Convolution Neural Networks With Two Pathways for Image Style Recognition

Tiancheng Sun, Yulong Wang, Jian Yang, and Xiaolin Hu, *Senior Member, IEEE*

***Abstract*— Automatic recognition of an image's style is important for many applications, including artwork analysis, photo organization, and image retrieval. Traditional convolution neural network (CNN) approach uses only object features for image style recognition. This approach may not be optimal, because the same object in two images may have different styles. We propose a CNN architecture with two pathways extracting object features and texture features, respectively. The object pathway represents the standard CNN architecture and the texture pathway intermixes the object pathway by outputting the gram matrices of intermediate features in the object pathway. The two pathways are jointly trained. In experiments, two deep CNNs, AlexNet and VGG-19, pretrained on the ImageNet classification data set are fine-tuned for this task. For any model, the two-pathway architecture performs much better than individual pathways, which indicates that the two pathways contain complementary information of an image's style. In particular, the model based on VGG-19 achieves the state-of-the-art results on three benchmark data sets, WikiPaintings, Flickr Style, and AVA Style.**

***Index Terms*— Image style recognition, neural network.**

I. Introduction

THE style in a painting or photograph plays a significant role in people's perception of the image. Take Figure 1 as an example: the lower picture conveys a nostalgia feeling compared to the upper one, notwithstanding the same scene depicted in the pictures. Human beings can sense this subtle difference easily, but it is still a difficult task for computers. In this study, our focus is to design an efficient algorithm to recognize the style of an image.

Fig. 1. Two images with similar content but different styles.

Paintings and artistic photographs are two important types of images that have distinctive styles. The styles of paintings are usually named according to art movements, which is marked by a trend in the art history. For instance, the famous *Impressionism* is an art movement in 1880s, which is characterized by relatively small, thin, yet visible brush strokes, and accurate depiction of light. This kind of styles are well defined yet not easy for non-artists to distinguish. In contrast, the styles of photographs are easier for ordinary people to distinguish as the names of the styles often come from the feeling the photos convey (e.g., Romantic), or just the technique used when taking the photo (e.g., HDR).

The research on automatic image style recognition has not gained much progress until recent years [1]. One reason is that the style information of an image may be encoded in any component of the image including local and global, shape and color features. Another reason is that the definition of a style is often subjective, especially for photography works, which may vary from person to person. It is hard to be

Manuscript received October 10, 2016; revised April 23, 2017; accepted May 21, 2017. Date of publication June 9, 2017; date of current version June 23, 2017. This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2013CB329403 and Grant 2014CB349303, in part by the National Natural Science Foundation of China under Grant 91420201, Grant 61332007, Grant 61621136008, Grant 61620106010, and Grant 61472187, in part by the Program for Changjiang Scholars, and in part by a Grant from Sensetime. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Peter Tay. *(Corresponding author: Xiaolin Hu.)*

T. Sun is with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China.

Y. Wang is with the Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.

J. Yang is with the Department of Computer Science, Nanjing University of Science and Technology, Nanjing 210094, China.

X. Hu is with the Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, and the Center for Brain-Inspired Computing Research, Tsinghua University, Beijing 100084, China.

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.2017.2710631

accurately related to a set of image features. In recent years, along with great success of deep convolution neural network (CNN) on image classification [2], [3], the accuracy of image style recognition has been boosted significantly by using CNN-based approaches [4], [5]. Most of these works rely on the ability of CNN for feature representation. In addition, they usually adopt high-level features. The underlying assumption is that CNN's features, which are good at distinguishing high-level semantic contents (e.g., objects) are also good at distinguishing image styles. This turns out to be true. That is why these CNN-based approaches have yielded better results than previous approaches.

However, it is possible that these features are still insufficient to characterize image styles. In fact, a recent study [6] reveals that texture information is also important for human perception of image style, which is not emphasised in recent deep learning models [4], [5] for style recognition. By separating texture and content information of an image encoded in intermediate layers of a CNN, Gatys et al. proposed a method that can change the texture of an image without modifying its content, which leads to the change of the style of the image [6]. In this way, an ordinary photograph can be converted to, for example, an image with the style of *The Starry Night* by the famous painter Vincent van Gogh in 1889.

The results in [6] motivate us to evaluate the effectiveness of texture information for image style recognition. We aim to propose a CNN-based model that integrates both content (or object) information and texture information for image style recognition. Our primary concern is whether texture information will help to boost the accuracy of deep learning models.

Object information can be obtained in the same way as in standard feedforward deep learning models for general image classification. It is defined as responses of features in one or more layers of a deep learning model, assuming the features (or filters) are local or global object detectors, which makes sense by visualizing the features [7]. But the definition of texture information is abstract. According to Sarfraz [8], texture "gives information about the spatial arrangement of color or intensities in an image or selected region of an image". Gatys *et al.* [6], [9] suggested a powerful and practical approach for extracting this information. The basic idea is to characterize it using the correlation between responses of all (or a subset of all) features in certain layers of a CNN. Since this metric is independent of feature location, it measures the distribution of features across the entire image, which is a hallmark of texture. By considering its simplicity and effectiveness in style transformation, we adopt this approach to extract texture in our study. But unlike in Gatys *et al.*'s works [6], [9] where the texture features were fixed because the network models were fixed (the variables there were input pixel values), in our case, a set of optimal texture features need to be learned by adapting the network weights for style recognition.

The main contribution of this work is the proposal of a deep learning network consisting of an object pathway and a texture pathway for image style recognition. It outperforms the state-of-the-art models on three benchmark datasets. The rest of the paper is organized as follows. A review of previous works related to this topic is presented in Section II. The architecture of the model and training method are presented in Section III. Experimental results on three benchmark datasets are presented in Section IV. Finally, Section V concludes the paper with a discussion of possible directions to improve the proposed approach.

II. Related Work

A. Convolutional Neural Network

The convolutional neural network (CNN) was proposed for image classification first in 1989 [10]. But it had not gained much attention until 2012 when a large CNN outperformed other methods on a large image classification benchmark dataset ImageNet [11]. Along with the vast advancement of parallel computing techniques, CNN has exhibited excellent performance in many computer vision tasks ranging from low-level image processing such as super-resolution [12] to high-level image understanding such as image classification [2], [3], [11], [13], object detection [14]–[16] and scene labeling [17], [18]. The success of CNN is mainly attributed to its ability for different levels of representation of complex natural images. Basically, it has two types of operations in different layers, convolution and pooling. Convolutional layers perform template matching and pooling layers perform subsampling, and both perform feature integration but with different methods. Nonlinear transformations (i.e., neural activation) are often combined with them. Note that subsampling can be also realized by convolutions with larger stride than one. With this trick, a recent work showed that pooling could be omitted in CNN [19]. Besides the two types of layers, in recent years, many other types of layers have been proposed to facilitate training and improve generalization ability including local response normalization layer [11], batch normalization layer [20] and dropout layer [21].

