Distinguishing Computer Graphics from Natural Images Using Convolution Neural Networks

Nicolas Rahmouni
Ecole Polytechnique
Palaiseau, France
Email: nicolas.rahmouni@polytechnique.edu

Vincent Nozick
JFLI (UMI 3527)
Tokyo, Japan
Email: vincent.nozick@u-pem.fr

Junichi Yamagishi*, Isao Echizen†
National Intitute of Informatics
Email: *jyamagis@nii.ac.jp,
†iechizen@nii.ac.jp

***Abstract*—This paper presents a deep-learning method for distinguishing computer generated graphics from real photographic images. The proposed method uses a Convolutional Neural Network (CNN) with a custom pooling layer to optimize current best-performing algorithms feature extraction scheme. Local estimates of class probabilities are computed and aggregated to predict the label of the whole picture. We evaluate our work on recent photo-realistic computer graphics and show that it outperforms state of the art methods for both local and full image classification.**

I. Introduction

Digital image forensics is a research field mostly dedicated to the detection of image falsification. A large part of the current research focuses on the detection of image splicing, copy-past or camera identification [1]. This paper addresses the problem of identifying whether an image is a computer graphics or a natural photograph.

A. Computer Graphics vs Natural Photographs

Some recent advances in image processing, like the real-time facial reenactment face2face [2], show the importance of having some tools to distinguish computer graphics (CG) from natural photographic images (PG). Although the distinction between CG and PG depends not only on image properties, but also on cognitive characteristics of viewers [3], people show inaccurate skills at differentiating between altered and non-altered images [4].

B. State of the art

In 2003, Farid and Lyu [5] presented a method to detect steganography and suggested to apply it to distinguish CG from PG. Like the quasi-totality of the methods that followed, the authors perform a "wavelet-like" decomposition of the images to generate some features vectors, combined with machine learning for classification.

Distinguishing CG from PG is by nature strongly related to computer graphics performances in generating photo-realistic images. Ng et al. [6] is the first paper to mention the expected difference between CG and PG images, mainly consisting in the image smoothness due to triangles. Dirik et al. [7] consider that this difference better resides in the statistical noise properties of the image. Wang et al. [8] consider that the image edges are more relevant for this problem.

Wang and Moulin [9] introduce the use of histograms on the wavelet filtered data, before classification. This approach will be followed by many similar methods, like [10] and [11]. Among these methods, Wu et al. [12] is clearly the easiest to implement. Li et al. [13] recommend to perform the computation in HSV color space.

Alternatives to the "wavelet-like" decomposition are to work directly on the noise image, as proposed by Khanna et al. [14], or to detect traces of demosaicing that should not appear in CG images, as proposed by Gallagher and Chen [15]. Note that this last method will handle only images at native resolution since scaling an image will strongly alter the demosaicing artifacts.

For more detailed overviews, the reader can refer to Tokuda et al. [16], as well as to Wang et al. [8] and Ng and Chang [17].

In the rest of the paper, we will mainly compare our method to Wu et al. [12], i.e. an histogram based method derived from [13], known as one of the most effective from the current state of the art.

C. Motivations

Statistical properties of filtered images are good discriminators for distinguishing CG from PG, whether computations involve image gradient [12] or more sophisticated wavelet transformations [9]–[11]. For all these methods, the question of using the best filters is central, i.e. finding the ones that will extract the most meaningful information. Most of the previous works used hand-crafted filtering step which is unlikely to be optimal.

Our aim is to show that optimizing this filtering step dramatically improves classification results compared to current state of the art. A promising way to do so is to use a Convolutional Neural Network (CNN) to intelligently brute-force the solution.

WIFS'2017, December, 4-7, 2017, Rennes, France. 978-1-5090-6769-5/17/$31.00 ©2017 IEEE.

Deep-learning approaches, and in particular CNN, have recently become very popular for many computer vision problems: classification [18], denoising [19], steganalysis [20], etc. CNN are not only able to learn classification boundaries but also to find conjointly the best representation of the data. We also remark that the overall structure is adapted to our problem: convolution layers model filters while densely connected layers can replace efficiently other classification methods. Motivated by those observations, we implemented a special pooling layer to extract statistical features within a CNN framework and trained the network to distinguish CG from PG.

