World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:12, No:10, 2018

# Classification of Computer Generated Images from Photographic Images Using Convolutional Neural Networks

Chaitanya Chawla, Divya Panwar, Gurneesh Singh Anand, M. P. S Bhatia

***Abstract*—**This paper presents a deep-learning mechanism for classifying computer generated images and photographic images. The proposed method accounts for a convolutional layer capable of automatically learning correlation between neighbouring pixels. In the current form, Convolutional Neural Network (CNN) will learn features based on an image's content instead of the structural features of the image. The layer is particularly designed to subdue an image's content and robustly learn the sensor pattern noise features (usually inherited from image processing in a camera) as well as the statistical properties of images. The paper was assessed on latest natural and computer generated images, and it was concluded that it performs better than the current state of the art methods.

***Keywords*—**Image forensics, computer graphics, classification, deep learning, convolutional neural networks.

## I. Introduction

IN recent years, with the developments in computer software, generating realistic computer generated images (RCG) has become a much easier and achievable task. This software render images which are very tough to differentiate from natural images by the human eye. While their emergence has revolutionised the multimedia industry providing ability to create realistic animations and images conveniently, they could as well pose a grave security threat to the public if used in fields such as law, authority, and journalism [16]. Hence, distinguishing CG from NI has become an important topic in the field of image forensics [2], attracting a lot of attention in the past decade.

Deep learning has achieved great success, in recent times, showcasing its effectiveness in many new fields. The CNN, amongst other deep neural networks, have the capability to obtain higher order features automatically and efficiently decrease its complexity and dimensionality [6].

While CNNs have played a major role in tasks such as object classification and image segmentation, where the networks were expected to learn features based on image content, their contribution in problems demanding learning of features based on image statistics automatically has been limited. Reference [1] solved this problem by proposing a special convolutional layer and used it in the detection of image manipulation techniques such as Gaussian blurring and Median filtering. They suggest that certain regional relationships prevail between pixels which are independent of the content of the image. Motivated by this observation, the paper uses the special layer as a primary filter and puts forward a deep convolutional network architecture for approaching the problem of classifying CG and NI. This layer allows the CNN to learn certain statistical properties such as residual signals and sensor pattern noise added to the image on processing by digital cameras.

Similar to the approach in [3], the networks in the paper are trained on image patches of a smaller size than the image. While breaking down a huge image into small patches indeed leads to a huge loss in statistical information and hence the features learned, it makes the proposed approach feasible for detection of regional splicing and also provides for a much lower execution time. Further, after training modified networks on various image patch sizes, it was observed that the results tend to saturate with the increase of sizes. Hence, it illustrates that, despite dividing the image into smaller and computationally feasible patches, the proposed network successfully extracts the statistical correlation of pixels in individual patches to predict the outcome of the original image. Further, the proposed method was observed to work and make high accuracy predictions on images of varying JPEG compression quality factors exhibiting the robustness of the trained model.

The paper is formulated as follows: Section II contains the related work, Section III explains the new convolutional layer, while Section IV describes the overall architecture of the proposed network. Sections V and VI provide the details of the dataset, results and comparisons with state of the art methods.

## II. Related Work

Classifying and distinguishing PG from CG is firmly related to the performance of computer graphics and rendering software in developing realistic images by nature. [10] is one of the earlier papers to highlight the distinction of CG from PG images primarily detected in the image smoothness due to triangles.

In recent times, some researchers have successfully utilized deep learning to find solutions to problems concerned with image forensics, such as image manipulation detection [1], where Bayar et al. focus on extracting features based on statistical correlation between pixels of the image. Researches have also utilized deep learning in camera model identification [14], [4], steganalysis [5], copy move forgery detection [2].

Chaitanya Chawla is with the Netaji Subhas Institute of Technology, India (e-mail: chaitanyachawla1996@gmail.com).

Fig. 1 Some examples of CGI from dataset in [7]

Fig. 2 Some examples of NI from dataset in [8]

Farid and Lyu [9] applied High-order wavelet statistics in digital forensics. They presented a method to detect steganography and proposed an approach applying it to distinguish CG from PG. In this and later works, the image is decomposed into a “wavelet-like” structure, to generate a vector containing features, and these features are used by machine learning algorithms for classification.

Dirik et al. [11] suggested that the difference between CG and PG images mainly lies in the sensor noise pattern of the image, although [12] speculated that the edges of an image are more essential in this issue. [13] used histograms on wavelet filtered data before its classification.

A fine-tuned CNN inspired approach was presented by Gando et al. [15]. This work outperformed other models, inclusive of all the custom CNN-based models and traditionally applied models which used precalculated features. Their model is capable of automatically distinguishing illustrations from photographs.