B. Image Style Recognition

Style recognition has been studied for many years. An early work [22] proposed to divide a painting into small blocks and extract DCT coefficients in the blocks as local features. Then the naive Bayes is applied to identify its painter. Another work [23] proposed to identify not only the painter of a painting but also the school of art it represents (impressionism, surrealism or abstract expressionism) based on a large set of local features including Radon transform features [24], Gabor filters, Haralick features [25]. A recent work [26] proposed a multi-task dictionary learning strategy which can dissociate artist-specific and style-specific patterns in paintings. However, the effectiveness of these works were only demonstrated on a small number of artworks. The largest dataset among these had only 1616 images [26].

In recent years, several large datasets were constructed for image style analysis. The AVA Style dataset was introduced by Murray *et al.* [27] which has about 14000 images with photographic styles. The WikiPainting dataset and Flickr Style

dataset were introduced by Karayev *et al.* [1], which contain 85,000 images and 80,000 images, respectively. They were designed for painting styles and photography styles classification, respectively. Traditional image features such as SIFT and LBP were used for style recognition on the AVA Style dataset [27]. It was shown that high level CNN features outperformed traditional features on this dataset [1]. Fusion of these features achieved better results than single features on all of the three datasets [1]. An independent study by Bar *et al.* [4] arrived at the same conclusion about the effect of feature fusion on the WikiPainting dataset. But it is possible that CNN features alone can achieve better results [5]. In view that style recognition requires both holistic information and fine-grained details, Lu *et al.* [5] proposed to aggregate the outputs of CNNs on multiple patches of an image. This approach keeps the record of result on the AVA Style dataset.

C. Texture Synthesis

As mentioned in Section I, the style of an image is closely related to its texture. Texture synthesis is a hot topic in computer graphics. One class of approaches make use of local regions of the original image by copying, transformation and stitching [28], [29]. Another class of approaches take statistical measurements on filter responses on the original image and try to make the synthesized image match these measurements [30]–[32]. It is reasonable to infer that the performance of the latter class of approaches relies on the filters. In [30]–[32] only simple filters such as Gabor filters are used. In the era of deep learning, a natural extension of those works is to substitute the simple features with sophisticated CNN features. This is the basic idea of Gatyes et al.'s works [6], [9]. They found that the texture of images can be effectively represented by the gram matrix of intermediate layers of a CNN pretrained on a large image classification dataset ImageNet. By optimizing the values of input pixels, a new image can be created with the same texture as the original image. If the content of the original image, which is represented by filter responses at different layers of the CNN, is intentionally preserved, the method performs texture synthesis [9]. If the content of the original image (say, image A) is changed to that of another image (say, image B), then a new image with image B's content and image A's texture (or style) is created [6]. Later works further optimized the running time [33], [34] and flexibility [38] of the framework: producing images with multiple styles in real time now becomes possible.

III. The Model

The proposed model consists of two pathways for style recognition. The "object pathway" is the standard pathway of a CNN, which is from the bottom layer to the output layer. The "texture pathway" has multiple paths to the output layer, and each path carries the correlations between filter responses in a particular layer of the texture pathway. Therefore two pathways are not independent. Their predictions about the style of the image are averaged as the final output. The overall structure of the model is illustrated in Figure 2.

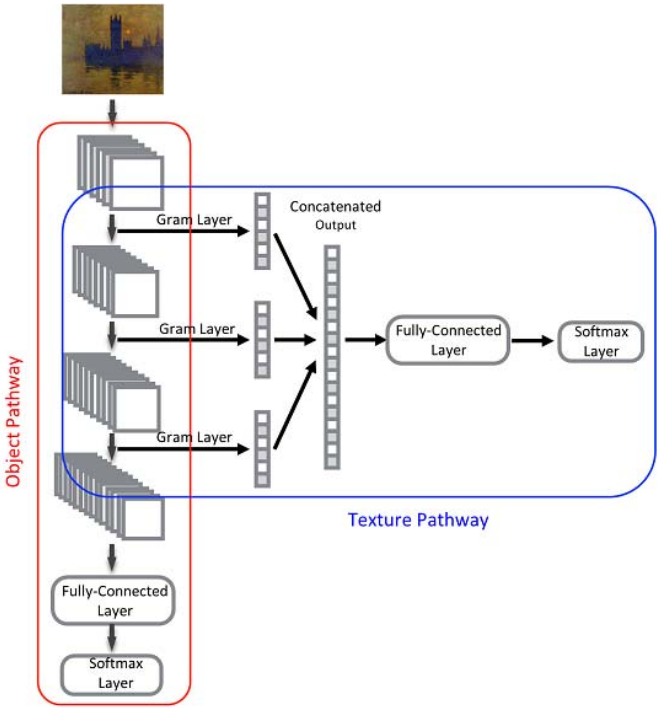


Fig. 2. The structure of the proposed model. The two pathways make their predictions based on softmax functions separately, which are averaged to give the final prediction.

A. Object Pathway

The object information is computed using CNN: the image is sequentially passed through different layers of CNN. A typical CNN has many layers including convolutional layers, pooling layers, local normalization layer, fully connected layers and so on. The proposed method in the present paper is not limited to specific a CNN architecture, and any existing CNN can be used. The output layer of the object pathway is a classification layer which gives an object-base prediction of the image style. The softmax function is used as the output function and the cross-entropy error is used as the loss function.

B. Texture Pathway

The texture pathway consists of multiple *gram layers*, each of which computes correlations between filter responses in a particular layer in the object pathway. In practice, we calculate the inner products of filter responses in a layer and obtain a gram matrix. The gram matrices calculated in lower layers capture finer texture information and the gram matrices in higher layers capture coarser texture information. This representation of texture follows Gatys et al.'s idea for image synthesis [6], [9].

Specifically, to get the texture information of an image, we first pass the image through CNN and compute the response of each intermediate convolutional layer l. Assume that layer l contains N_l filters and therefore N_l feature maps, and each map is vectorized to a vector with length M_l (Figure 3). Then, these feature maps can be represented as a two-dimension matrix $\mathbf{F}^l$ in the space $\mathcal{R}^{N_l \times M_l}$. The correlations between each pair of feature maps is an element

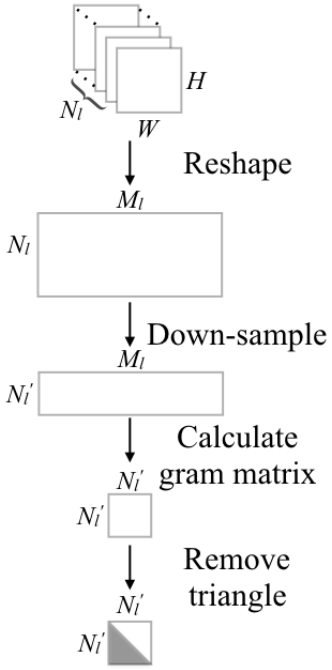


Fig. 3. The computing process of the gram layer. The feature maps of a layer in the object pathway are first converted to a two-dimensional matrix, then go through multiple stages of processing.

in the gram matrix

$$\mathbf{G}^l = \mathbf{F}^l \cdot \mathbf{F}^{l\top} \tag{1}$$

where $\top$ denotes transpose.