The paper is organized as follows: Section II presents the overall classification process while Section III to VI describe more precisely each step. Section VII provides a listing of our results and some comparison with Wu et al. [12].

II. Architecture

Our approach mainly differs from the usual pipeline in the size of the image computed. Indeed, classifying smaller images is more challenging since the extracted statistics are less significant but dividing a large image into smaller tiles makes it possible to detect local splicing from a CG to a PG (and reciprocally). Moreover, smaller images are better suited to be used in a CNN due to hardware memory limitations.

In practice, we split the input images into N_p tiles (or patches) of resolution 100×100. This size was chosen as a trade-off between execution time and statistical meaningfulness. An image will be classified to be CG or PG according to a certain decision rule defined in Section VI that aggregates the results of all the tiles composing the image, as depicted in Fig.1.

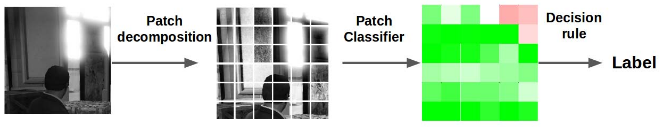


Fig. 1: Pipeline applied to a full-size image: patch decomposition, then patch classification, and finally results aggregation

As for the patch classifier, illustrated in Fig.2, our approach reproduces the usual 3-steps procedure: filtering, statistical feature extraction and classification. It computes N_f filtered images from 100×100 input tiles. Then, it extracts N_s statistical quantities for each filtered image and feeds the $N_f \times N_s$ long feature vector to a usual classifier which estimates the posterior probability of each class. Those values are then used to decide the label of the global image.

The next sections detail each step of the overall procedure.

III. Filtering

The filtering step consists in computing multiple convoluted images from which will be extracted statistical features. As filters are to be learned from the data, we use convolution layers instead of some fixed filters.

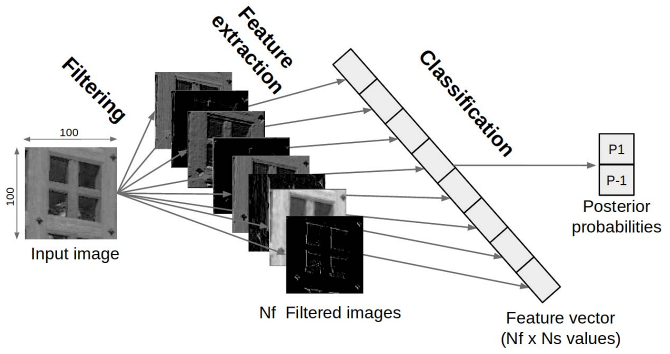


Fig. 2: Patch classifier 3-steps procedure: filtering using CNN, statistical feature extraction and classification

More precisely, we consider a finite set of N convolution kernels of size $k \times k$. For example, with $k = 3$:

$$\left\{ K_{3,n} = \begin{pmatrix} w_{0,0}^n & w_{0,1}^n & w_{0,2}^n \\ w_{1,0}^n & w_{1,1}^n & w_{1,2}^n \\ w_{2,0}^n & w_{2,1}^n & w_{2,2}^n \end{pmatrix} \mid n \in [0..N-1] \right\} \tag{1}$$

where the $w_{i,j}^n$ are the weights which are to be optimized. Then, the input image is convoluted to each kernel. Biases are added and the result is composed with a Rectified Linear Unit (ReLU), a non-linear activation function which improves training performances [21].

Outputted images can be filtered again by stacking other convolution layers. This operation models simple filtering while creating new degrees of freedom (represented by weights and biases). Thus, not only the classifier but also the convolution kernels can be optimized.

After training, we found some well-known kernels used in previous methods but also ome which are less trivial. Those results are shown in Fig.3. For example, kernels 24 and 25 are very close to horizontal difference while kernel 19 looks like a vertical difference. On the other hand, we observed filters like 3 or 11 which seem to combine different kernels (diagonal ones in this case). Searching the space of possible kernels permitted to obtain more complicated filters that would have been hard to find intuitively.

