COLOR NAMES LEARNING USING CONVOLUTIONAL NEURAL NETWORKS

Yuhang Wang, Jing Liu, Jinqiao Wang, Yong Li and Hanqing Lu

The National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, China
{yuhang.wang, jliu, jqwang, yong.li, luhq}@nlpr.ia.ac.cn

ABSTRACT

In this paper, we propose a two-stage CNN-based framework to learn color names from web images, aiming to predict color names for tiny image patches. To deal with the noisy labels widespread in web images, we propose a self-supervised CNN (SS-CNN) model in the first stage. The SS-CNN model is trained on image patches with their own color histograms as supervision information. Thus its outputs are able to reflect the color characteristics of images without the influence of the noisy labels. In the second stage, we finetune the SS-CNN model to learn the mapping from image patches to color names, where the patch labels are inherited from its father images. Besides, sample selection is imported iteratively in turns with the finetuning process, which helps filtering out some noisy samples and further improves the model accuracy. Our model shows high representation ability to colors and achieves better performance of color naming compared with the state-of-the-art methods.

Index Terms— Color naming, self-supervised CNN

1. INTRODUCTION

Color is one of the most important attributes used to describe images in vision tasks like image retrieval. To make the color description more consistent with human cognition, color naming approaches to link color values with different linguistic labels receive considerable attention by researchers. As a pioneer work, Berlin *et.al* [1] defined 11 basic color names with extensive universality, including black, blue, brown, green, grey, orange, pink, purple, red, white, and yellow. Based on the definition, several studies on color naming strive for color value mapping to the basic color names. This is also our focus in this paper.

To model the associations between color values and color names, some early work is built on color stimuli such as [2], while recent work tends to employ web images which are collected through search engines and thus cost-free. Most of these methods are based on statistical models. [3] used multinomial probability distributions to describe the likelihood of a color value to be predicted to a color name, while [4] employed a mixture of Gaussian distributions. [5] and [6] are the most representative work in recent years, which turned to topic model and improved PLSA [7] to learn the pixel-wise prediction. [8] introduced a randomized HSL transformation to make artificial images approach natural color distributions, and implemented a χ^2 ranking to initialize their model and help removing outliers from the training data.

We also construct our model on web images, but unlike previous work, we intend to make a "block-wise" prediction in this paper, meaning that we will predict the color names for tiny image patches according to the dominant colors in them. Compared with isolated pixels, image patches contain more abundant semantic information and always show more robustness to noises. Moreover, we employ a discriminative model and treat the task of color naming as a classification one to enhance the discriminability of different color names. Convolutional Neural Network (CNN) [9][10] becomes our first choice for classification, due to its amazing performance achieved in vision tasks [10][11]. An intuitive idea for this task is to train a CNN model directly on the image patches labeled with color names. However, the intuitive way is improper to our task, because the training images explored from web are not labeled precisely, and furthermore the noisy labels are assigned with images, not to particular image patches. The contaminated training samples certainly will weaken the learnability of CNN. Generally, a typical solution, which has been adopted extensively and successfully by several researchers [11][12], turns to a pre-trained CNN model on a large-scale dataset, like ImageNet, and uses its outputs as an representative feature description to enter the further learning task. However, for the case of color naming, there is no such dataset large enough for CNN to be well trained. And since the supervision information of ImageNet is object category, it is hard to be transfered to the color naming problem.

To address the problems above, in this paper, we propose a two-stage CNN-based learning framework for the task of color naming. In the first stage, we intend to pre-train a CNN model with generalizable feature extraction layers, which can preserve and concentrate the color characteristics of images without the influence of the noisy labels. For this purpose, we propose a self-supervised CNN (SS-CNN) model trained on image patches randomly selected from the training set, and use the color histograms of patches as their own supervision

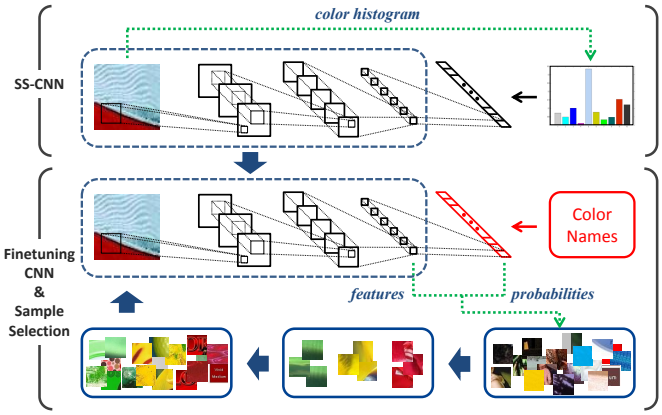


Fig. 1. Framework of our two-stage approach for color names learning. The two networks describe the SS-CNN and the finetuned classic CNN, respectively. The parts surrounded by blue dot lines represent the layers fixed during finetuning, while the red parts represent the variation. The flow chart in the bottom describes our sample selection process, which can be applied in turns with finetuning iteratively.