To improve generalization ability of the proposed model, one usually uses existing CNNs pretrained on the ImageNet dataset. Such CNNs usually have tens of filters or even more in a layer. Then there would be thousands of elements in the gram matrix, which are too many to forward to the next layer. In order to shrink the size of the gram matrix, before the gram matrix computation, we first down-sample the feature maps from $N_l \times M_l$ to $N_l' \times M_l$, which means we choose one fixed feature map out of every N_l/N_l' feature maps. Note that we do not use max pooling and average pooling here because nearby elements in the gram matrix are not correlated as in the case of image or feature map. In addition, since the gram matrix is symmetric, only the lower triangle part of the matrix is fed into the final classification layer. Then, each gram matrix contributes $N_l'(N_l'+1)/2$ inputs to the next layer. In summary, the forward computation of the gram layer involves multiple stages: (1) down-sample feature maps, (2) calculate the gram matrix, (3) remove the upper triangle and (4) connect to the next layer. Each stage can be implemented as a sublayer. These stages are illustrated in Figure 3.

Backward computation is required for each sublayer. The last sublayer can be realized as a fully-connected layer in most off-the-shelf deep learning tools such as Caffe [35]. The other three sublayers do not involve parameters, and we only need to compute the local sensitivity (also called local gradient), which is defined as the derivative of the loss function with respect to the total input to each neuron. For the two downsampling sublayers, this can be done by simply copying the local sensitivities in corresponding locations. The problem then reduces to calculating the local sensitivity for the first sublayer, whose forward calculation is described in (1). Since the input to each neuron is F_{mn}^l, the local sensitivity is

$$\begin{aligned}\frac{\partial E}{\partial F_{mn}^l} &= \sum_{i,j} \frac{\partial E}{\partial G_{ij}^l} \cdot \frac{\partial G_{ij}^l}{\partial F_{mn}^l} \\ &= \sum_i \frac{\partial E}{\partial G_{im}^l} \cdot \frac{\partial G_{im}^l}{\partial F_{mn}^l} + \sum_i \frac{\partial E}{\partial G_{mi}^l} \cdot \frac{\partial G_{mi}^l}{\partial F_{mn}^l} \\ &= \sum_i \frac{\partial E}{\partial G_{im}^l} \cdot F_{in}^l + \sum_i \frac{\partial E}{\partial G_{mi}^l} \cdot F_{in}^l \\ &= \sum_i \left(\frac{\partial E}{\partial G_{im}^l} + \frac{\partial E}{\partial G_{mi}^l} \right) \cdot F_{in}^l \end{aligned} \tag{2}$$

or in matrix form

$$\frac{\partial E}{\partial \mathbf{F}^l} = \left(\frac{\partial E}{\partial \mathbf{G}^l} + \frac{\partial E}{\partial \mathbf{G}^{l\top}} \right) \cdot \mathbf{F}^l. \tag{3}$$

Note that $\frac{\partial E}{\partial \mathbf{G}^l}$ is the local sensitivity in the second sublayer defined before, since $\mathbf{G}^l$ represents input to neurons in this sublayer. The local sensitivity $\frac{\partial E}{\partial \mathbf{F}^l}$ in the first sublayer will be used for calculating the local sensitivity and the derivative of the loss function with respect to parameters in the layer just before the gram matrix layer.

The outputs of multiple gram layers from different layers of the object pathway are then concatenated and go through a scaling layer,

$$\mathbf{y} = \mathbf{a} \odot \mathbf{x} + \mathbf{b}$$

where $\mathbf{x}$ and $\mathbf{y}$ denote the input and output of the scaling layer, respectively, $\mathbf{a}$ and $\mathbf{b}$ are trainable parameters which have the same length as $\mathbf{x}$, and $\odot$ denotes elementwise multiplication. This layer is designed to scale the output of gram layer, which is usually in the order of 10^9, to normal values. The output of this scaling layer $\mathbf{y}$ is fully connected to a multiplayer Perceptron (MLP) to give the prediction of the texture pathway (Figure 2).

Note that the texture pathway has multiple paths from the input layer to the output layer. In the forward pass, the outputs of some layers are fed into multiple layers. As a result, when doing backpropagation, the gradients from multiple layers will be added and backpropagated to the current layer. This is explained as follows. Suppose the intermediate result x is fed into two layers to get y and z, then according to the chain rule in calculus,

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial x} + \frac{\partial E}{\partial z} \cdot \frac{\partial z}{\partial x},$$

where E is the loss function for this pathway.

C. Merging

The object pathway and the texture pathway have their own cross-entropy error functions. Since many layers are shared by the two pathways, the gradients of the loss functions in these layers are summed over the two pathways. The final prediction of the proposed model is based on the averaged outputs of

Fig. 4. Example images in three datasets.

the softmax functions in the last layers of the two pathways. It is possible to merge the pathways by concatenating their penultimate layers and connect it to a single classifier. But we experimentally found that the current approach was better than this "early merging" approach.

One feature of the proposed model is that intermediate layers of a standard CNN (i.e., the object pathway) are forwarded to two separate pathways for classification. Similar strategies were used in other deep learning models including the deeply-supervised nets (DSN) [36] and GoogLeNet [2], though in the latter the auxiliary classifiers connected to intermediate layers were only present at training time but discarded at inference time. We experimentally found that, however, this "deep supervision" strategy was not the main cause of good performance of the proposed model. See Section IV-D for details.

IV. Experimental Results

The proposed approach is applied on two pretrained CNNs, AlexNet [11] and VGG-19 [3]. Three style recognition datasets are experimented: WikiPainting, Flickr Style, and AVA Style. All experiments were carried out on the Dell R720 server equipped with Nvidia Titan X GPUs using the deep learning tool caffe [35]. The source code is available.[1]

A. Datasets

1) Wikipainting: The WikiPainting dataset was introduced by Karayev *et al.* [1]. It contains 85,000 artistic images – mostly paintings – with 25 style/genre labels such as Impressionism, Realism, Romanticism and so on. Figure 4 shows several samples with their labels indicated below. Each style has more than 1,000 images. As far as we know, this is the largest dataset of artistic works with style labels. Unlike many other datasets, this dataset only provides a url for each image. But some url's are no longer valid, and some downloadable images are corrupted. After removing these samples, we obtained 82405 valid samples. The specified split ratio of train/validation/test is 60:20:20.

The distribution of samples in the 25 categories is highly unbalanced (Figure 5, left). In the sequel, the distribution of samples in the training set would also be highly unbalanced since the samples were randomly selected, which would deteriorate the performance of the models. In order to have a balanced distribution of the data, we cropped multiple patches from every image in the training set, and the number of patches from an image was roughly proportional to the inverse of the portion of its style category in the training set. The images were first resized to 256×256, then 224×224 patches were randomly cropped from them. The distribution of the cropped patches is shown in the right panel of Figure 5. Only one patch was randomly cropped in every image in the validation set. For every test image, four patches were cropped and the final prediction was based on the average probabilities over the four patches.

2) Flickr Style: The Flickr Style was also introduced by Karayev *et al.* [1]. It contains 80,000 images with 20 style labels. The labels were grouped into several categories (Optical techniques, Atmosphere, Mood,Composition styles, Color and Genre) and each category contains two to five labels. For example, the Atmosphere category has two labels, Hazy and Sunny. Figure 4 shows several samples with their labels indicated below. The train/validation/test split is specified and the ratio is 60:20:20.