A custom pooling layer was presented by Rahmouni et al. [3] to excerpt sensor pattern noise features and a CNN to distinguish lifelike computer-generated graphics from photographic images. They further used a scheme based on weighted voting to anticipate the label of the whole image by aggregating the local estimates of class probabilities.

## III. Convolutional Layer

As suggested in [1], the crucial concept behind using the modified convolutional layer is that, certain regional structural relations lie amidst image pixels and are not dependent on the image’s content. The correlations between pixels in photographic images differ from those amongst images generated by computer. As a result, the classifier should determine the relationship between a pixel and its surroundings while simultaneously concealing image’s content. In order to ensure this, some prediction error filters have to be forced on to the first convolutional layer.

Prediction error filters are filters that foresee the value of the pixel at the center of the filter. This value is then subtracted to yield the prediction error. All of the K filters $w_k^{(1)}$ in the first layer of the CNN have the following restraints placed on them:

$$w_k^{(1)}(0,0) = -1 \tag{1}$$

$$\sum_{l,m\neq 0} \quad w_k^{(1)}(l,m) = 1 \tag{2}$$

where $w_k^{(1)}(l,m)$ is the filter weight at the (l,m) position and $w_k^{(1)}(l,m)$ is the filter weight at the center of the filter window.

```
1.  Initialize  using randomly drawn weights
2.   i=1
3.  while i ≤ max iter do
4.          Set $w_k^{(1)}(0,0) = 0$ for all K filters
5.          Normalize $w_k^{(1)}$'s such that $\sum_{l,m\neq 0} w_k^{(1)}(l,m) = 1$
6.          Set $w_k^{(1)}(0,0) = -1$ for all K filters
7.          Do feedforward pass
8.          Update filter weights through stochastic gradient descent and backpropagate errors
9.          i = i+1
10.       if training accuracy converges then exit
11. end
12. Enforce constraints on  given in Steps 4 through 6
```

Fig. 3 Algorithm for the convolutional layer

The algorithm for the new convolutional layer as suggested in [1] is given in Fig. 3. Each filter in the layer is initialized by randomly choosing each filter weight. Then, before the start of each gradient descent iteration, it is ensured that the constraints in (1) and (2) are enforced.

## IV. Proposed Method

As proposed in [3], the input images are first divided into N patches each of dimension 100 × 100. The particular size is chosen as a tradeoff; if too large a size increases training time of the classifier, while too small a size leads to loss of statistical information. The patches obtained undergo an image filtering step, and the outputs are probabilities of the patch belonging to CG or NI category. In the next step, the image is classified by aggregating the results of all the patches contained in the image using two different types of aggregation functions.

### *A. Patch Classification*

This section presents the proposed CNN architecture (addressed as SL+5 throughout the paper) for performing classification between CG and PG image. Fig. 4 shows the suggested CNN architecture and some precise information about the dimensions of all the layers. The network has eight layers, firstly the suggested new convolution layer, five sets of typical convolutional layers to which batch normalization and average pooling operations are applied. They are followed by two fully connected layers. An input layer is used to feed images into the CNN, known as the data layer. The input to the first layer is a 100×100 image.

The first layer in the network is the convolutional layer discussed in Section III. The later sections explain the detailed steps of the overall architecture of the network following the new convolutional layer.

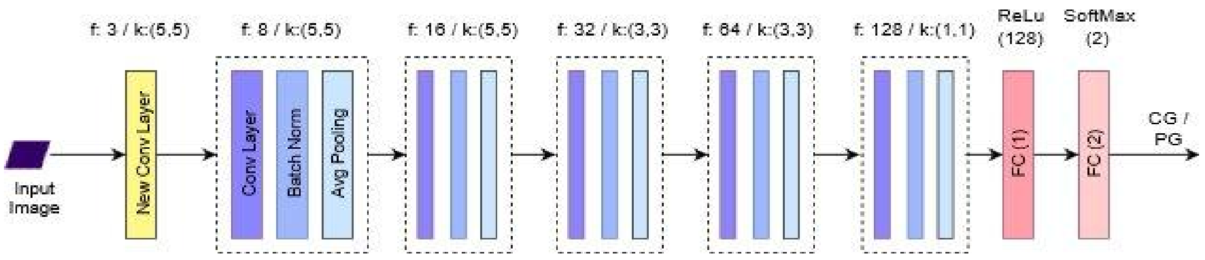

Fig. 4 Proposed Network Architecture of SL+5, f and k denote the number of filters and kernel size respectively

### B. Typical Convolutional Layer

Let w*h*c be the input image to a regular convolutional layer (where w is the width, h is the height and c is the number of feature maps or alternatively number of channels in the input image). Further let the layer consist of k kernel filters of size a*a. The filter is slid over all the a×a regions of the input image and convolved with it with an overlying distance known as stride, to produce the output image.