information to train the SS-CNN model. In this way, we can augment our dataset thousands of times larger and satisfy the demand of CNN training, yet without any cost. Meanwhile, with the guidance of the color histograms, the network will be trained to grasp color information from the input patches with no use of the noisy labels. In the second stage, we finetune the SS-CNN model with only the last layer retrained, on image patches labeled with the 11 color names. Sample selection is applied iteratively in turns with the finetuning process, in order to filter out part of noisy samples detrimental to our model. The flowchart of our integrated process is shown in Fig. 1. Experiments on a publicly available image set demonstrate the effectiveness of our proposed method.

2. OUR APPROACH

Our approach consists of two stages. We first train the SS-CNN model to grasp general color characteristics, and finetune it in the second stage to make it specialized for our task.

2.1. Training Self-supervised CNN Model

In this stage, we train the SS-CNN model on a newly built training set consisting of randomly selected image patches from the original training set. We choose the color histogram as the supervision to form the self-supervised learning structure, considering that it describes the probabilistic distribution of different colors in an image patch and hence provides effective guidance for the network to distinguish between them. In this paper, we employ a cluster-based color histogram for a better description on different colors, and modify the softmax loss function to fit the new supervision information.

We do not employ the commonly used color histogram algorithm which divides the color space equally, because the spatial distribution of colors in RGB space is uneven. Some colors may occupy very small areas and are too close to each other which are hard to be separated, such as black and grey, while some others are just the reverse. In our approach, we first implement a K-means cluster on all the pixels in the training set with each of them represented by its RGB values. Then we use the cluster centers as a codebook, and form the color histogram of a certain image patch by mapping each of its pixels into the corresponding cluster class. After that, we sharpen the color histogram by doing an exponentiation to each element of it to make the dominant colors more outstanding and weaken the other colors at the same time. Finally, we normalize the sharpened color histogram and use it as the supervised label vector of the image patch. Each element in the label vector can represent the probability of the image patch to be predicted to the corresponding cluster class.

With the sharpened color histogram as supervision, we modify the softmax loss function as [13] to minimize the KL-Divergence between the supervised label vector and the predictions made by the network. The new loss function used in the SS-CNN model can be written as:

$$Loss = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c}\hat{p_{ij}}log(p_{ij}) \quad (1)$$

$$p_{ij} = \frac{exp(f_j(x_i))}{\sum_{k=1}^{c} exp(f_k(x_i))} \quad (2)$$

where n is the batch size used when training the SS-CNN model, and c is the number of cluster classes which equals 128 in our settings. p_{ij} is the probability of image x_i to be predicted to class j by the network, while $\hat{p_{ij}}$ is the corresponding supervised probability in the label vector. $f(x_i)$ represents the output of the last fully connected layer of the network for image x_i, and $f_j(x_i)$ is the jth element of $f(x_i)$.

2.2. Finetuning with Classic CNN Structure

In this stage, we finetune the pre-trained SS-CNN model targeting the 11 color names proposed by [1], along with a sample selection method based on the features provided by the SS-CNN model. During the finetuning, the model is trained with all the feature extraction layers fixed and only the last fully connected layer is retrained, following the structure of classic CNN models.

Assuming that the pre-trained SS-CNN model is represented as $Net1$, we first finetune it directly on the training patches we collected above but label them with the color names of their "father" images. The newly obtained model is represented as $Net2$, which is certainly affected by the weakly labeled samples. So we apply sample selection in the following steps to discard part of the noisy samples injurious to our model.

	cars	dresses	pottery	shoes	overall
CH	65.36 ± 1.74	84.82 ± 1.29	74.00 ± 0.47	83.55 ± 0.52	76.93 ± 0.48
c-CNN	73.45 ± 1.47	79.18 ± 1.00	75.55 ± 0.29	88.00 ± 0.57	79.05 ± 0.56
SS-CNN*	70.55 ± 1.61	87.64 ± 0.64	78.45 ± 0.61	86.45 ± 0.52	80.77 ± 0.34
SS-CNN	73.18 ± 1.37	$\mathbf{91.82 \pm 0.43}$	83.36 ± 0.96	91.18 ± 0.86	$\mathbf{84.89 \pm 0.46}$
PLSA-bg [5]	71.82	86.36	**83.64**	**92.73**	83.64
χ^2 ranking [8]	**73.63**	88.18	79.01	**92.73**	83.41

Table 1. Color naming results in percentages on foreground objects of *Ebay color name set*

First, we predict color names for all the training patches with $Net2$. And for each color name, we select the top N samples with the highest scores from the patches correctly predicted, as "seeds" for this color name. Then, we extract the color features for all the training patches from the penultimate layer of $Net1$ which possess a powerful feature extraction ability. (It can also be accomplished with $Net2$ which shares the same feature extraction layers with $Net1$.) We average the features of the "seeds" within each of 11 color classes and regard them as the class centers. Finally, we select samples in each color class according to the Euclidean distance between them and the class center, and discard the samples that are too far away.