The images were preprocessed by resizing to 256×256. Then four 224×224 patches were randomly cropped from every image. This step augmented the training set. For test images, we cropped four patches per image in the same way. The final prediction was the average of the predictions over four patches.

3) AVA Style: The AVA Style dataset is part of the AVA dataset which was introduced by Murray *et al.* [27]. It contains 14,079 photographs from www.dpchallenge.org with 14 labels: Complementary Colors, Duotones, High Dynamic Range, Image Grain, Light on White, Long Exposure, Macro, Motion Blur, Negative Image, Rule of Thirds, Shallow DOF, Silhouettes, Soft Focus, Vanishing Point. The train/test split is prescribed. The training set has 11,270 photographs and the test set has 2,809 photographs. In contrast to the other two datasets discussed above, the test image of AVA Style may have multiple labels. However, some photographs in the test set do not have labels, which were excluded in our experiments (same as in [5]). This resulted in 2,573 photographs

[1] https://github.com/kevinkingo/ImageStyle-TwoPath

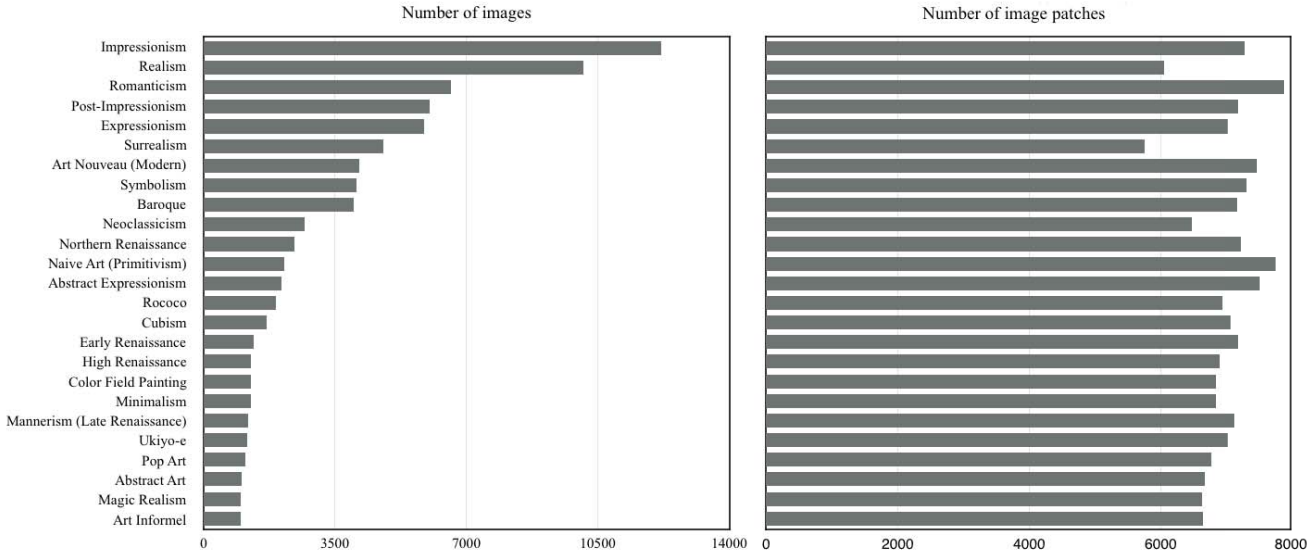


Fig. 5. The distribution of images in WikiPainting categories. *Left*: The distribution on the entire dataset; *Right*: The distribution on the training set after data balance operation.

TABLE I
ALEXNET SETTING

input	connection	learning rate ratio
conv11-96 maxpool		0.17
local response normalization	→ gram layer 1	
conv5-256 maxpool		0.2
local response normalization	→ gram layer 2	
conv3-384	→ gram layer 3	0.25
conv3-384	→ gram layer 4	0.33
conv3-384		0.5
maxpool	→ gram layer 5	
FC-4096		1
FC-4096		1
Softmax		10

The numbers in the right column are learning rate ratios over the base learning rate during the second fine-tuning stage for TPN and MNet. Note that the learning rates for the last three layers are applicable only in MNet.

TABLE II
VGG-19 SETTING

input	connection	learning rate ratio
conv3-64 conv3-64		0.17
maxpool	→ gram layer 1	
conv3-128 conv3-128		0.2
maxpool	→ gram layer 2	
conv3-256 conv3-256 conv3-256 conv3-256		0.25
maxpool	→ gram layer 3	
conv3-512 conv3-512 conv3-512 conv3-512		0.33
maxpool	→ gram layer 4	
conv3-512 conv3-512 conv3-512 conv3-512		0.5
maxpool	→ gram layer 5	
FC-4096		1
FC-4096		1
Softmax		10

The numbers in the right column are learning rate ratios over the base learning rate during the second fine-tuning stage for TPN and MNet. Note that the learning rates for the last three layers are applicable only in MNet.

in the test set. We did a random train/validation split on the original training set with ratio 6:1.

The images were preprocessed by resizing to 256×256. Then eight 224×224 patches were randomly cropped from every image to augment the training data. For test images, we cropped eight patches per image in the same way. The final prediction was the average of the predictions over eight patches.

B. Settings

We tested our method on two deep CNNs, AlexNet [11] and VGG-19 [3], which were pretrained on the ImageNet classification dataset. Their structures are shown in Tables I and II, respectively, where the convolution parameters are denote as "conv⟨kernel window size⟩-⟨channel number⟩". The ReLU layers [11] are omitted in the tables. For both networks, five intermediate layers were used to calculate texture features, which are denoted by gram layer 1 to gram layer 5 in the tables.

In the object pathway, only the last layer FC-1000 was substituted with a fully connected layer which had proper output size. In the texture pathway, a multi-layer perceptron (MLP) with two hidden layers were used, which took the output of the scaling layer as input. The two hidden layers had 2048 and 1024 neurons respectively. The output was the softmax function.

For both networks, the second and third gram layers were down-sampled to 64×64, and the other gram layers were down-sampled to 32×32.

Our model has two pathways. Although the final prediction was based on the average of their predictions, the two pathways were jointly fine-tuned during training. To have a deep understanding of the performance of the model, we also measured the performance of the two pathways separately, which are called *Object Pathway Network (OPN)* and *Texture Pathway Network (TPN)*, respectively. They were trained using the corresponding cross-entropy errors, respectively. For example, for measuring the performance of OPN, only the cross-entropy error on the object pathway was used, but those layers shared by the two pathways took part in the training process. For convenience, we denote the entire model with the two pathways by *Merged Network (MNet)*.

The OPN, TPN and MNet were fine-tuned from pre-trained deep CNNs separately. Stochastic gradient descent optimization method was used. We set a base learning rate $\alpha_0 = 10^{-4}$, which was multiplied by 0.1 once the accuracy on the validation set did not increase. The layers which are not part of the original CNNs were initialized using the method proposed by Glorot and Bengio [37] and their learning rate was 10 times as the base learning rate. The learning rates for different networks were set using different strategies, which are specified in what follows.

