Distinguishing Computer-Generated Images from Natural Images Using Channel and Pixel Correlation

Ruisong Zhang[1,2], Weize Quan[1,2], Lubin Fan[3], Liming Hu[4], and Dong-Ming Yan[1,2,4*], Member, CCF, ACM, IEEE

[1]*National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*

[2]*University of Chinese Academy of Sciences, Beijing 100049, China*

[3]*Alibaba Group, Hangzhou Zhejiang 310023, China.*

[4]*State Key Laboratory of Hydro-Science and Engineering, Tsinghua University, Beijing 100084, China*

E-mail: zhangruisong2019@ia.ac.cn; qweizework@gmail.com; lubin.flb@alibaba-inc.com; gehu@tsinghua.edu.cn; yandongming@gmail.com

Abstract With the recent tremendous advances of computer graphics rendering and image editing technologies, computer generated fake images, which in general do not reflect what happens in the reality, can now easily deceive the inspection of human visual system. In this work, we propose a convolutional neural network (CNN)-based model to distinguish computer-generated (CG) images from natural images (NIs) with channel and pixel correlation. The key component of the proposed CNN architecture is a self-coding module that takes the color images as input to extract the correlation between color channels explicitly. Unlike previous approaches that directly apply CNN to solve this problem, we consider the generality of the network (or subnetwork), *i.e.*, the newly introduced hybrid correlation module can be directly combined with existing CNN models for enhancing the discrimination capacity of original networks. Experimental results demonstrate that the proposed network outperforms state-of-the-art methods in terms of classification performance. We also show that the newly introduced hybrid correlation module can improve the classification accuracy of different CNN architectures.

Keywords natural image, computer-generated image, channel and pixel correlation, convolutional neural network

1 Introduction

Natural images (NIs), captured by digital cameras, can accurately and objectively record the real-world scenes and are considered as an important carrier of visual information. In our daily life, NIs are often used for the accurate dissemination of news and the effective recording of evidence. Because of the strong artistic and realistic expression, computer-generated (CG) images are also an important carrier of visual information. With the advances in computer graphics rendering techniques, it becomes much easier to generate CG images with strong photorealism. It becomes more and more difficult to distinguish CG images from NIs by naked human eyes, as shown in Fig. 1. Although CG images sometimes can give good visual experience, they also potentially bring security problems to news and justice.

(a) (b)

Fig.1. Two pairs of images about landscape and architecture. (a) Computer generated images. (b)Nature images. These images are from the SPL2018 dataset [1].

Consequently, distinguishing CG images from NIs has become an important research problem in multimedia security and visual media processing community. To address

Regular Paper

this problem, a lot of efforts have been made using the standard machine learning framework [2–7], which usually consists of two stages: manually designing discriminative features and training the classifiers. These hand-crafted features more or less depend on human prior knowledge and sometimes can achieve limited performance, especially for complex datasets. Considering the strong learning capacity and unified "end-to-end" optimization of convolutional neural network (CNN), recent approaches proposed CNN-based models [1, 8–12] to distinguish CG images from NIs, and achieved better detection performance compared with classic approaches [2–7]. However, the generality of the CNNs, or the module of a network, *i.e.*, a module can be directly combined with existing CNN-based models and further improve their performance, is rarely considered in previous works.

In this paper, we propose a new "end-to-end" CNN architecture to address this problem. We expose different characteristics between NIs and CG images using the channel and pixel correlation information. A self-coding module is designed to deeply explore the correlation between three color channels, and then several convolutional layers without pooling operation are applied to better extract the correlation between image pixels. The newly designed network is called ScNet (Self-coding Network). The main contributions of our work include:

- We design a self-coding module at the beginning of the CNN to explicitly extract correlation information between image color channels and thus enhance the discrimination capacity of the whole CNN model. Experimental results show that our proposed CNN model outperforms the state-of-the-art methods in terms of classification performance.
- We combine the proposed self-coding module with consecutive convolutional layers (without pooling operation) to extract the low-level features of input images, and this subnetwork can be directly installed in the beginning of other existing CNN models. Consequently, the performance of these existing CNN models are further improved, which validates the generality of our designed ScNet.

The rest of paper is organized as follows. Section 2 discusses relevant existing literature, including hand-crafted-feature-based and deep-learning-based methods. Section 3 presents the motivation and details of our proposed network. Section 4 validates our network design and compares with state-of-the-art approaches. Section 5 explains the self-coding module via advanced visualization techniques. Section 6 draws the conclusions.

2 Related Work

Existing approaches for distinguishing CG images from NIs mainly can be classified into two categories: hand-crafted-feature-based approaches and deep-model-based approaches. The former usually involves designing hand-crafted features (in either the spatial domain or a transformed domain) and training the classifiers (*e.g.*, support vector machine (SVM)). The latter often follows a generic "end-to-end" framework by using deep neural networks.

