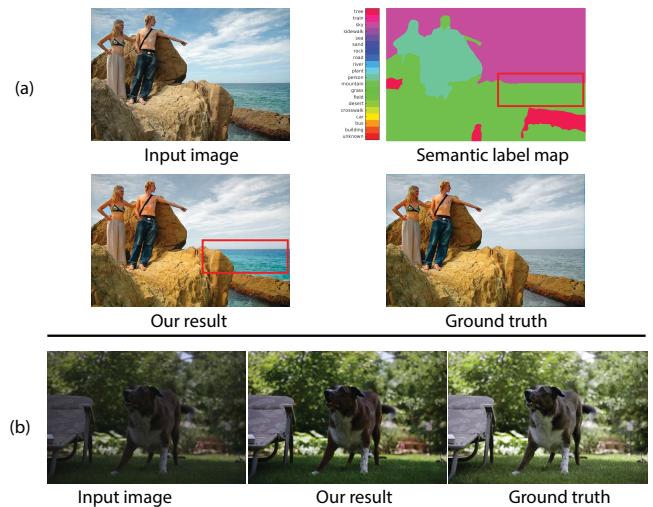


Fig. 19. Two failure cases. **Top row:** a failure case on Foreground Pop-Out effect. In the semantic label map, an area with incorrect semantic labeling is highlighted. Correspondingly, this area receives incorrect adjustments in our result. **Bottom row:** another failure case in “High Variance 50” test set of MIT-Adobe FiveK dataset.

activation functions. They together give rise to a time-consuming trial-and-error process in search of a suitable DNN architecture for the given task. In addition, DNN behaves as a black box and it is not completely clear how the network combines features at different scales and predicts the final color transforms. In fact, interpreting the internal representations of deep neural networks is still an ongoing research topic [Zeiler and Fergus 2013; Szegedy et al. 2013].

ACKNOWLEDGMENTS

We are grateful to Vladimir Bychkovsky and Sung Ju Hwang for fruitful discussions and suggestions. This work was partially supported by Hong Kong Research Grants Council under General Research Funds (HKU17209714).

REFERENCES

- AN, X. AND PELLACINI, F. 2008. Appprop: all-pairs appearance-space edit propagation. *ACM Trans. Graph.* 27, 3.
- ARBELAEZ, P., MAIRE, M., FOWLkes, C., AND MALIK, J. 2011. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 5, 898–916.
- BAE, S., PARIS, S., AND DURAND, F. 2006. Two-scale tone management for photographic look. *ACM Trans. Graph.* 25, 3, 637–645.
- BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 4 (Apr.), 509–522.
- BREIMAN, L. 2001. Random forests. *Machine learning* 45, 1, 5–32.
- BYCHKOVSKY, V., PARIS, S., CHAN, E., AND DURAND, F. 2011. Learning photographic global tonal adjustment with a database of input/output image pairs. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR ’11. 97–104.
- CAICEDO, J., KAPOOR, A., AND KANG, S. B. 2011. Collaborative personalization of image enhancement. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 249–256.

- COHEN-OR, D., SORKINE, O., GAL, R., LEYVAND, T., AND XU, Y.-Q. 2006. Color harmonization. *ACM Trans. Graph.* 25, 3 (jul), 624–630.
- DALE, K., JOHNSON, M., SUNKAVALLI, K., MATSIK, W., AND PFISTER, H. 2009. Image restoration using online photo collections. In *Computer Vision, 2009 IEEE 12th International Conference on*. 2217–2224.
- FELZENZWALB, P., MCALLESTER, D., AND RAMANAN, D. 2008. A discriminatively trained, multiscale, deformable part model. In *IEEE Conf. on Computer Vision and Pattern Recognition*.
- FELZENZWALB, P. F. AND HUTTENLOCHER, D. P. 2004. Efficient graph-based image segmentation. *Int. J. Comput. Vision* 59, 2 (Sept.), 167–181.
- HINTON, G. E., SRIVASTAVA, N., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR abs/1207.0580*.
- HORNIK, K., STINCHCOMBE, M., AND WHITE, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Netw.* 2, 5 (July), 359–366.
- HWANG, S. J., KAPOOR, A., AND KANG, S. B. 2012. Context-based automatic local image enhancement. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part I*. ECCV'12. 569–582.
- JOSHI, N., MATSIK, W., ADELSON, E. H., AND KRIEGMAN, D. J. 2010. Personal photo enhancement using example images. *ACM Trans. Graph.* 29, 2 (Apr.), 12:1–12:15.
- KANG, S. B., KAPOOR, A., AND LISCHINSKI, D. 2010. Personalization of image enhancement. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 1799–1806.
- KAUFMAN, L., LISCHINSKI, D., AND WERMAN, M. 2012. Content-aware automatic photo enhancement. *Comp. Graph. Forum* 31, 8 (Dec.), 2528–2540.
- KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* 25, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. 1106–1114.
- LISCHINSKI, D., FARBMAN, Z., UYTTENDAELE, M., AND SZELISKI, R. 2006. Interactive local adjustment of tonal values. *ACM Trans. Graph.* 25, 3, 646–653.
- LIU, C., YUEN, J., AND TORRALBA, A. 2011. Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 12, 2368?382.
- SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I., AND FERGUS, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.
- TIBSHIRANI, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.
- TIGHE, J. AND LAZEBNIK, S. 2010. Superparsing: Scalable nonparametric image parsing with superpixels. In *Proceedings of the 11th European Conference on Computer Vision: Part V*. ECCV'10. 352–365.
- VINCENT, P., LAROCHELLE, H., BENGIO, Y., AND MANZAGOL, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*. 1096–1103.
- VIOLA, P. AND JONES, M. 2001. Rapid object detection using a boosted cascade of simple features. In *IEEE Conf. on Computer Vision and Pattern Recognition*.
- WANG, B., YU, Y., AND XU, Y.-Q. 2011. Example-based image color and tone style enhancement. In *ACM SIGGRAPH 2011 Papers*. SIGGRAPH '11. 64:1–64:12.
- WANG, X., YANG, M., ZHU, S., AND LIN, Y. 2013. Regionlets for generic object detection. In *ICCV'13: Proc. IEEE 14th International Conf. on Computer Vision*.
- ZEILER, M. D. AND FERGUS, R. 2013. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*.