For OPN, the learning rate for the last layer was $10\alpha_0$ and α_0 for other layers.

TPN and MNet have branching paths from input to output. As mentioned in Section III, during backward computation in training, a branching layer should add the gradient from multiple layers next to it in the forward computation. Since there are many branches in the two models, the gradients may accumulate to an improperly large value, which may lead to stability issue in training. To avoid is problem, we used smaller learning rates in lower convolutional layers and larger learning rates in higher convolutional layers during the fine-tuning stage so that the magnitude changes on different convolution layers were roughly the same.

Specifically, a two-stage fine-tuning strategy was used for TPN. In the first stage, we trained the fully connected layers only by fixing the weights shared with OPN. In other words, the learning rate was $10\alpha_0$ for the fully connected layers (because they were randomly initialized) and 0 for the convolutional layers. To avoid weight dependance between TPN and OPN, and ensure fair comparison, the convolutional weights were initialized with the weights pretrained on ImageNet classification dataset, not with the weights in OPN fine-tuned for style classification as described before. In the second stage, we trained all layers by changing the learning rates for the fully connected layers from $10\alpha_0$ to α_0 and the rates for the convolutional layers from 0 to the values specified in Tables I and II.

Since the calculation of gram matrix is relatively slow, to accelerate the training process for TPN, during the first fine-tuning stage, the input to the scaling layer was saved on the hard disk for reusing in later training epochs. Then each image went through the shared layers between OPN and TPN only once in this stage.

Similarly, a two-stage fine-tuning strategy was used for MNet. The first stage was designed to obtain better initial weights for training the texture pathway. Therefore it was the same as the the first stage for fine-tuning TPN. In the second stage, we trained all layers by using the learning rates described as follows: (1) the learning rate for the last layer in the object pathway was $10\alpha_0$ (because this layer was randomly initialized); (2) the learning rates for the fully connected layers specific to the texture pathway were α_0; and (3) the learning rates for the convolutional layers (shared by the two pathways) were specified in Tables I and II.

C. Evaluation Metrics

Three metrics were used to evaluate the performance of the models: *Sample Accuracy*, *Category Accuracy* and *Mean Average Precision*. The first two were for single-label classification only and third was for both single-label and multi-label classifications.

Sample Accuracy (SA) is defined as the ratio of correct predictions among all predictions, while every image has one and only one prediction for its category label. This metric is biased to categories with more samples then others. For unbalanced distribution of samples in different categories, this metric cannot reflect the performance of a model on all categories.

Category Accuracy (CA) is introduced to circumvent the problem of SA for unbalanced distribution of samples. We first calculate the accuracy for each category, that is, the number of correct predictions for a category divided by the number of samples in this category, then we average the results over categories.

Mean Average Precision (mAP): We need to define *Average Precision* first, which is the area under the precision-recall curve. Let D denote the dataset. For category i, we sort all the data in the descending order according to the predicted probability of belonging to that category (note that many classifiers, including softmax function used in this study, can produce a probability of belonging to a category), then each data j gets an index in category i, denoted by $\text{index}_i(j)$. Let $\delta_i(j)$ be an indicator function which is 1 if data j is in category i and 0 otherwise. Then AP for category i is defined as

$$\text{AP}(i) = \frac{\sum_j P@j(i) \cdot \delta_i(j)}{|\{j|j \in D, \delta_i(j) = 1\}|},$$

where

$$P@k(i) = \frac{|\{j|\text{index}_i(j) \leq \text{index}_i(k),\, j \in D, \delta_i(j) = 1\}|}{|\{j|j \in D, \text{index}_i(j) \leq \text{index}_i(k)\}|}.$$

mAP is the mean of AP for each category

$$\text{mAP} = \frac{1}{|T|} \sum_{i \in T} \text{AP}(i),$$

where T denotes the set of categories.

D. Results

1) WikiPainting: OPN performed slightly better than TPN for both AlexNet and VGG-19 (see Table III). By merging them together, state-of-the-art results were obtained. Note that though with two pathways, MNet do not have much more

TABLE III
RESULTS ON THE WIKIPAINTING DATASET

Methods	SA (%)	CA (%)	mAP(%)
Ours, AlexNet-based OPN	48.5	51.8	52.1
Ours, AlexNet-based TPN	41.7	46.9	44.5
Ours, AlexNet-based MNet	48.5	52.4	53.4
Ours, VGG-19-based OPN	52.2	54.3	56.1
Ours, VGG-19-based TPN	49.9	54.0	54.6
Ours, VGG-19-based MNet	**53.8**	**58.1**	**60.4**
VGG-19-based control	51.8	55.9	56.4
Bar et al. [4]	43.0	36.0	n/a
Karayev et al. [1]	n/a	n/a	47.3

TABLE IV
THE NUMBER OF PARAMETERS OF DIFFERENT NETWORKS

Model	Parameter Num (in million)
AlexNet-based OPN	58.4
AlexNet-based TPN	17.6
AlexNet-based MNet	72.3
VGG-19-based OPN	123.2
VGG-19-based TPN	17.3
VGG-19-based MNet	136.9
VGG-19-based control	153.6

TABLE V
CONFIGURATION OF THE POOLING LAYERS IN THE CONTROL MODEL

Layer	Kernel Size	Stride	Output Size	Output Size (MNet)
Pooling 1	28	28	1024	528
Pooling 2	16	8	4608	2080
Pooling 3	7	7	4096	2080
Pooling 4	10	4	2048	528
Pooling 5	5	2	2048	528

parameters than OPN, the standard CNN structure (Table IV). For example, using VGG-19 as the basic structure, the former has 137 million parameters, which are only 11.4% more than that of the latter.

The results of VGG-19 were better than those of AlexNet, which verified the importance of depth for deep learning models. For this reason, we only analyzed the results of VGG-19, as detailed below.

First of all, to dissociate the contributions of the texture features and the "deep supervision" strategy in the texture pathway, we designed a control model to verify our point. In the control model, we substituted the gram layers in MNet with the average pooling layers,[2] which reduced the size of the feature maps. For fair comparison, the pooling sizes and strides were chosen such that the number of parameters in the control model was *not smaller* than that of MNet. The specific configuration is shown in Table V. For comparison, the output sizes of the five gram layers are also shown in the table; see the rightmost column. Other parts of the network remained the same. The same training strategy as before was employed. The control model performed similarly to VGG-19-based OPN (see Table III) on this dataset, even its number of parameters exceeds that of OPN for a huge amount. For SA it performed worse than VGG-19-based OPN but for CA and mAP it performed better. We found that the SA of the side pathway was 32.8%, which was quite low compared with that of the texture pathway in MNet, 49.8%. This partly explains why the addition of the side pathway in the control model has not led to much improvement to the original CNN, that is, VGG-19-based OPN. These results show that it were the texture features in the gram layers that led to good performance of the proposed model, not the "deep supervision" strategy.