2.1 Methods Based on Hand-crafted Features

Several hand-crafted-feature-based approaches have been proposed to distinguish CG images from NIs. Based on the differences in the generation process of NIs and CG images in target models, light transmission, and acquisition methods, Ng *et al* [2] proposed to identify CG images using features aided by fractal and differential geometry. Lyu and Farid [3] proposed a feature by combining first-order and higher-order wavelet statistics to distinguish between photographic and photorealistic images. Chen *et al* [4] designed the feature using the statistical moments of wavelet characteristic function in the HSV color space. Gallagher and Chen [5] captured the original decoding traces of

photographic images to separate CG images from NIs and achieved a good detection performance. Based on the previous studies, Sankar *et al* [6] proposed a combination of features, including periodic correlation based features [13], color histogram features [14], momentum-based statistical features in YCbCr color space [4] and local patch statistical features [2]. From the image perception perspective, Pan *et al* [15] proposed the discriminative features are derived from fractal dimension and several generalized dimensions to respectively capture the difference in color and the detailed texture between CG images and NIs. Zhang *et al* [16] proposed a method based on the statistical property of local image edge patches. Li *et al* [17] explored a local texture descriptor, *i.e.*, uniform gray-scale invariant local binary patterns (LBP) [18], to classify NIs and CG images. Peng *et al* [7] extracted the residuals of images after Gaussian low-pass filtering using the linear regression model, and then combined histogram statistics and multi-fractal spectrum of the residuals with the fitness of regression model as features to distinguish natural images from rendered images. Above approaches usually achieve a well performance on the relatively simple datasets, but they are often exhibit limited performance in complex and challenging datasets.

2.2 Methods Based on Deep Learning

More recently, considering the powerful learning capacity of deep neural networks [19–21], some deep-model-based methods were proposed to solve this security problem. Rahmouni *et al* [8] developed a special pooling layer to extract the statistical quantities from the convoluted images, and this can be optimized in an "end-to-end" CNN framework to distinguish computer-generated graphics from real photographic images. Quan *et al* [10] proposed a method of adding cascaded filtering layer on the top of CNN to improve network performance. This network structure can be simply adjusted to the input of different patch sizes, and Maximal Poisson-disk Sampling (MPS) method [22] was used to assist patch augmentation process. In addition, they tried to provide insights into improving the photorealism of rendered images and synthesized images [23] via understanding the learned deep model. Yao *et al* [11] proposed an approach to separating CG images from NIs based on sensor mode noise and deep learning. For the input images, three high-pass filters (HPF) were used to remove the low-frequency signal representing image content and eliminate the interference of image content to the discrimination. He *et al* [1] combined CNN and recursive neural network (RNN) to classify CG images and NIs. They used pre-processing operations of color space transformation and Schmid filtering to extract color and texture features. A dual-path CNN was designed to combine the color and texture feature representation of each patch and conduct global modeling of local feature representation via the directed acyclic graph RNN. Nguyen *et al* [12] extended the application of the capsule network [24, 25] to identify the CG images. Recently, Bhalang Tarianga *et al* [26] proposed an attention-based deep convolutional recurrent model to classify computer-generated images and NIs.

To the best of our knowledge, there is no existing work that considers the "generality" of the CNNs (or the module of a network). Our first study in this direction can effectively improve the classification performance of existing deep models. In this work, we design a self-coding module based on the channel correlation, and then combine consecutive convolutional layers to construct a generality-well subnetwork.

3 Proposed Method

In this section, we first explain the motivation of our method and then illustrate the proposed network architecture, which involves channel correlation module, pixel correlation module, feature fusion, and final decision.

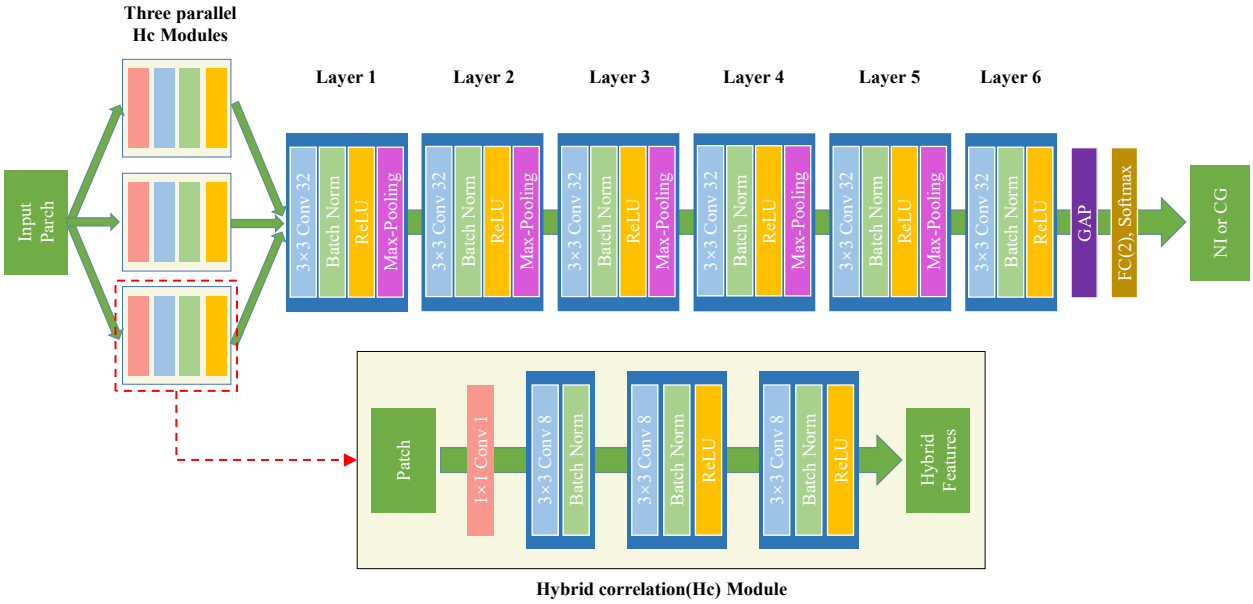

Fig.2. Architecture of our network ScNet and the hybrid correlation module. The network input is a 96×96 image patch, and output is the label (NI or CG). Each convolutional layer shows the kernel size and the number of feature maps, *e.g.*, "3×3 Conv 32" means the kernel size is 3×3 and the number of output channels is 32.