After getting the "purified" training patches, we finetune $Net2$ again with the same method as above. The sample selection and finetuning process can be applied in turns, to improve the performance of our model iteratively.

2.3. Training Details

Our network is implemented with Caffe [14]. We build the SS-CNN model with 3 convolutional layers and 1 fully connected layer, with 32, 96, 96, 128 nodes for each of them respectively. The 1st and 2nd convolutional layers are followed by ReLU layers, max pooling layers, and local response normalization layers [10], while for the 3rd convolutional layer, there are only a ReLU layer and a max pooling layer. The sizes of the convolutional filters are all 3×3, and the strides used are 1, 1, 2 respectively. For the max pooling layers, the sizes of pooling masks are also 3×3, and the strides are all 2. The sizes of image patches used in this paper are 37×37, which can be changed if needed, along with small adjustments on the network parameters.

3. EXPERIMENT

We pre-train and finetune our model on the *Google color name set* [5], and test it on the *Ebay color name set* [5] by predicting color names for the entire objects as well as pixels, to validate the performance of our approach in different situations.

Google color name set: The dataset contains 1100 images evenly divided into 11 basic color names which are chosen based on the study of [1]. All the images in the dataset are collected using Google Image and most of the images are weakly labeled.

Ebay color name set: The dataset contains 528 images evenly distributed in the 11 color classes from 4 object categories, of which 440 images are used as the testing set while the rest as the validation set. Images in this dataset are all collected from the the Ebay website and hand-segmented into foreground and background. Following the work in [5], we only consider foreground objects in our experiments.

3.1. Color Naming for Entire Objects

We first make an "object-level" prediction with our model to judge the color names for the entire foreground objects in *Ebay* images. For the evaluation, we randomly select 100 quadrate patches from the foreground of each image with stochastic sizes, then we resize them and use the average of their predictions as the prediction for the image. We repeat this process for 10 times, and report the mean accuracy along with the standard deviation.

The effectiveness of our model mostly relies on the pretrained SS-CNN model, which combines the CNN structure and the color histogram, so we use them seperately for this task as two baselines of our method. To use the CNN model alone, we train a classic CNN model (c-CNN) directly on image patches labeled with the 11 color names, and for the color histogram (CH), we train a softmax classifier on it, which is the same with the last layer of our model. Sample selection is applied for both of the two methods. We report the results of our model (SS-CNN), as well as the results with no use of sample selection (SS-CNN*). For other baselines, we compare our method with the PLSA-bg model provided by [5] and the χ^2 ranking model proposed by [8].

The performance of different methods is shown in Table 1. It can be found that the model trained directly on color histograms achieves the worst results, while the SS-CNN based model achieves the best, which demonstrates that the CNN model supervised by traditional hand-craft features can learn a more powerful feature representation which is consistent with the supervisory feature but possesses a stronger expression ability. Compared with the classic CNN based model, the SS-CNN based model without sample selection already shows a better performance. And sample selection further improves our results significantly by about 4% on overall accuracy. The obvious outperformance over the classic CNN

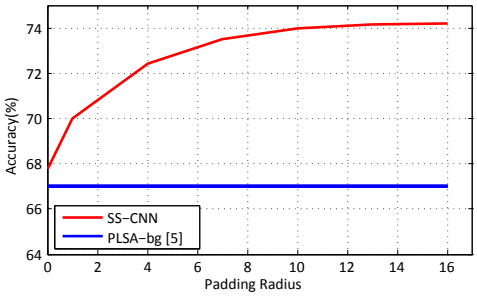
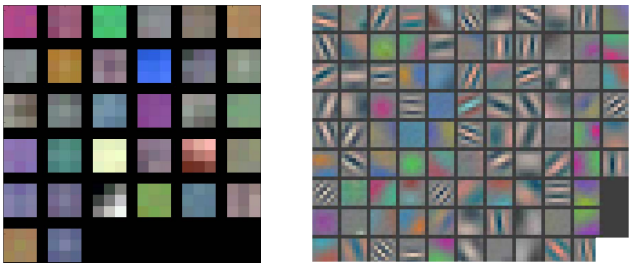


Fig. 2. Pixel classification accuracy with different padding radiuses



(a) Our model (b) Zeiler's model

Fig. 3. Visualization of filters in the first convolutional layer of our model, compared with Zeiler's model [12].

based model also proves that our model is much more robust and thus achieves a better learning on color characteristics, benefitting from the self-supervised structure.