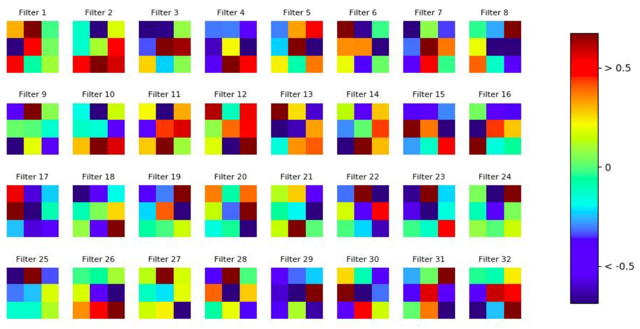


Fig. 3: 32 trained kernels from the first convolution layer of the Stats-2L model

In practice, we tried different parameters for the number of layers N_L and the number of kernels per layer N_K^i, but we fixed the kernel size to $k = 3$. Indeed, during preliminary experiments we observed that increasing k to 5 did not improve the overall results. Moreover, we only considered small networks ($N_L \leq 3$) because the idea is to work only on low-level image representations. All tested architectures are described in section VII.

IV. Statistical features extraction

This Section explains how statistical information is extracted from the convoluted images. In a deep-learning context, this operation can be viewed as a pooling layer. Usually, after convolution layers, a local maximum pooling is computed to reduce the dimension of the data representation before classification. As for image forensics, other global statistical quantities are known to be more useful. Thus a special pooling layer is developed for adapting neural nets to this particular task.

We explored two different approaches: computing simple statistics or estimating the histogram of the convoluted images.

A. Simple statistics

Previous methods (for example [5]) suggest that deriving really simple quantities from filtered images, in general low-order moments of the distribution, is enough to solve our classification problem. In our implementation, for each filtered image F_n with $n \in [0..N-1]$ of size $S_1 \times S_2$ we compute four quantities:

- The estimate mean: $m_n = \sum_{p \in F_n} \frac{p}{S_1 S_2}$
- The estimate variance: $s_n^2 = \sum_{p \in F_n} \frac{(p - m_n)^2}{S_1 S_2}$
- The maximum: $F_n^{max} = \max_{p \in F_n}(p)$
- The minimum: $F_n^{min} = \min_{p \in F_n}(p)$

Remark that those quantities are differentiable with respect to the pixel values which will allow gradients to be back-propagated through this layer during training. This gives a $4 \times N_f$ long features-vector (where N_f is the number of filters in the last convolution layer) which is fed to the classifier.

B. Histograms

The second option is to compute the normalized histogram of the pixel distribution which may capture more information than simple statistical quantities [12]. In order to integrate this layer into our framework, we adapted the way bin values are usually calculated. Their computation includes the use of indicator functions which gradients value zero almost everywhere. This prevents the weights of the convolution layers to be updated during back-propagation. This issue can be avoided using Gaussian kernels instead of indicator function to estimate the value of each histogram bin, as described by Sedighi et al. [20].

We use a 11-bin histogram for each filtered image and the bin values are concatenated into a $11 \times N_f$ long feature vector.

V. Classification

From the feature vectors computed previously (either simple statistics or histograms), we train a classifier to separate the two classes CG and PG. We tried the following popular methods.

A. LDA

Fisher's Linear Discriminant Analysis (LDA) [22] is one of the most widely used tool for linear classification. It aims to maximize the "between class variance" and minimize the "within class variance" over a set of linear classifiers.

B. SVM

Support Vector Machines (SVM) [23] are also very popular and present more flexibility than LDA. In its linear form, SVM is quite similar to LDA (although it minimizes another loss function), but it can also be applied for non-linear classification using the kernel trick. Indeed, feature vectors can be mapped to a higher dimensional space with a kernel function, in our case the radial basis function (RBF).

C. Neural network

Multi-layers perceptrons (MLP) are another example of flexible "model free" classifying method. This kind of neural networks show very good results on various classification problems [24]. It uses a back-propagation algorithm for training which permits to plug our feature extraction layers on top and optimize both at the same time.

Our MLP is composed of a hidden layer with 1024 ReLU activated neurons and a read-out layer with 2 SoftMax activated neurons (one per class). We also use dropout on the hidden layer as described by Strivastava et al. [25] to avoid over-fitting. The loss to minimize is the cross-entropy function for the two-classes problem, which can be interpreted as the minus log-likelihood of the label data under our model. Finally, we used Adam algorithm [26] to optimize synchronously MLP's and convolution layers' weights.