3.1 Motivation

During the process of producing photographic images, the light collected by commercial digital cameras is filtered by the Color Filter Array (CFA) before reaching the camera sensor. Each pixel of the filtered Bayer image contains only part of the spectrum [one primary color in red (R), green (G), and blue (B)], which causes two-thirds of the color information in the digital image to be lost. To obtain the complete digital color image, the sensor successively interpolates the known color in the neighborhood to estimate the missing color information of each pixel. This step leaves the correlation at the pixel and channel level. Because CG images are generated by the rendering program instead of interpolation, these two levels of correlation may be weak in CG images. Therefore, pixel neighbor correlation and channel correlation can be used as important evidence to distinguish CG images from NIs.

Convolution operation can extract pixel correlation in the window of the convolutional kernel. However, a feature map is obtained by the direct summation of multiple convolutional channels (the process of convolution operation in the CNNs), which may destroy the correlation between image color channels to some extent. Image channel correlation has attracted attention in the field of visual media security. For example, Yan *et al* [27] used differential images, *i.e.*, the differential between two color channels, to aid the identification of recolored images. Although the differential image is a representation of channel correlation, it does not necessarily depict the optimal correlation. To fully explore the correlation between R, G, and B channels, we designed a module for the convolutional neural network to automatically learn such channel correlation.

3.2 Network Architecture

In the following, we describe the structure, operation, and mechanism of each module in proposed network.

3.2.1 Channel Correlation

As shown in the bottom of Fig. 2, we designed a *self-coding* module with the 1×1 convolution of output channel 1 (the pink block) to explicitly learn channel correlation of input image patches. This self-coding channel cor-

relation module is called as *Conv1* module. The coefficient $[w_1, w_2, w_3]$ of 1×1 convolutional kernel represents the weight of R, G, and B channel in this correlation. This convolution operation can be expressed as

$$C_{ij} = w_1 \times R_{ij} + w_2 \times G_{ij} + w_3 \times B_{ij},$$

where C_{ij} is the pixel-wise output of the *Conv1* module. When the coefficient is $[1, -1, 0]$, $[1, 0, -1]$, $[0, 1, -1]$ or other special cases, it can be regarded as learning the differential image. Compared with the hard-coding operation of the differential image, *e.g.*, R-G in [27], the module designed by us learns the weight of three channels and has a larger parameter space to model channel correlation in a more flexible manner. As reported in Section 4.2, the performance of adding three *Conv1* modules is better than that of three hard-coding differential branches.

3.2.2 Pixel Correlation

After *Conv1* module, the color channel correlation of NIs and CG images is modeled. To extract the correlation between neighboring pixels, we used three 3×3 convolutions of output channel 8 without any pooling operation. Each convolution can subtly extract the correlation of 9 pixels in the local neighbor patch, and the output of the convolution is formulated as

$$O_{ij} = \sum_{u=-1}^{1} \sum_{v=-1}^{1} F(u, v) \cdot I^k(i + u, j + v),$$

where F stands for the 3×3 convolutional kernel and I^k stands for the k-th channel of the input of the convolutional layer. There is no pooling operation for all three convolutional layers, so as to retain the original useful information as much as possible. We call the combination of *Conv1* module and consecutive three convolutional layers (red dotted block in Fig. 2) as the *hybrid correlation* module (channel and pixel correlation), which is abbreviated as *Hc* module hereafter.

3.2.3 Fusion and Decision

Considering the three color channels in RGB image, we use three parallel *Hc* modules to learn the hybrid correlation of input image simultaneously (the top-left corner in Fig. 2). Note that, these three *Hc* modules are independent, and the corresponding parameters are also not shared with each other. The three *Hc* modules can extract different hybrid correlation information of the input image, and then we directly concatenate the output feature maps of *Hc* modules. Next, we use 6 convolutional layers with max-pooling to further learn the hierarchical representation (Layer 1-6 in Fig. 2), using the above concatenated feature maps as input. In addition, the number of output channels of each convolutional layer remains the same or increases by a power of 2 from 32 to 256. All max-pooling layers have the same kernel size of 3×3 and a stride of 2. We apply a global average pooling (GAP) operation to transform the final feature maps into a high-dimensional vector. Fully-connected (FC) layer and softmax layer perform final decision on vectors and obtain probabilities belonging to two classes (NI or CG).

4 Experimental Results

In this section, we validate the network design and the generality of the proposed *Hc* module by comparing it with existing corresponding counterparts. The proposed network was also compared with other representative state-of-the-art networks, *i.e.*, LiNet [10], BSP-CNN [1], YaoNet [11], and very recent attention-based model [26]. Finally, the compression robustness and generalization of these networks were evaluated.