A convolutional layer is regularly succeeded by the application of a nonlinear function enforced in an activation layer. This nonlinear function is applied on to each pixel value. A Rectified Linear Unit (ReLU), i.e., function(x) = maximum(0,x) [18] was used. It has been demonstrated that the training time for a CNN with ReLU function as a neural activation is subsequently less in real life [17].

### C. Pooling and Batch Normalization Layers

After every typical convolutional layer, an overlapping average-pooling layer was applied. The main task of a pooling layer is to cut down the special size of the data flowing through the network and lower the resolution of the image and make it robust to learn high level features. The functioning of the pooling layer is such that it computes the average value in every locality at various positions. This task is done by using a kernel size of 2 and a stride of 1. Following the average-pooling layers is Batch Normalization which reduces the amount by which the hidden unit values shift around (covariance shift). It also allows each layer of a network to learn by itself, independent of other layers.

### D. Fully-Connected Layers

The network consists of two fully connected layers, the first one having 128 neurons with ReLu activation followed by a class determining fully connected layer having two neurons (one per class) with SoftMax activation. The input to the first Fully-Connected Layer is the output of the previous layer flattened.

The network aims to minimize the binary cross-entropy loss function commonly used in classification problems and uses Adam's algorithm for training the network.

### E. Output Aggregation

To predict the outcome of the complete image from the classification probabilities of image patches obtained from the network above, an output aggregation function is used. This successfully incorporates the statistical correlation of pixels in individual patches to predict the outcome of the original image.

For this, two methods are used: - a weighted voting scheme where each patch of the input image contributes its difference of the log of probability of the label [3] and a majority voting scheme where the label occurring in the majority of patches is assumed as the label for full size image.

## V. Dataset

The proposed deep learning based approach was compared with the existing state of the art methods in [3]. The same dataset of photographic and computer generated images used in [3] is deployed here. It consists of 1800 computer-generated graphics and 1800 natural images.

The computer-generated graphics were taken from the Level Design Reference Database [7], which consists around 55,000 screenshots of video games. Game toolbars, Subtitles and other non-real elements were cropped out from the images.

The photographic images are obtained from the RAISE dataset [8] which consists of an ample variety of outdoor and indoor scenarios such as faces, people, countrysides and different man-made items (e.g. buildings, motor vehicles), etc. of varying dimensions. All of these natural images were obtained in NEF data format.

TABLE I
Architectures for Patch Classification

| Name | SL | Conv1 | Conv2 | Conv3 | Conv4 | Conv5 | FC1 | FC2 |
|---|---|---|---|---|---|---|---|---|
| SL+5 | f:3 k:(5,5) | f:8 k:(5,5) | f:16 k:(5,5) | f:32 k:(3,3) | f:64 k:(3,3) | f:128 k:(1,1) | 128 | 2 |
| SL+4 | f:3 k:(5,5) | f:8 k:(5,5) | f:16 k:(5,5) | f:32 k:(3,3) | f:64 k:(1,1) | NA | 64 | 2 |
| SL+3 | f:3 k:(5,5) | f:8 k:(5,5) | f:16 k:(5,5) | f:32 k:(3,3) | NA | NA | 64 | 2 |
| SL+2 | f:3 k:(5,5) | f:8 k:(5,5) | f:16 k:(3,3) | NA | NA | NA | 64 | 2 |

## VI. Experimental Setup and Results

The 3600 RAW images files in the dataset were compressed to JPEG images of different Quality Factors (78, 85, and 95) to form three datasets. Three different databases using these 3600 images were made and different tests were carried out on them. All classes were split into training, testing and validation as 70%, 20%, and 10% of the dataset, forming the Full-size database. Finally, 55,000 patches sized at 100 × 100 were extracted for training, 15000 patches for testing and 7000 patches for validating the patch classifier.

Implementation of all our networks was done on Keras deep learning framework with Tensorflow backend. The experiments were run on Tesla K80 GPU with 61 GB RAM. For training, Adam's algorithm was used tuned a learning rate of $10^{-4}$ and momentum exponential decay rates $\beta1 = 0.9$ and $\beta2 = 0.99$. The batch size for training and testing was set to 32 images.