In practice, MLP will be used as the default classifier for training the CNN while SVM and LDA will only be tested by plugging them instead of the MLP after the already trained feature extraction layer.

VI. Output aggregation

A regular image is usually big enough to be composed of many tiles. The classification of a full image will be decided according to the classification probability of each of its tiles. In practice, we use a weighted voting scheme where each patch contribution is the log likelihood of the label:

$$y_{pred} = \operatorname{sgn}\left(\sum_{i=1}^{N_p} \log \frac{\mathbb{P}(Y = 1 \mid X_i = x_i)}{\mathbb{P}(Y = -1 \mid X_i = x_i)} \right) \quad (2)$$

with Y and X_i the random variables modeling the labels and the patches, x_i the real observations and $\operatorname{sgn}(x)$ a function that returns ± 1 according to the sign of x.

We used this rule because it is fairly easy to obtain posterior probabilities (with our MLP for example, it just corresponds to the two output values of the read-out layer) but also because it can be interpreted as a global Maximum Likelihood Estimation criterion for the parameter Y.

VII. Tests and results

A. Database

State of the art methods classically use the Columbia dataset [27] to estimate their performances. Since this dataset was created in 2004, it is obvious that the CG images are completely out of comparison with current CG images and it makes no sense to use them anymore. Thus, we collected our own dataset as follows.

Our CG were downloaded from the Level-Design Reference Database [28] which contains more than 60,000 good resolution (1920×1080 pixels) video-game screenshots in JPEG format. Only 5 different video-games were judged photo-realistic enough (namely The Witcher 3, Battlefield 4, Battlefield Bad Company 2, Grand Theft Auto 5 and Uncharted 4) which reduces the set to 1800 images. Some examples are shown in Fig.4. Furthermore, in-game information (dialogues, life bar, mini-maps, ...) were removed by cropping the images.

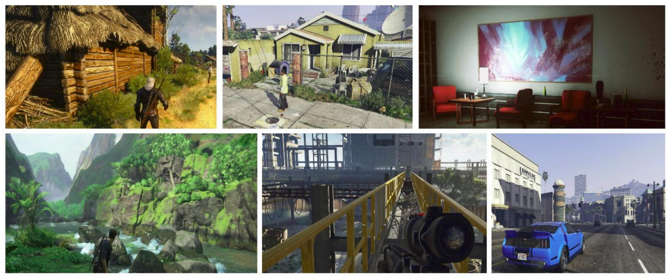


Fig. 4: Sample CG images from our dataset.

The Photographic images are high resolution images (about 4900×3200 pixels) taken from the RAISE dataset [29], directly converted from RAW format to JPEG. Some samples are shown in Fig.5.

Fig. 5: Sample photographic images from our dataset

Images from both classes cover a wide range of outdoor and indoor scenes: landscapes, people bodies and faces, man-made objects (e.g architecture, cars), etc.

From those 3600 images, we constructed 3 databases on which our tests were carried out. Firstly, we selected the green channel of each image. Each class was then divided into training (70%), testing (20%) and validation (10%) to form the Full-size database. From this original database, we constructed a lower size one by cropping each image to 650×650. Finally, we randomly extracted 43000 patches sized at 100×100 for training the patch classifier.

Information about our data are gathered in TABLE I.

TABLE I: CG vs PG datasets description

Name	N_{train}	N_{test}	N_{val}	Size
Full-size	2520	720	360	Various
Low-Size	2520	720	360	650×650
Patch	40000	2000	1000	100×100

B. Experimental setup

Our method was implemented using the Tensorflow framework [30]. Our statistical and histogram layers were directly integrated to the pipeline, using Tensorflow native functions.

During training, we set batch size to 50 images and dropout probability to 0.25 for the MLP hidden layer. Adam algorithm parameters are set as follows: learning rate is $l = 10^{-4}$ and the momentum exponential decay rates are respectively $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

TABLE II: Tested architectures for patch classifier

Name	N_L	N_f^1	N_f^2	N_f^3	Features	N_e	CLF
Stats-1	1	32	NA	NA	Simple stats	128	MLP
Stats-2L	**2**	**32**	**64**	**NA**	**Simple stats**	**256**	**MLP**
Stats-2S	2	16	32	NA	Simple stats	128	MLP
Stats-3	3	32	64	64	Simple stats	256	MLP
Hist-2	2	32	64	NA	Histogram	704	MLP
SVM	2	32	64	NA	Simple stats	256	SVM
LDA	2	32	64	NA	Simple stats	256	LDA

Evaluated architectures are described in TABLE II, with the corresponding number of layers N_L, the number of output filtered images of each layer N_f^1, N_f^2 and N_f^3 (NA if the layer does not exist), the feature extraction method (simple statistics or 11-bin histogram), the length of the extracted feature vector N_e and the final classifying method CLF. We recall that all convolution kernels have a fixed size $k = 3$.