4.1 Experimental Setup

In this work, we use the SPL2018 dataset contributed by He *et al* [1], which contains 6,800 NIs and 6,800 CG images. The advantage of this dataset is that CG images are obtained by more than 50 pieces of rendering software, while NIs are

obtained in different environments using different models of cameras. This is close to the real-world application. Following [1], the 192×192 central region of each image is cropped. Then, these cropped image patches are randomly divided into training, validation, and testing sets (with the ratio 10:3:4).

In view of computational cost and the fair comparison, all networks have the same input size of 96×96. In the training stage, we use a batch size of 64, including 32 NIs and 32 CG images. We randomly crop 96×96 patch from 192×192 image to augment training set, and shuffle the order of training set after each epoch. Stochastic Gradient Descent (SGD) [28] is used to optimize the parameters of CNN models. The initial learning rate is set as 0.001 and

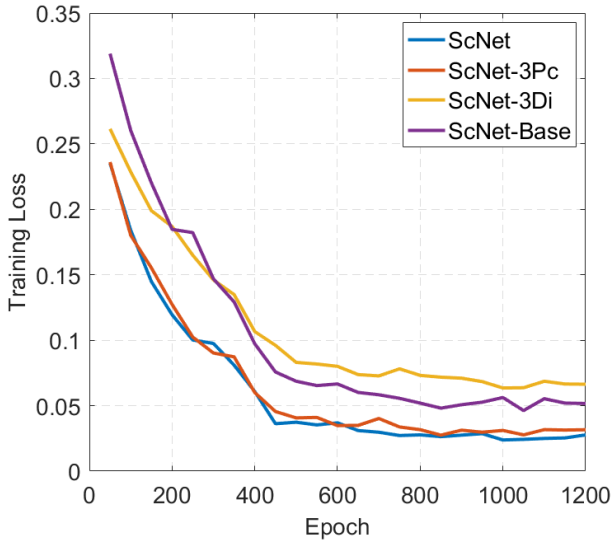

Fig.3. The training loss of the ScNet and three corresponding variants.

divided by 10 every 400 epochs. The training process stops after 1,200 epochs, and all test results are reported at 1,200 epochs. For SGD optimizer, we employ a weight decay of 0.0001 and a momentum of 0.9. In the testing stage, we follow the ten-crop average: for each test image (192×192), we crop five patches of 96×96 pixels (the center and four corner patches), flip these patches horizontally, and finally average the predictions of total 10 patches as the final result. All experiments are implemented with PyTorch 0.4.1. We train the network for 1,200 epochs, which takes about 105 minutes on a GeForce® GTX 1080Ti. In addition, all reported results are the average of three random splits.

4.2 Validation of Network Architecture Design

We evaluate our network by comparing classification accuracies of ScNet and three corresponding variants, *i.e.*, ScNet-3Pc, ScNet-3Di, and ScNet-Base. 3*Pc* stands for the variant of three parallel *Hc* modules removing *Conv1* module ("1×1 Conv 1" in Fig. 2). 3*Di* stands for the variant of three parallel *Hc* modules replacing *Conv1* module with three hard-coding differential images (R-G, R-B, and G-B) [27] respectively. *Base* stands for the architecture starting from "Layer 1" in Fig. 2. We record the training loss

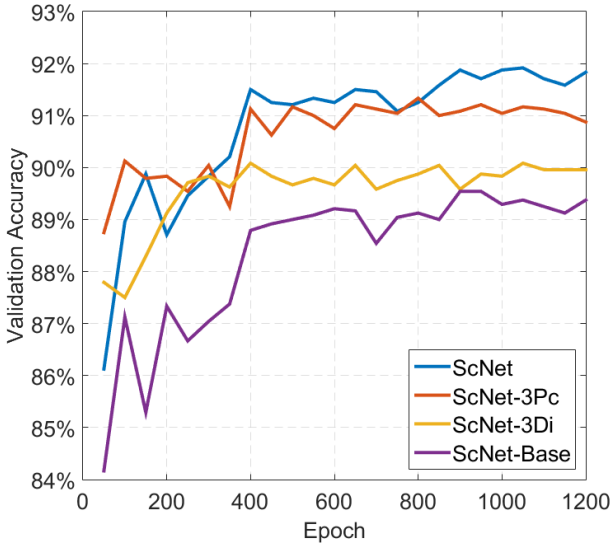

Fig.4. The accuracy rate of the ScNet and three corresponding variants on the validation set.

of above four networks for one random split and the corresponding curve is plotted in Fig. 3. The training loss quickly decreases in the first 600 epochs and the network reaches the stability after about 1,000 epochs. Fig. 4 shows the classification accuracies of these four networks on the validation set. Among four networks, ScNet achieves the best performance.

Table 1 reports the classification accuracies of ScNet, ScNet-3Pc, ScNet-3Di, and ScNet-Base on the testing set.

As shown in the last column of Table 1, ScNet is stably superior to the other three variants, such as ScNet-3Di and ScNet-Base by 2.01% and 1.72%, respectively. Compared with ScNet-3Pc, the average classification accuracy of ScNet is improved by 0.46%. This improvement indicates that our *Conv1* module can effectively capture the correlation between image color channels, and the channel correlation information is useful for the classification of NIs and CG images. In addition, we find that our self-coding strategy for exploring the correlation information between color channels is more flexible and effective than the hard-coding strategy (see the rows of "ScNet" and "ScNet-3Di" in Table 1). In fact, the former follows the philosophy of deep learning, whereas the latter follows that of feature engineering.

