Signal Processing: Image Communication 66 (2018) 113–126

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

Exposing computer generated images by using deep convolutional neural networks

Edmar R.S. de Rezende [a], Guilherme C.S. Ruppert [a], Antônio Theóphilo [a], Eric K. Tokuda [c], Tiago Carvalho [b,*]

[a] CTI Renato Archer, Campinas-SP, 13069-901, Brazil
[b] Federal Institute of São Paulo (IFSP), Campinas-SP, 13069-901, Brazil
[c] University of São Paulo (USP), São Paulo-SP, 05008-090, Brazil

ARTICLE INFO

Keywords:
Digital forensics
CG detection
Deep learning
Transfer learning
Fake news

ABSTRACT

The recent computer graphics developments have upraised the quality of the generated digital content, astonishing the most skeptical viewer. Games and movies have taken advantage of this fact but, at the same time, these advances have brought serious negative impacts like the ones yielded by fake images produced with malicious intents. Digital artists can compose artificial images capable of deceiving the great majority of people, turning this into a very dangerous weapon in a timespan currently know as "Fake News/Post-Truth" Era. In this work, we propose a new approach for dealing with the problem of detecting computer generated images, through the application of deep convolutional networks and transfer learning techniques. We start from Residual Networks and develop different models adapted to the binary problem of identifying if an image was, or not, computer generated. Differently from the current state-of-the-art approaches, we do not rely on hand-crafted features, but provide to the model the raw pixel information, achieving the same 0.97 performance of state-of-the-art methods with three main advantages: (i) executes considerably faster than state-of-the-art methods with equivalent accuracy; (ii) eliminates the laborious and manual step of specialized features extraction and selection, and (iii) is very robust against image processing operations as noise addition, blur and JPEG compression.

1. Introduction

The 2016 *Global Games Market Report*[1] presented the economic potential of digital games, which traded more than 99.6 billions of dollars, an increment of 8% when compared to the previous year. The growth in this market always pushes forward the quality and development of associated industries and technologies as, for example, computer graphics methods. These methods are essential to make games more realistic through high quality graphics.

Another entertainment field that takes advantage of advanced computer graphics methods is the movies industry. Thinking about realism, in the last years we have experienced huge steps towards a complete deceiving of our visual senses. Productions as *Rogue One: A Star Wars Story* showed the potential of Computer Graphics (CG) characters construction, introducing in a live action movie characters entirely based on real actors.

The search for a perfect generation of digital scenarios, objects and even people is endless and recently reached an astonishing point, mostly helped with the latest advances of computing processing, in special the modern GPU cards (Graphics Processing Units). One current example of such an achievement was the digital reproduction of the actress Carrie Fisher in the last Star Wars movie,[2] with the same appearance of the beginning of her career in the 70's.

In spite of the safe and benign results of these advances, once the goal of perfect CG image generation is accomplished, some threats come along and introduce new challenges to other science areas as pointed out by Holmes et al. [1]. One example of such a challenge is the identification if an image was a photo generated (PG — the one generated by a digital camera) or generated by CG methods. Fig. 1, shows an example of how difficult is to discern between PG and CG images.

Recent studies showed how easy is to deceive people using images [2]. In special, several examples of undesired situations can be described involving the CG images. Imagine, for example, a CG image

[2] http://www.imdb.com/title/tt3748528/.

* Corresponding author.
E-mail address: tiagojc@gmail.com (T. Carvalho).
[1] https://goo.gl/xkWPon.

https://doi.org/10.1016/j.image.2018.04.006
Received 29 November 2017; Received in revised form 7 February 2018; Accepted 6 April 2018
Available online 24 April 2018

(a) PG. (b) CG.

Fig. 1. Example of how challenging is to recognize PG and CG images by simple visual analysis.

depicting a terrorist execution of a kidnapped report spreading across the globe. Or another CG image of a rising politician, posted in social networks putting him in an embarrassing or criminal situation days before an election. We are living in what some are calling the Fake News/Post-Truth Era [3,4], where mass communication platforms (as social media networks) can be used to influence and deceive people [5,6]. If a well-crafted invented text can have a great impact on public opinion, imagine the effects of a CG image produced by a very (probably well-paid) skilled professional posted on social networks.

The distinction between PG and CG has even more complex legal implications when related to child pornography. In Brazil, any person who produces, reproduces, directs, takes pictures or records, in any way, scenes involving explicit sexual or pornographic act involving children or teenagers, can be sued according to Brazilian Law N° 11,829 published on November 25th, 2008. This legal process can result in 4 to 8 years in jail. This situation raises a fundamental legal and ethical question created by technology: What happens if the material is proved to be CG generated? Are the legal consequences the same?

The task of CG image and video detection was already studied and several Digital Forensics methods were proposed [7–10]. However, the results are far from considering the problem as completely solved. Very often these methods are based on the discovery of inconsistencies in very specific situations, hindering their wide application. For example, Conotter et al. [11] developed a method based on blood flow information of CG constructed people in videos. In contrast, Tokuda et al. [7] proposed a more generic method that applies machine learning techniques to solve the CG image identification task and is more similar with the one presented by this work.

The rise of Deep Neural Networks (DNN) in the past few years, presented a shift in classification process, specially in the feature engineering step of this process. Algorithms based on DNN have outperformed other approaches in image classification, becoming the standard approach for these tasks. They consist of learning algorithms with multiple stages, acting over the raw input (image pixels, for example) transforming the representation at one level into another at a higher, slightly-more-abstract level [12–14]. As this stack of layers gets bigger, more complex functions can be learned from data. Besides this power of representing more complicated mappings, a great advantage of DNN is that there is no need for human engineered features, with a general purpose algorithm learning direct from raw data.