To set up optimal hyper-parameters for the filtering step, we evaluated the first 4 models by plotting the validation accuracy, i.e. the proportion of true classification among the images in the validation set, along training.

Fig.6 shows that the best performing model after training (when the validation curve flattens), has two layers. For this number of layer, using a larger number of kernels is a better choice.

C. Benchmark

For next tests, we have chosen the parameters that gave the best results during validation: $N_L = 2$, $N_f^1 = 32$ and $N_f^2 = 64$.

Our method is then evaluated by computing the detection accuracy and the area under ROC curve (AUC). The closer

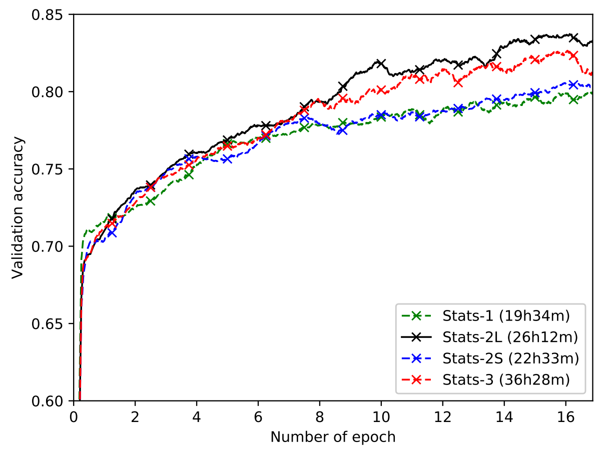


Fig. 6: Smoothed validation accuracy during training for different models

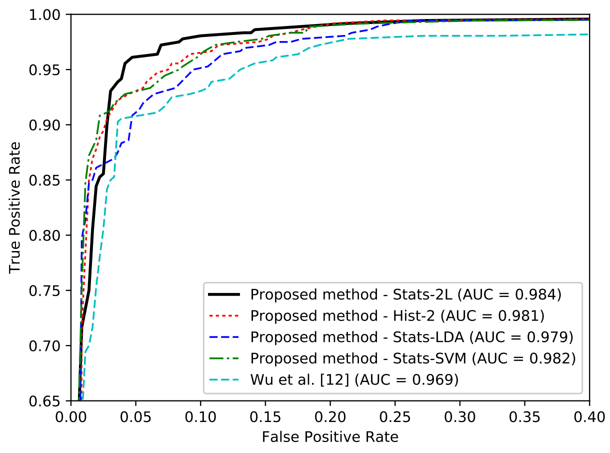


Fig. 7: ROC curve comparison on the full-size testing data

accuracy/AUC are to $100\%/1.0$, the more efficient is the classification.

We compared those results with our implementation of Wu et al. method [12] (the number of histogram features was set to $k = 7$ as suggested by the authors). Final classification was done with LDA and the training/testing procedure was performed on the corresponding part of our datasets. TABLE III shows accuracy for different methods and datasets.

TABLE III: Accuracy results comparison for each dataset (An asterisk * indicates that patch aggregation is used)

Method	Patch	Low-size	Full-size
Wu et al. [12]	75.0%	89.0%	88.5%
Proposed method - SVM	$79,7\%$	$91.4\%^*$	$91.8\%^*$
Proposed method - LDA	79.2%	$91.1\%^*$	$93.1\%^*$
Proposed method - Hist-2	82.7%	$92.4\%^*$	$93.1\%^*$
Proposed method - Stats-2L	$\mathbf{84.8\%}$	$\mathbf{94.4\%}^*$	$\mathbf{93.2\%}^*$

We can see that optimizing the filtering step improves significantly the accuracy for patch by patch classification compared to one of the most effective state of the art method (accuracy rises by 9.8%).