Table 1. Classification accuracies of ScNet and three corresponding variants. "First", "Second", and "Third" respectively corresponds to one random split. "AVG" is the average value of these three random splits. For clarity, the highest accuracy is in bold, and this is same for the remaining tables.

Architecture	First	Second	Third	AVG
ScNet	**94.72%**	**93.69%**	**94.13%**	**94.18%**
ScNet-3Pc	94.31%	93.34%	93.51%	93.72%
ScNet-3Di	92.28%	92.22%	92.00%	92.17%
ScNet-Base	93.34%	91.88%	92.16%	92.46%

4.3 Validation of Module Generality

From Table 1, we show that the *Hc* module achieves the improvement by 1.72% when comparing "AVG" of "ScNet" and "ScNet-Base". In the following, we evaluate the generality of this *Hc* module, *i.e.*, the impact of combining *Hc* module with existing CNN models. In this work, we consider three recent CNN models: LiNet [10], BSP-CNN [1], and YaoNet [11]. The corresponding results are reported in Table 2. The two variants of *3Hc* mentioned in Section 4.2, *i.e.*, 3*Pc* and 3*Di*, are also tested. In addition, "Base" in Table 2 stands for the original network designed by its authors, and the meaning of symbol "Base" is different from that mentioned in Section 4.2.

Comparing the rows of "3Hc" and "Base" in Table 2, we find that the classification accuracy of 3*Hc* (the architecture of three parallel *Hc* modules shown in Fig. 2) for three networks is higher than that of *Base* by 1.55%, 1.55%, and 4.62%, respectively. This indicates that 3*Hc* module can further improve the classification performance of existing networks through directly adding this hybrid correlation module at the beginning of existing networks. It is worth noting that the accuracy of 3*Pc* for all three networks is lower than that of 3*Hc*, which implies that *Conv1* module also has good generality, *i.e.*, explicitly extract the channel correlation information to enhance the discrimination capacity of networks. In addition, the accuracy of 3*Di* of LiNet is lower than that of LiNet by 0.36%, whereas this drop does not exist for BSP-CNN and YaoNet (comparing the rows of "3Di" and "Base" in Table 2) . This implies that the differential images (even similar hard-coding strategy) may be difficult to guarantee their generality, *i.e.*, consistently and stably improving the performance of existing CNN models.

Table 2. Generality evaluation of *Hc* module in three recent CNN models (LiNet [10], BSP-CNN [1], and YaoNet [11]). "Base" stands for the original network proposed by its authors.

	LiNet	BSP-CNN	YaoNet
3Hc	**93.93%**	**93.34%**	**93.69%**
3Pc	93.57%	93.14%	93.28%
3Di	92.02%	91.99%	91.70%
Base	92.38%	91.79%	89.07%

4.4 Comparison with State of the Art

We first compare the designed ScNet with state-of-the-art approaches. By comparing the "AVG" of "ScNet-Base" in Table 1 with the row of "*Base*" in Table 2, we find that the average classification accuracy of ScNet-Base (starting from "Layer 1" in Fig. 2) is higher than that of LiNet [10], BSP-CNN [1], and YaoNet [11], respectively. This suggests that the *Base* structure of ScNet can extract the correlation between patch pixels better than other networks. Compared with three recent networks adding modules of 3*Hc*, 3*Pc*, and 3*Di*, ScNet, ScNet-3Pc, and ScNet-3Di still show the best

Table 3. Classification accuracies of ScNet-Base, ScNet and model in [26] on RAISE vs. PRCG for four different patch sizes.

Patch Size	ScNet-Base		ScNet		Bhalang T. *et al* [26]	
	Patch	Voting	Patch	Voting	Patch	Voting
240×240	99.40%	100%	99.94%	100%	97.40%	97.20%
120×120	99.27%	100%	99.73%	100%	96.90%	97.69%
60×60	98.64%	100%	99.63%	100%	94.90%	94.55%
30×30	97.57%	100%	99.21%	100%	90.90%	92.67%

performance. For example, for 3*Hc* module, the average classification accuracy of ScNet (the "AVG" of "ScNet" in Table 1) is higher than that of LiNet, BSP-CNN, and YaoNet (the row of "3Hc" in Table 2) by 0.25%, 0.84%, and 0.49%, respectively. It indicates that ScNet is superior than the other three networks. Furthermore, the accuracy of ScNet is 0.31% higher than the best discrimination result of 93.87% in SPL2018 dataset reported by [1], which indicates that the network designed by us has superior discrimination performance. Note that, our ScNet is more simple than the network used in [1] where they combined the dual-path CNN with hand-crafted preprocessing operations and the DAG-RNN.

We also compare ScNet-Base and ScNet with the latest attention-based deep convolutional recurrent model [26]. The following experiments are conducted on the RAISE [29] versus PRCG [30], and the experiment setup are the same as those described in [26]. The corresponding results are reported in Table 3. The average classification accuracies for both patch and voting of ScNet-Base and ScNet are always higher than that of network proposed in [26]. In addition, the patch accuracy of ScNet (*i.e.*, with 3*Hc* module) is higher than that of ScNet-Base for all patch sizes (see the column "Patch" of "ScNet" and "ScNet-Base" in Table 3). This is also consistent with previous analysis that our proposed *Hc* module can further improve the classification performance.