Then we investigated relative contributions of the two pathways in MNet. The performances of OPN and TPN provided some information, but since they were not trained jointly, the comparison might be unfair. As both pathways in the proposed model MNet made their own predictions, we compared these predictions. The accuracies of the two pathways of the model based on VGG-19 are shown in Figure 6. For

[2]We empirically found that average pooling performed better than max pooling.

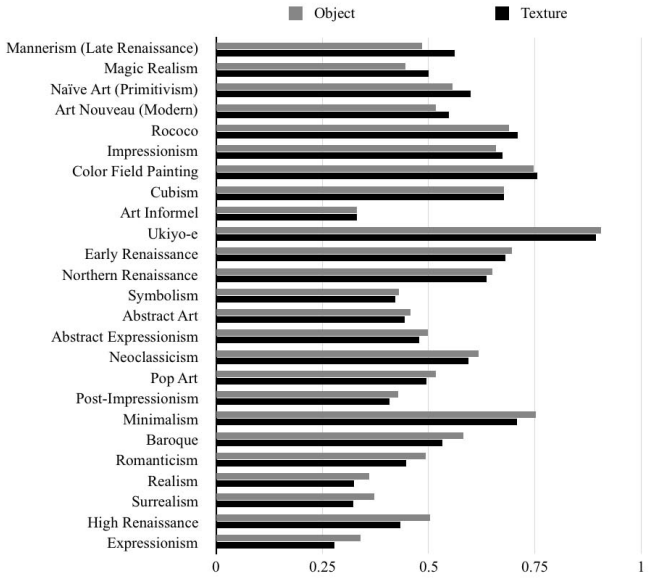


Fig. 6. SA of the two pathways in the VGG-19-based MNet on the WikiPainting test set.

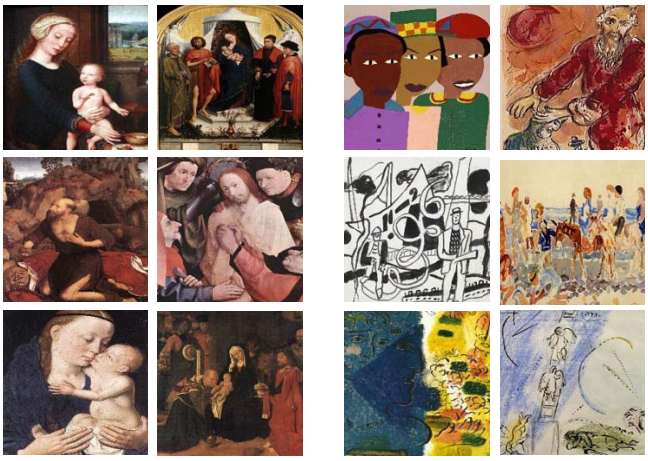


Fig. 7. Sample images from two style categories in the WikiPainting dataset which had higher SA for one pathway than the other pathway in the VGG-19-based MNet. *Left:* The object pathway had higher SA. *Right:* The texture pathway had higher SA.

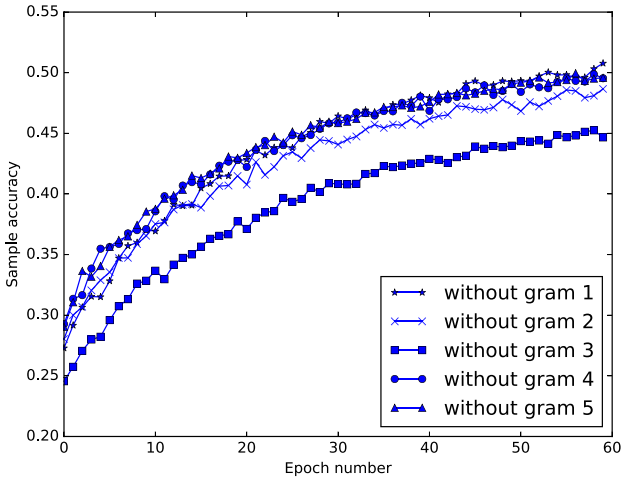


Fig. 8. SA of the TPN after removing a gram layer.

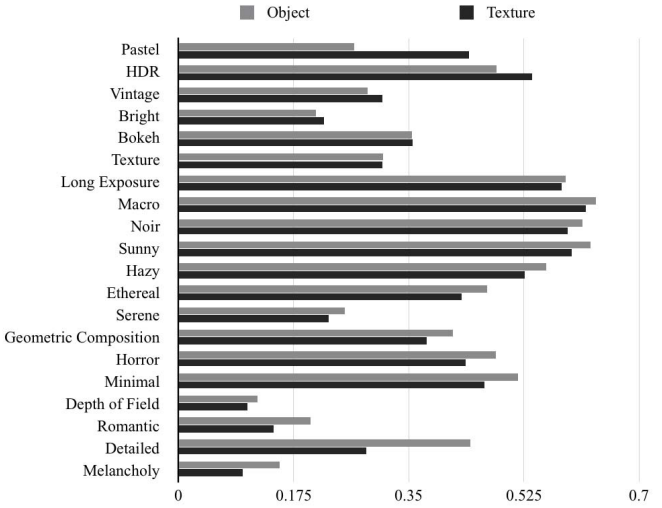


Fig. 9. SA of the two pathways in the VGG-19-based MNet on the Flickr Style test set.

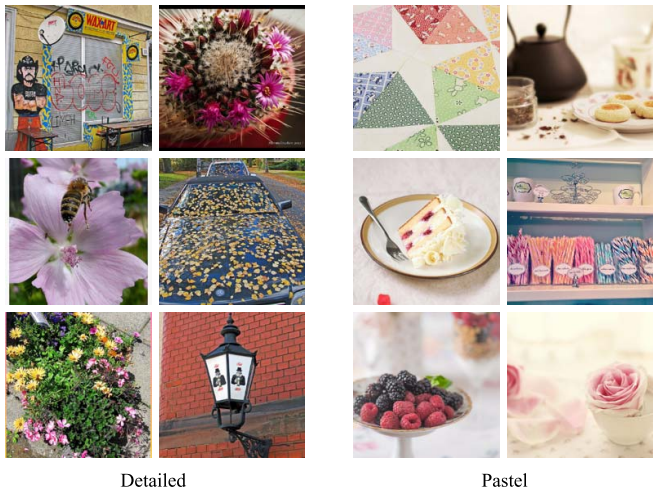


Fig. 10. Sample images from two style categories in the Flickr Style dataset which had higher SA for one pathway than the other pathway in the VGG-19-based MNet. *Left:* The object pathway had higher SA. *Right:* The texture pathway had higher SA.