For low-size and full-size datasets, we used the aggregation scheme described in Section VI for all four of the proposed algorithm, with the patch classifier trained on the Patch database. For a fair comparison, Wu et al. baseline was trained on whole images which makes classification straight-forward. In this configuration, our method shows better accuracy results on both datasets. Furthermore, ROC curves and AUC scores concord with this observation on full-size images as depicted in Fig.7.

The best performance is obtained using the Stats-2L model, showing that plugging a SVM or a LDA on top of our feature extractor instead of a MLP leads to less accurate classification.

We also see that training the network with the histogram layer does not improve the results. This can be explained by the lack of training samples which prevents efficient training for more complex models (256 features for simple statistics against 704 for histograms).

Finally, Fig. 8 exhibits some examples of local CG detection using our method. Sample PG have been corrupted by copy/pasting computer graphics and we show that our method permits to visualize which area have been modified.

VIII. Conclusion

In this study, we described a novel method for classifying computer graphics and real photographic images that integrates a statistical feature extraction to a CNN frameworks and finds the best features for efficient boundary . A boosting method is used to aggregate local estimations of the label. To confront our algorithm with nowadays computer graphics, we collected a photo-realistic video-game screenshots database. Compared to state of the art methods, this work is superior in terms of detection accuracy.

References

[1] H. Farid, “Image forgery detection,” *Signal Processing Magazine, IEEE*, vol. 26, no. 2, pp. 16–25, 2009.

[2] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, “Face2face: Real-time face capture and reenactment of rgb videos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2387–2395.

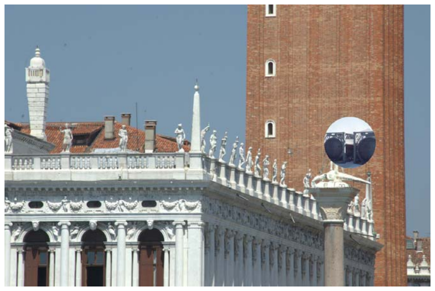
[3] S. Fan, T.-T. Ng, J. S. Herberg, B. L. Koenig, and S. Xin, “Real or fake?: human judgments about photographs and computer-generated images of faces,” in *SIGGRAPH Asia 2012 Technical Briefs*. ACM, 2012, p. 17.

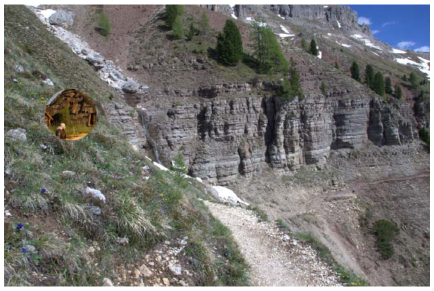
[4] V. Schetinger, M. M. Oliveira, R. da Silva, and T. J. Carvalho, “Humans are easily fooled by digital images,” *arXiv preprint arXiv:1509.05301*, 2015.

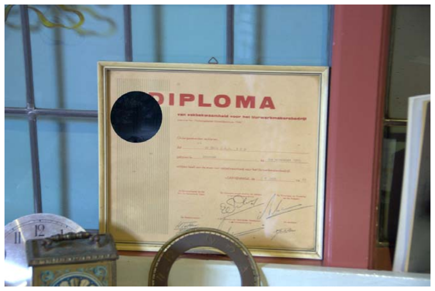
[5] H. Farid and S. Lyu, “Higher-order wavelet statistics and their application to digital forensics,” in *IEEE Workshop on StatisticalAnalysis in Computer Vision*, Madison, Wisconsi, 2003, p. 94.

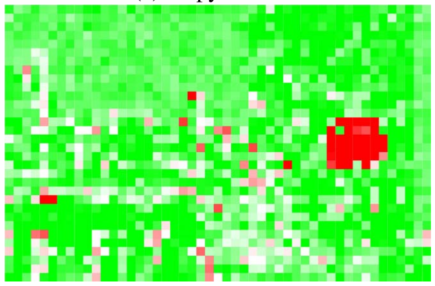
[6] T.-T. Ng and S.-F. Chang, “An online system for classifying computer graphics images from natural photographs,” in *Electronic Imaging 2006*. International Society for Optics and Photonics, 2006, pp. 607 211–607 211.