TABLE II
Patch Accuracy

| Name | Patch Accuracy |
|---|---|
| SL+5 | 90.23 |
| SL+4 | 87 |
| SL+3 | 85.2 |
| SL+2 | 82.5 |

TABLE III
Accuracy's Variation with Quality Factor

| Quality Factor | Patch-wise Accuracy | Full image Accuracy (MV) |
|---|---|---|
| 75 | 89.90 | 99.02 |
| 85 | 90.05 | 99.23 |
| 95 | 90.23 | 99.50 |

TABLE IV
Full-Size Image Classification Accuracy Comparison

| Name | Weighted Voting | Majority Voting |
|---|---|---|
| Rahmouni et al. [3] | 93.2 | 91.7 |
| SL+5 | 99.5 | 100 |

### A. Patch Classification

The different architectures experimented on are shown in Table I (SL+5 denotes architecture with Special Layer followed by five convolutional layers and so on) where f and k denote the number of filters and kernel size respectively. These architectures differ by the number of convolution layers following the special layer as well as the kernel sizes and number of filters per layer. The database with JPEG images of Quality factor 95 was used to train and test all the four architectures introduced in the experiment.

It was observed that three of the four architectures fared better in patch classification than the state of the art method, and SL+5 showed a steep increase of 5.43% in patch wise accuracy proving to be the best suited of the proposed architectures. These results are recorded in Table II. The validation accuracy (for patches) and training loss observed for the four architectures and the state of the art method are shown in Fig. 5 and Fig. 6, respectively.

### B. JPEG Dataset of Different Quality Factors

To further evaluate the model SL+5, it was trained and tested on the three databases of quality factor 75, 85, and 95. The Patch-wise accuracy and full-size image accuracy, computed using Majority Voting (MV), were evaluated and are listed in Table III. Minimal variation was observed in the accuracy with respect to the quality factor. The images with highest resolution, i.e. quality factor 95 showed the best accuracy as expected.

### C. Full-Size Image Classification

For a fair comparison (in terms of classifying the full-size images), between the best suited architecture SL+5 and the state of the art method, the model in [3] was also evaluated, with both majority voting scheme and weighted voting scheme. The results are recorded in Table IV. It was observed that the proposed model outperformed [3] under both schemes and gave an accuracy of 100% using Majority Voting Scheme.