4.5 Robustness against Post-Processing

The robustness of detection models against post-processing is important because the using of post-processing operation can weaken the statistical characteristics of CG images and NIs, so as to deceive the detectors. Here, we mainly examine the impact of JPEG compression on the identification of CG images, with compression factors from 95 to 35 with a step of 10. In the testing set, both natural samples and computer-generated samples are compressed. There is no obvious visual difference between the compressed and uncompressed images. All the following experiments use the trained models in Section 4.2 and Section 4.3, and there is no additional training for compressed images.

Table 4 shows the statistics of the robustness test of the 3*Hc* structure and *Base* structure in the four networks (ScNet, LiNet, BSP-CNN, and YaoNet) to JPEG compression with seven different quality factors. For each of the seven quality factors (each row shown in Table 4), it is always ScNet, with 3*Hc* module, that has the highest accuracy expect for two cases (87.18% for LiNet-3Hc, and 87.12% for BSP-CNN-3Hc in row of "85"). Note that, the corresponding accuracy for ScNet-3Hc is 87.07%, and it is very close to the above two values. In addition, the accuracy of network with 3*Hc* is better than that of network without 3*Hc*. Take the ScNet as an example, the values in column of "3Hc" is higher than that in column of "Base". When images under the strong compression (quality factor starting from 75 to 35), the declining speed of accuracy of network with 3*Hc* is slower than that of the network without 3*Hc*. For example, the gap is 2.70% for BSP-CNN-3Hc (from 80.66% to 77.96%), while 5.33% for BSP-CNN-Base (from 81.83% to 76.50%). This observation further confirms that our proposed 3*Hc* module can enhance the robustness of original

Table 4. Performance evaluation of four networks with/without 3*Hc* module on the JPEG compressed testing set. "-" means results on the original testing set. Note that all models are trained on original training set.

Quality Factor	ScNet		LiNet		BSP-CNN		YaoNet	
	3Hc	Base	3Hc	Base	3Hc	Base	3Hc	Base
-	**94.18%**	92.46%	93.93%	92.38%	93.34%	91.79%	93.69%	89.07%
95	**93.26%**	91.43%	92.43%	91.74%	92.06%	91.11%	91.70%	87.59%
85	87.07%	86.73%	**87.18%**	86.62%	87.12%	86.67%	86.48%	82.20%
75	**82.36%**	82.25%	81.97%	79.83%	80.66%	81.83%	82.01%	79.96%
65	**81.94%**	78.62%	77.92%	75.50%	80.35%	78.53%	80.53%	77.24%
55	**80.08%**	76.83%	76.67%	72.97%	78.47%	76.64%	77.79%	74.75%
45	**79.95%**	76.43%	76.75%	71.85%	78.39%	76.13%	77.54%	73.44%
35	**79.44%**	76.39%	76.39%	71.82%	77.96%	76.50%	77.81%	72.27%

Table 5. Generalization performance evaluation of four networks with/without 3*Hc* module on the Google vs. PRCG. "First", "Second", and "Third" shows the classification accuracies of networks trained on SPL2018 dataset with one random split, respectively. "AVG" is the average value of these three random splits.

Architecture	ScNet		LiNet		BSP-CNN		YaoNet	
	3Hc	Base	3Hc	Base	3Hc	Base	3Hc	Base
First	**82.81%**	79.38%	78.31%	76.94%	82.19%	77.75%	74.50%	74.62%
Second	**82.44%**	77.88%	76.31%	76.50%	81.13%	76.69%	73.75%	72.69%
Third	**81.69%**	79.25%	78.06%	76.38%	81.00%	77.13%	74.50%	74.06%
AVG	**82.31%**	78.83%	77.56%	76.61%	81.44%	77.19%	74.25%	73.79%

networks for JPEG compression.

4.6 Generalization Evaluation

To apply a CNN-based detector to the real-world scenario, the generalization performance, *i.e.*, testing on "unseen" dataset, is an important factor. We evaluate the generalization capability of proposed method on the highly challenging dataset of Google versus PRCG [2], which comprises NIs and CG images of heterogeneous origins and is thus close to the real-world application. All the following experiments use the trained models in Section 4.2 and Section 4.3, and these models use the training dataset of SPL2018 [1].

Table 5 reports the classification accuracies of the 3*Hc* structure and *Base* structure in the four networks (ScNet, LiNet, BSP-CNN, and YaoNet) on Google versus PRCG dataset. Comparing the column of "Base" of all four networks, we find that ScNet has the highest accuracy. Note that, for LiNet, BSP-CNN and YaoNet, "Base" refers to the network architecture reported in their papers [1, 10, 11], therefore, this illustrates that the *Base* structure of ScNet (starting from "Layer 1" in Fig. 2) has better generalization performance. When comparing the average accuracies between "Base" and "3Hc" for four networks (the row of "AVG" in Table 5), the accuracy increases by 3.48%, 0.95%, 4.25%, and 0.46%, respectively. This improvement shows that adding the 3*Hc* module can enhance the network generalization capability. Furthermore, the classification accuracy of ScNet-3Hc is the highest among all of validation networks, and this means that our proposed network holds the superior generalization capability.