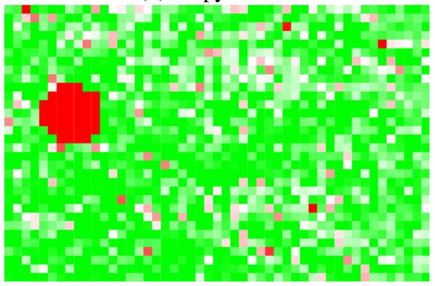
[7] A. E. Dirik, S. Bayram, H. T. Sencar, and N. Memon, “New features to identify computer generated images,” in *IEEE International Conference on Image Processing, ICIP 2007*, vol. 4. IEEE, 2007, pp. IV–433.


(a) Copy/Paste 1


(b) Copy/Paste 2


(c) Copy/Paste 3


(d) Probability map 1


(e) Probability map 2

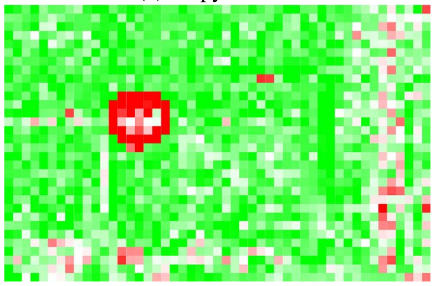

(f) Probability map 3

Fig. 8: Examples of CG copy/paste detection on PG. Patches identified as CG (respectively PG) are in red (respectively in green). The intensity of the coloration represents the posterior probability for the corresponding class.

[8] R. Wang, S. Fan, and Y. Zhang, "Classifying computer generated graphics and natural imaged based on image contour information," *J. Inf. Comput. Sci*, vol. 9, no. 10, pp. 2877–2895, 2012.

[9] Y. Wang and P. Moulin, "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.

[10] X. Cui, X. Tong, G. Xuan, and C. Huang, "Discrimination between photo images and computer graphics based on statistical moments in the frquency domain of histogram," *Procedings of the CIHW2007, Nanjing*, pp. 276–283, 2007.

[11] W. Chen, Y. Q. Shi, G. Xuan, and W. Su, "Computer graphics identification using genetic algorithm," in *19th International Conference on Pattern Recognition, ICPR 2008*. IEEE, 2008, pp. 1–4.

[12] R. Wu, X. Li, and B. Yang, "Identifying computer generated graphics via histogram features," in *18th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2011, pp. 1933–1936.

[13] W. Li, T. Zhang, E. Zheng, and X. Ping, "Identifying photorealistic computer graphics using second-order difference statistics," in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, vol. 5. IEEE, 2010, pp. 2316–2319.

[14] N. Khanna, G. T.-C. Chiu, J. P. Allebach, and E. J. Delp, "Forensic techniques for classifying scanner, computer generated and digital camera images," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008*. IEEE, 2008, pp. 1653–1656.

[15] A. C. Gallagher and T. Chen, "Image authentication by detecting traces of demosaicing," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW'08*. IEEE, 2008, pp. 1–8.

[16] E. Tokuda, H. Pedrini, and A. Rocha, "Computer generated images vs. digital photographs: A synergetic feature and classifier combination approach," *Journal of Visual Communication and Image Representation*, vol. 24, no. 8, pp. 1276–1292, 2013.

[17] T.-T. Ng and S.-F. Chang, "Discrimination of computer synthesized or recaptured images from real images," in *Digital Image Forensics*. Springer, 2013, pp. 275–309.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[20] V. Sedighi and J. Fridrich, "Histogram layer, moving convolutional neural networks towards feature-based steganalysis," in *Electronic Imaging, Media Watermarking, Security, and Forensics 2017*, 2017.

[21] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, J. Frnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 807–814. [Online]. Available: http://www.icml2010.org/papers/432.pdf

[22] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2000.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.

[25] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] T.-T. Ng, S.-F. Chang, J. Hsu, and M. Pepeljugoski, "Columbia photographic images and photorealistic computer graphics dataset," *Columbia University, ADVENT Technical Report*, pp. 205–2004, 2005.

[28] M. Piaskiewicz. (2017, May) Level-design reference database. [Online]. Available: http://level-design.org/referencedb/

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

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/