While compared with [5] and [8], our model also shows the best performance on overall accuracy. However, for single categories, our model performs best in only one of them. But it should be noticed that, in the category of "dresses", our model achieves a result far exceeding the other two methods, and in the other categories, our model also achieves results very close to the best results, which is impossible for the other two methods. It shows the general applicability of our model under different reflection properties of objects and in various scenes. Moreover, the model of [8] is trained on *Google-512* dataset which is 5 times as large as our training set, and the model of [5] demands several preprocessing steps. In contrast, our approach can be accomplished on a smaller dataset with no need of any pretreatment, yet achieves better performance.

3.2. Color Naming for Pixels

We also conduct a "pixel-level" experiment using our model to predict color names for pixels. To evaluate the performance of our model, we traverse all the foreground pixels of *Ebay* images and calculate the classification accuracy. For the baseline, we cite the pixel classification result reported in [5] with the PLSA-bg method.

For a single pixel, we can simply duplicate it into a patch with the size fit for our network and predict its color name. And we have also tried predicting the color name for a pixel with the help of its surrounding pixels, since the adjacent pixels often have similar colors with the central pixel in a real-world image, which helps improving our accuracy. We resize the small patch made by our object pixel together with its surrounding pixels and predict its color name with the same method as above. The padding radius ranges from 0 to 16 in our settings, where padding 0 pixel means making predictions with a single pixel.

Our results are shown in Fig. 2. It can be found that our approach also achieves satisfying results in pixel classification. With no padding pixels, our approach has already achieved a result a little higher than [5]. And by padding with just 1 pixel around, which makes a single pixel into a 3×3 patch, our result shows an obvious improvement. As the padding radius becomes larger, the result of our approach rises and then tends to be stable, because the padding pixels increase the robustness of our prediction. When the padding radius becomes too large, the results also benefit from the consistency of the foreground colors. The results demonstrate the effectiveness of our model to describe colors for more refined tasks such as pixels or small image regions.

3.3. Filter Visualization

To intuitively investigate the ability of our model in grasping color characteristics, we visualize the filters in the first layer of the SS-CNN model. Compared with the filters achieved in previous work, such as [12] which trains the CNN model on the ImageNet dataset, our filters show more color-specificity.

Fig. 3 shows the comparison of our filters with Zeiler's filters [12]. It can be found that our model shows more sensibility on various colors while almost ignores the edge information of the images, which could be an explanation for the color feature extraction ability of our model.

4. CONCLUSION

We propose a two-stage framework for color names learning, which predicts color names for tiny image patches by grasping the dominant colors in them. To avoid the influence of noisy labels, we pre-train a self-supervised CNN model guided by the color histograms which shows fairly good generality and feature extraction ability. Finetuning is then applied for color names classification along with sample selection to further improve the performance. Extensive experiments have shown the effectiveness of our model in color naming under different situations.

5. ACKNOWLEGMENT

This work was supported by 973 Program (2012CB316304) and National Natural Science Foundation of China (61332016, 61272329, and 61472422).

6. REFERENCES

[1] Brent Berlin and Paul Kay, "Basic color terms: their universality and evolution," *Berkeley: University of California*, 1969.

[2] Robert Benavente, Francesc Tous, Ramon Baldrich, and Maria Vanrell, "Statistical modelling of a colour naming space," in *Conference on Colour in Graphics, Imaging, and Vision*, 2002, pp. 406–411.

[3] Jeffrey Heer and Maureen Stone, "Color naming models for color selection, image editing and palette design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012, pp. 1007–1016.

[4] Dimitris Mylonas, Lindsay MacDonald, and Sophie Wuerger, "Towards an online color naming model," in *Color and Imaging Conference*, 2010, pp. 140–144.

[5] Joost Van De Weijer, Cordelia Schmid, and Jakob Verbeek, "Learning color names from real-world images," in *Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[6] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus, "Learning color names for real-world applications," *Image Processing, IEEE Transactions on*, vol. 18, pp. 1512–1523, 2009.

[7] Thomas Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 1999, pp. 50–57.

[8] Boris Schauerte and Gernot A Fink, "Web-based learning of naturalized color models for human-machine interaction," in *International Conference on Digital Image Computing: Techniques and Applications*, 2010, pp. 498–503.

[9] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[12] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014*, 2014, pp. 818–833.

[13] Yunchao Gong, Yangqing Jia, Thomas Leung, Alexander Toshev, and Sergey Ioffe, "Deep convolutional ranking for multilabel image annotation," *arXiv preprint arXiv:1312.4894*, 2013.

[14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.