5 Visualization and Understanding

To better explain the working principle of *Conv1* self-coding module, we visualize the coefficient of 1×1 convolutional kernel in *Conv1* module and the feature map after *Conv1* encoding.

Fig. 5 shows the color mapping image of weights of each 1×1 convolutional kernel of ScNet trained on three random splits. We find that the weights of the three *Conv1* modules of each random split are roughly arranged in (+, 0, -), with

no fixed order, which is similar to the idea of the input differential image. In addition, we find that the absolute value of the three convolutional kernel weights is in three orders of magnitude, which are the large value (positive mapping to red, negative mapping to blue), the median value (positive mapping to orange, negative mapping to cyan) and the small value (positive and negative mapping to green). For example, in the first random split, the weight of the first *Conv1* (Conv1-1) is in a small magnitude and the weight color mapping is almost green. The weight of the second *Conv1* (Conv1-2) is in a medium magnitude and the non-zero weight color mapping is orange and cyan. The weight of the third *Conv1* (Conv1-3) is in a large magnitude and the non-zero weight color mapping is red and blue. Comparing the *Conv1* of three random splits in Fig. 5, the *Conv1* convolutional kernel in the first random split best conforms to the above description, and ScNet trained on the first random

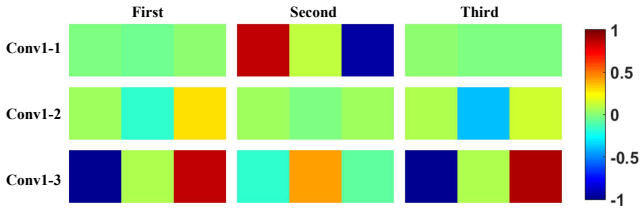

Fig.5. The color mapping of the weights of three parallel *Conv1* modules ("Conv1-1", "Conv1-2" and "Conv1-3") for three random splits ("First", "Second" and "Third").

split achieves the best results (94.72% in the row of "ScNet" in Table 1). The indirect difference of image color channels coding in three orders of magnitude can extract richer features between channels, which enable CNN to learn features better.

Fig. 6 visualizes the feature maps after the *Conv1* in the first random split. The first row is a natural image, and the second row is a computer-generated image. Each column from left to right is the original input image and the respective feature map of three *Conv1* modules with the weights in small, medium and large magnitude. For the NI, the words (red block of the first row of Fig. 6) becomes more and more obscure from left to right. However, the words in the CG image (red block of the second row of Fig. 6) are sharp for three feature maps, which is similar to [31]. The work of [31] has shown that high-frequency components across NI color channels are strongly correlated and similar. The feature maps of NI are harmonious and smooth, showing good channel correlation. The feature maps of CG image, *e.g.*, paint and words on the wall, are abrupt and show poor correlation. By automatically learning the correlation between color channels, the distance between NI and CG image in the feature domain is increased, and thus the network identification result is improved.

6 Conclusion

In this paper, channel and pixel correlation were used to model the differences between NIs and CG images in terms of statistical characteristics. We proposed a self-coding

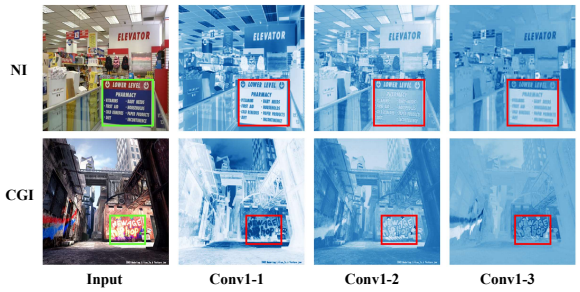

Fig.6. Feature maps on 1×1 convolutional layer of three parallel *Conv1* modules in the first random split.

module to extract features between image color channels and designed a CNN to better extract features between image pixels. We conducted a number of experiments to evaluate the complete framework. Compared with other advanced technologies, our framework obtained better detection performance. More importantly, the self-coding module with consecutive convolutional layers constructs a hybrid correlation module, which can be directly combined with existing CNN models, and this can further enhance their discrimination capacity. The source code of our method is attached and will be publicly released with final version. In future work,

we would like to apply this novel framework to other multimedia security tasks, *e.g.*, recolored image detection and image manipulation detection.

Acknowledgment

We would like to thank Dr. He for kindly sharing the SPL2018 dataset [1], Dr. Dang-Nguyen for the RAISE dataset [29], and Dr. Ng for the Google and PRCG dataset [30].

This work was supported by the National Key R&D Program of China (2019YFB2204104), the Beijing Natural Science Foundation (L182059), the National Natural Science Foundation of China (61772523, 61620106003, and 61802406), Alibaba Group through Alibaba Innovative Research Program, and the Joint Open Research Fund Program of State key Laboratory of Hydroscience and Engineering and Tsinghua-Ningxia Yinchuan Joint Institute of Internet of Waters on Digital Water Governance.

References

[1] He P, Jiang X, Sun T, Li H. Computer graphics identification combining convolutional and recurrent neural networks. *IEEE Signal Processing Letters*, 2018, 25(9):1369–1373.

[2] Ng T T, Chang S F, Hsu J, Xie L, Tsui M P. Physics-motivated features for distinguishing photographic images and computer graphics. In *Proceedings of the ACM International Conference on Multimedia*, 2005, pp. 239–248.