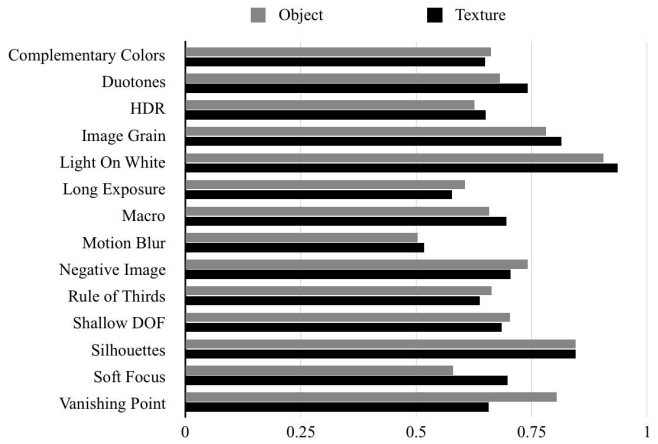


Fig. 11. SA of the two pathways in the VGG-19-based MNet on the AVA Style test set.

most styles the accuracies were similar, but for several styles, the accuracy of one pathway was much higher than that of the other. For example, for the style Northern Renaissance the object pathway had higher accuracy, but for the style Naive Art (Primitivism) the texture pathway had higher accuracy. Some representative paintings from the two styles are shown in Figure 7. The Northern Renaissance paintings often depict religious occasions, which are easier to be captured by the object pathway. Therefore the object pathway played a more important role in this case. In contrast, Naive Art paintings are characterized by hasty strokes, though sometimes they also contain people. This information is easier to be captured by the texture pathway. Therefore the texture pathway played a more important role in this case.

Both MNet and TPN have five gram layers, whose inputs are from different layers of CNN. We investigated relative contributions of the layers to style prediction based on TPN. We trained TPN by removing a gram layer and repeated the process for each gram layer. For fair comparison, the gram matrices of all layers were down-sampled to the same dimension 32×32. The results on the validation set showed that removing the second and third gram layers affected most to the performance of the model (Figure 8). It suggests that texture features produced based on middle-level features of CNN would be most important for texture discrimination. That is why we down-sampled the gram matrices in the second and third gram layers to 64×64 and the gram matrices in other layers to 32×32.

2) Flickr Style: The results on this dataset showed that though the performances of OPN and TPN based on AlexNet were not as good as that of the model proposed by Karayev *et al.* [1], the performance of the merged network MNet was better (Table VI). Using better object features or texture features alone also led to higher accuracy than Karayev et al.'s model (see VGG-19-based OPN and VGG-19-based TPN in Table VI). Their combination resulted in even higher accuracy with a significant margin (see VGG-19-based MNet in Table VI). To the best of our knowledge, Karayev et al.'s model is the only model evaluated on this dataset and reported in the literature. The proposed

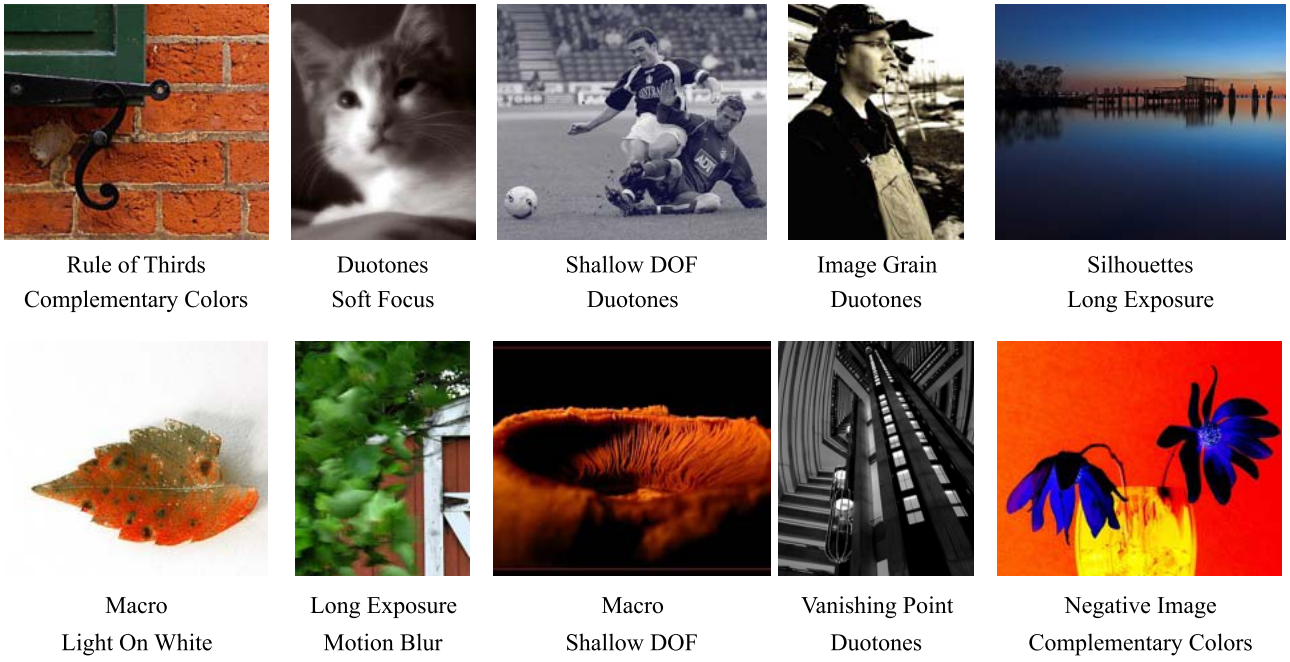


Fig. 12. Sample images in the AVA Style test set having two labels predicted correctly by two pathways in the VGG-19-based MNet. The upper label under each image is predicted by the object pathway and the lower label is predicted by the texture pathway.

TABLE VI
RESULTS ON THE FLICKR STYLE DATASET

Model	SA (%)	CA (%)	mAP(%)
Ours, AlexNet-based OPN	37.6	37.6	36.4
Ours, AlexNet-based TPN	36.8	36.8	35.1
Ours, AlexNet-based MNet	38.9	38.9	38.2
Ours, VGG-19-based OPN	39.7	39.7	38.4
Ours, VGG-19-based TPN	38.5	38.5	37.4
Ours, VGG-19-based MNet	**41.6**	**41.6**	**41.0**
VGG-19-based control	40.5	40.5	39.1
Karayev et al. [1]	n/a	n/a	36.8

TABLE VII
RESULTS ON THE AVA STYLE DATASET

Model	mAP(%)
Ours, AlexNet-based OPN	59.1
Ours, AlexNet-based TPN	56.2
Ours, AlexNet-based MNet	62.0
Ours, VGG-19-based OPN	62.1
Ours, VGG-19-based TPN	59.6
Ours, VGG-19-based MNet	**65.5**
VGG-19-based control	64.7
Murray et al. [27]	53.9
Karayev et al. [1]	58.1
Lu et al. [5]	64.1

MNet based on VGG-19 performed better on this dataset. In addition, this MNet achieved better results than the control model, which further validates the effectiveness of the gram layers.

The prediction accuracies of the object pathway and texture pathway based on VGG-19 for individual styles are plotted in Figure 9. The accuracies of the two pathways differed much more than the results on the WikiPainting dataset. For instance, the accuracy of the texture pathway for Pastel style was almost twice of that of the object pathway. In fact, the images of the Pastel style contain a diverse set of objects (Figure 10), which are difficult to be classified into one category by using the object pathway. In contrast, images of style Detailed often contain objects which have a lot of sharp edges (Figure 10), such as flowers, leaves, or brick walls. In this case, the object pathway contributed more to the recognition performance.