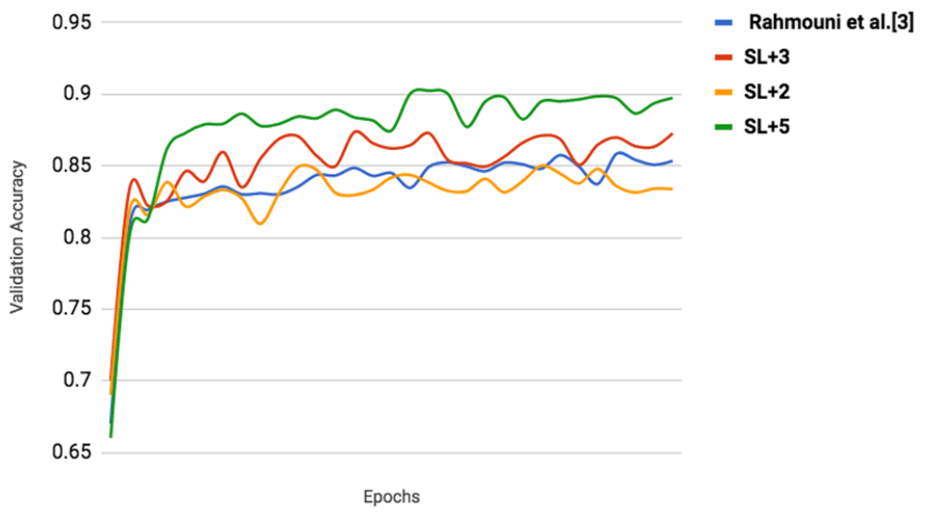

Fig. 5 Validation accuracy

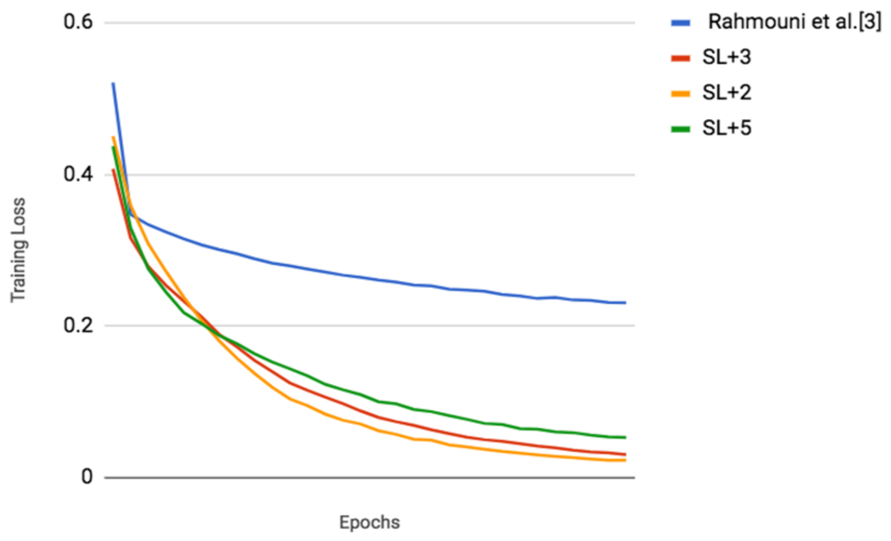

Fig. 6 Training loss during training of various models

## VII. Conclusion

This paper incorporates the idea of introducing a special convolution layer that focuses on correlation between pixels in the problem of distinguishing computer generated and natural images. To challenge the proposed algorithm with the most naturalistic computer graphic images, screenshots from lifelike video-games were used. 100% accuracy was observed in classifying full size images, and this is an improvement over many state of the art methods in classifying patches of images.

## References

[1] Bayar, B., Stamm, M.: A Deep Learning Approach to Universal Image Manipulation Detection Using A New Convolutional Layer.

[2] Rao, Y.; Ni, J.: A deep learning approach to detection of splicing and copy-move forgeries in images. In Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS), Abu Dhabi, UAE, 4–7 December 2016; 1-6.

[3] 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 (WIFS), Rennes, France, December 4-7, 2017; 1-6.

[4] Tuama, A., Comby, F., Chaumont, M.: Camera model identification with the use of deep convolutional neural networks. In Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS), Abu Dhabi, UAE, 4–7 December 2016; 1–6

[5] Xu, G., Wu, H.Z., Shi, Y.Q.: Structural Design of Convolutional Neural Networks for Steganalysis. IEEE Signal Processing Letters 2016, 23, 708–712.

[6] Yao, Y., Shi, Y., Weng, S.: Guan, B. Deep Learning for Detection of Object-Based Forgery in Advanced Video. Symmetry 2018, 10, 1-10.

[7] Piaskiewicz, M.: Level-design reference database. (Online). Available: http://level-design.org/referencedb/.

[8] Dang-Nguyen, D.-T. Pasquini, C., Conotter, V. and Boato, G.:Raise: a raw images dataset for digital image forensics, in Proceedings of the 6th ACM Multimedia Systems Conference. ACM, 2015, pp. 219–224.

[9] Farid H. and Lyu, Su.: Higher-order wavelet statistics and their application to digital forensics, in IEEE Workshop on Statistical Analysis in Computer Vision, Madison, Wisconsin, 2003, p. 94.

[10] Ng T.-T and Chang, S.-F: An online system for classifying computer graphics images from natural photographs, in Electronic Imaging 2006. International Society for Optics and Photonics, 2006, pp. 607 211– 607 211.

[11] Dirik, A. E, Bayram, S., Sencar, H. T and Memon N.: New features to identify computer generated images, in IEEE International Conference on Image Processing, ICIP 2007, vol. 4. IEEE, 2007, pp. IV–433.

[12] Wang, R. Fan, S. and. Zhang, Y.: Classifying computer generated graphics and natural imaged based on image contour information, J. Inf. Comput.

[13] Wang, Y. and Moulin, P.: On discrimination between photorealistic and photographic images, in Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, vol. 2. IEEE, 2006, pp. II–II.

[14] Bondi, L., Baroffio, L., G¨uera, D., Bestagini, P. Delp, E. J.: Tubaro, S. First Steps Toward Camera Model Identification With Convolutional Neural Networks. IEEE Signal Processing Letters 2017, 24, 259–263.

[15] Gando, G., Yamada, T., Sato, H., Oyama, S.: Kurihara, M. Fine-tuning deep convolutional neural networks for distinguishing illustrations from photographs. Expert Systems with Applications 2016, 66, 295–301.

[16] Rocha, A., Scheirer, W., Boult, T., Goldenstein, S.: Vision of the unseen Current trends and challenges in digital image and video forensics. ACM Computing Surveys 2011, 43, 26–40.

[17] Krizhevsky, A, Sutskever, I, and Hinton, G. E: Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[18] Nair, V, Hinton, G. E.: Rectified linear units improve restricted boltzmann machines. In International Conference on Machine Learning, pages 807–814, 2010.