[3] Lyu S, Farid H. How realistic is photorealistic? *IEEE Transactions on Signal Processing*, 2005, 53(2):845–850.

[4] Chen W, Shi Y Q, Xuan G. Identifying computer graphics using HSV color model and statistical moments of characteristic functions. In *Proceedings of the IEEE International Conference on Multimedia & Expo*, 2007, pp. 1123–1126.

[5] Gallagher A C, Chen T. Image authentication by detecting traces of demosaicing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[6] Sankar G, Zhao V, Yang Y H. Feature based classification of computer graphics and real images. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1513–1516.

[7] Peng F, Zhou D l, Long M, Sun X M. Discrimination of natural images and computer generated graphics based on multi-fractal and regression analysis. *AEU-International Journal of Electronics and Communications*, 2017, 71:72–81.

[8] Rahmouni N, Nozick V, Yamagishi J, Echizen I. Distinguishing computer graphics from natural images using convolution neural networks. In *Proceedings of the IEEE International Workshop on Information Forensics and Security*, 2017, pp. 1–6.

[9] Yu I, Kim D, Park J, Hou J, Choi S, Lee H. Identifying photorealistic computer graphics using convolutional neural networks. In *Proceedings of the IEEE International Conference on Image Processing*, 2017, pp. 4093–4097.

[10] Quan W, Wang K, Yan D M, Zhang X. Distinguishing between natural and computer-generated images using convolutional neural networks. *IEEE Transactions on Information Forensics and Security*, 2018, 13(11):2772–2787.

[11] Yao Y, Hu W, Zhang W, Wu T, Shi Y Q. Distinguishing computer-generated graphics from natural images based on sensor pattern noise and deep learning. *Sensors*, 2018, 18(4):1296.

[12] Nguyen H H, Yamagishi J, Echizen I. Capsule-forensics: Using capsule networks to detect forged images and videos. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 2307–2311.

[13] Popescu A C, Farid H. Exposing digital forgeries by detecting traces of resampling. *IEEE Transactions on Signal Processing*, 2005, 53(2):758–767.

[14] Ianeva T I, de Vries A P, Rohrig H. Detecting cartoons: a case study in automatic video-genre classification. In *Proceedings of the IEEE International Conference on Multimedia & Expo*, volume 1, 2003, pp. 449–452.

[15] Pan F, Chen J, Huang J. Discriminating between photorealistic computer graphics and natural images using fractal geometry. *Science in China Series F: Information Sciences*, 2009, 52(2):329–337.

[16] Zhang R, Wang R D, Ng T T. Distinguishing photographic images and photorealistic computer graphics using visual vocabulary on local image edges. In *Proceedings of the International Workshop on Digital-Forensics and Watermarking*, 2012, pp. 292–305.

[17] Li Z, Ye J, Shi Y Q. Distinguishing computer graphics from photographic images using local binary patterns. In *Proceedings of the International Workshop on Digital-Forensics and Watermarking*, 2013, pp. 228–241.

[18] Ojala T, Pietikainen M, Maenpaa T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, 24(7):971–987.

[19] LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*, 2015, 521(7553):436–444.

[20] Zhang X J, Lu Y F, Zhang S H. Multi-task learning for food identification and analysis with deep convolutional neural networks. *Journal of Computer Science and Technology*, 2016, 31(3):489–500.

[21] Cheng M M, Hou Q B, Zhang S H, Rosin P L. Intelligent visual media processing: When graphics meets vision. *Journal of Computer Science and Technology*, 2017, 32(1):110–121.

[22] Quan W, Yan D M, Guo J, Meng W, Zhang X. Maximal poisson-disk sampling via sampling radius optimization. In *Proceedings of the SIGGRAPH ASIA Posters*, 2016, pp. 22:1–22:2.

[23] Barnes C, Zhang F L. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media*, 2017, 3(1):3–20.

[24] Hinton G E, Krizhevsky A, Wang S D. Transforming auto-encoders. In *Proceedings of the International Conference on Artificial Neural Networks*, 2011, pp. 44–51.

[25] Sabour S, Frosst N, Hinton G E. Dynamic routing between capsules. In *Proceedings of the Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[26] Bhalang Tarianga D, Senguptab P, Roy A, Subhra Chakraborty R, Naskar R. Classification of computer generated and natural images based on efficient deep convolutional recurrent attention model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 146–152.

[27] Yan Y, Ren W, Cao X. Recolored image detection via a deep discriminative model. *IEEE Transactions on Information Forensics and Security*, 2019, 14(1):5–17.

[28] Bottou L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of International Conference on Computational Statistics*, pp. 177–186. 2010.

[29] Dang-Nguyen D T, Pasquini C, Conotter V, Boato G. Raise: A raw images dataset for digital image forensics. In *Proceedings of the ACM Multimedia Systems Conference*, 2015, pp. 219–224.

[30] Ng T T, Chang S F, Hsu J, Pepeljugoski M. Columbia photographic images and photorealistic computer graphics dataset. Technical Report 205-2004-5, ADVENT, Columbia University, 2004.

[31] Gunturk B K, Altunbasak Y, Mersereau R M. Color plane interpolation using alternating projections. *IEEE Transactions on Image Processing*, 2002, 11(9):997–1013.