3) AVA Style: The AVA Style dataset is a little different from the other two datasets as the test images have multiple labels. The SA and CA defined in Section IV-C do not apply here. Therefore only mAP was used to measure the performance of the models (Table VII). The two MNets based on AlexNet and VGG-19 achieved higher mAP than their single pathway networks OPNs or TPNs, which verified the assumption that object features and texture features are complementary for this task. The best result was obtained by MNet based on VGG-19, which was higher than that of the state-of-the-art model proposed by Lu *et al.* [5]. The control model performed better than VGG-19-based OPN but worse than VGG-19-based MNet.

The prediction accuracies of the object pathway and texture pathway based on VGG-19 for individual styles are plotted in Figure 11. Again, for some styles the object pathway had more correct predictions and for some styles the texture pathway had more correct predictions. Since there may be multiple labels for a single image, we investigated how the two pathways perform differently on the same images. Among 2,573 test images, there are 1,086 images having more than one label. By outputting top two predictions, the MNet based on VGG-19 correctly predicted two labels for 412 images. Considering that the prediction was the average of the predictions of two pathways, we further analyzed individual

predictions of the two pathways for these 412 images. Among them, 149 images' two labels were predicted correctly by the two pathways, respectively (Figure 12). The gap between these two numbers (412 and 149) indicates the contribution of the cooperation between the two pathways.

V. Conclusion and Discussion

In this paper, we present a deep learning model with two pathways for image style recognition, which extract object features and texture features of images respectively. Two deep CNNs, AlexNet and VGG-19 were adapted for this purpose. Experiments showed that the two kinds of features contained complementary information for the style of an image as the two-pathway architecture always achieved better results than single pathway architectures. In addition, the model based on VGG-19 achieved the state-of-the-art results on three benchmark datasets, WikiPainting, Flickr Style and AVA Style.

The key to the success of the proposed model is the texture features, which essentially represent the correlations between object features on images. It was originally used to change the style of images [6], but there these features were fixed as a reference style and the task was to modify an image to match it. In our model, these features need to be learned. Therefore a unified training method is proposed to learn the object features and texture features together according to the given labels.

One limitation of the approach for extracting texture features is intense computation. For this reason we used only a subset of object features for producing these texture features in experiments. The good performance of the model indicates much redundance among the full set of texture features. Anyway, this subsampling method is somehow arbitrary. One possible improvement is to choose an optimal set of object features. But it is favorable to have a more concise representation of texture or style than gram matrix. We leave it as future work.

Acknowledgment

The authors would like to thank the editor and the anonymous reviewers for their critical and constructive comments and suggestions. They would also like to thank Fangzhou Liao for useful discussion.

References

[1] S. Karayev *et al.*, "Recognizing image style," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2014, pp. 1–20.

[2] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

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

[4] Y. Bar, N. Levy, and L. Wolf, "Classification of artistic styles using binarized features derived from a deep neural network," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2014, pp. 71–84.

[5] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Z. Wang, "Deep multi-patch aggregation network for image style, aesthetics, and quality estimation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 990–998.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. (Aug. 2015). "A neural algorithm of artistic style." [Online]. Available: https://arxiv.org/abs/1508.06576

[7] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[8] M. Sarfraz, *Computer Vision and Image Processing in Intelligent Systems and Multimedia Technologies*. Hershey, PA, USA: IGI Global, 2014.

[9] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 262–270.

[10] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[12] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Imagenet classification with deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2016, pp. 770–778.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[15] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1440–1448.

[16] 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.

[17] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009.

[19] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (Dec. 2014). "Striving for simplicity: The all convolutional net." [Online]. Available: https://arxiv.org/abs/1412.6806

[20] S. Ioffe and C. Szegedy. (Feb. 2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] D. Keren, "Painter identification using local features and naive Bayes," in *Proc. 16th Int. Conf. Pattern Recognit.*, vol. 2. Aug. 2002, pp. 474–477.

[23] L. Shamir, T. Macura, N. Orlov, D. M. Eckley, and I. G. Goldberg, "Impressionism, expressionism, surrealism: Automated recognition of painters and schools of art," *ACM Trans. Appl. Perception*, vol. 7, no. 2, p. 8, Feb. 2010.

[24] J. S. Lim, *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1990.

[25] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *EEE Trans. Syst., Man, Cybern.*, vol. 3, no. 6, pp. 610–621, Nov. 1973.

[26] G. Liu *et al.*, "Inferring painting style with multi-task dictionary learning," in *Proc. 24th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2015, pp. 2162–2168.

[27] N. Murray, L. Marchesotti, and F. Perronnin, "AVA: A large-scale database for aesthetic visual analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2408–2415.

[28] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn.*, 2001, pp. 341–346.

[29] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graph. (ToG)*, vol. 22, no. 3, pp. 277–286, 2003.

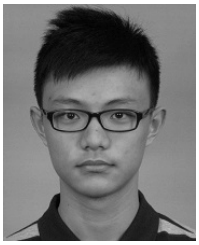
[30] B. Julesz, "Visual pattern discrimination," *IRE Trans. Inf. Theory*, vol. 8, no. 2, pp. 84–92, Feb. 1962.

[31] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn.*, 1995, pp. 229–238.

[32] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Comput. Vis.*, vol. 40, no. 1, pp. 49–70, Oct. 2000.

[33] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1–9.

[34] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.
[35] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[36] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. AISTATS*, 2015, vol. 2. no. 3, p. 6.
[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[38] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *Proc. Int. Conf. Learn. Represent.*, Toulon, France, Apr. 2017.



Tiancheng Sun is currently pursuing the bachelor's degree with the Institute of Interdisciplinary Information Science, Tsinghua University. He is going to begin the Ph.D. degree with the University of California, San Diego in 2017. His research interests include machine learning, material representation, and light field capture and display.

Yulong Wang received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2015. He is currently the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include computer vision and deep learning.

Jian Yang received the Ph.D. degree in pattern recognition and intelligence systems from the Nanjing University of Science and Technology (NUST) in 2002. In 2003, he was a Post-Doctoral Researcher with the University of Zaragoza. From 2004 to 2006, he was a Post-Doctoral Fellow with the Biometrics Centre, Hong Kong Polytechnic University. From 2006 to 2007, he was a Post-Doctoral Fellow with the Department of Computer Science, New Jersey Institute of Technology. He is currently a Chang-Jiang Professor with the School of Computer Science and Technology, NUST. He has authored over 100 scientific papers in pattern recognition and computer vision. His journal papers have been cited more than 4000 times in the ISI Web of Science, and 9000 times in the Web of Scholar Google. His research interests include pattern recognition, computer vision, and machine learning. He is a fellow of IAPR. He is also an Associate Editor of the *Pattern Recognition Letters*, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and *Neurocomputing*.

Xiaolin Hu (S'01–M'08–SM'13) received the B.E. and M.E. degrees in automotive engineering from the Wuhan University of Technology, Wuhan, China, and the Ph.D. degree in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2001, 2004, and 2007, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include artificial neural networks, computer vision and computational neuroscience. He is also an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.