In spite of the basic concepts of DNN being around for some decades, only now, with the plenty availability of data and the recent developments of GPU cards DNN showed its full potential, specially in image classification challenges such as the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) [15]. This highlighted the transition from hand-crafted features combined with shallow classifiers to deep classifiers acting directly on raw data as is the case of DNN.

Since then, there is been a trend of, as deeper the model is, the better is its performance and more difficult is the training process. This can be demonstrated by ImageNet challenge results. In 2012 the 8-layers AlexNet network [16] astonished the machine learning community winning the challenge with a top-5 classification error rate of 16.4% and a huge leap from the second place (this one using usually hand-crafted features and shallow classifiers). In 2014, two VGG DNN models (one with 16 layers and the other with 19 layers) got a top-5 classification error rate of 7.3% [17] while GoogleNet with its 22 layers won the challenge with 6.7% error rate [18]. Finally, in 2015, the Residual Network (ResNet) model, a DNN with 152 layers, achieved a top-5 classification error of 3.57% [19]. Also, in the same year, for the first time was presented a DNN technique capable of performing better than humans in image classification tests [20].

This paper presents a novel approach for dealing with the task of detecting CG image generation. Two different models are developed, starting from the DNN ResNet with 50 layers (ResNet-50) [19] and adapting it to the binary problem of CG image detection. Applying concepts of transfer learning [21], we were able to transfer the weights of ResNet-50 layers pre-trained on ImageNet dataset to our model, avoiding overfitting and achieving 97% of accuracy without the burden of designing complex hand-craft features. To our best knowledge, this is the first work to propose applying DNN techniques to this problem, not requiring human experts to design features.

Regarding the actual state-of-the-art method for detection of CG image generation [7], the main contributions of this paper are: (i) the proposal of a new approach based on DNN and transfer learning techniques that achieve the same accuracy of 0.97 as state-of-the-art methods without the need for human level feature extraction; (ii) the use of an extended dataset (more difficult for the task); (iii) a method robust against image processing operations as noise addition, filtering and JPEG compression; (iv) a faster method when compared against state-of-the-art methods[3] ; (v) evaluation of different kinds of classifiers in association with a DNN in order to find the best combination (features + classifier); (vi) and a qualitative analysis of bottleneck features produced by ResNet-50 in CG image detection problem.

The text is structured in the following way: Section 2 briefly presents the main works in the Digital Forensics literature that deal with the problem of detecting CG image generation. Section 3 explains with details the proposed methodology while Section 4 describes the main experiments conducted to validate the methodology and presents the achieved results, comparing with the state-of-the-art found on the literature. Lastly, Section 5 presents the main conclusions and some future research directions.

2. Related work

There are many works in the literature on the topic of distinguishing between CG and real images. Holmes et al. [1] discusses the legal aspects related to the problem, specially for child pornography. The authors investigated the perception of humans exposed to this kind of image performing two experiments: (i) the first one in which a set of images (CG and real) are shown to untrained users, and (ii) the second where there was a previous training for users before showing the images. The experiment consisted in submitting each user to 60 pictures of people. The user were asked to identify the sex (man or woman) and if the image was real or generated by computer. In the first round of experiments, the untrained users achieved an accuracy around 50% in CG image

[3] All the artifacts (code and dataset) produced by this work will be available in case of paper acceptance.

detection. In the second experiment, after a simple training, the users improved their accuracy at the task. Also, as the CG image quality improves, it becomes even easier to trick the perception of the user to distinguish real and CG images.

Farid [22] discusses the US Supreme Court decision on not considering a crime the computer generated child pornography and presents techniques for image tampering as well as approaches to detect some kinds of image manipulations.

Conotter et al. [11] proposed to use information associated with blood flow and perceptual details to detect computer generated people in videos. The method consists in evaluating small movements of cheeks and forehead to generate a distinguishable signal of CG and real images. This signal is more stable for real images, while CG images present many peaks.

Another work aimed at finding computer generated people in videos is the one presented by Dang-Nguyen et al. [23]. The authors propose an analysis based on a 3D model in order to identify synthetic characters detecting their limited variability over time.

Many methods based on machine learning have been proposed, which typically consists in extracting features and using a supervised learning classifier to identify patterns of CG or real images. Tokuda et al. [7] proposes a method using fusion of many classifiers combined with a big number of feature extraction schemes, achieving a 97% accuracy on his dataset (9700 images).

Tan et al. [24] uses Local Ternary Patterns (LTP) for features extraction, and rely only on texture features to distinguish CG and real images. Experiments reveal that the method achieves an accuracy of approximately 97% in a dataset of 2200 images collected from different sources, as for example, the Columbia University natural image library [25], using a Support Vector Machine (SVM) [26] classifier.

3. Proposed method

The CG detection method proposed in this work relies upon a deep architecture based on a convolutional neural network (CNN) to classify each image from the dataset using the raw RGB pixels values as features, without the need for manual feature extraction. The deep CNN deployed is based on the ResNet-50 model [19] and the method uses transfer learning techniques [21]. All the pipeline of the proposed method as well as fundamental concepts related with it will be explained in the next sections.

3.1. Complete model architecture

Our proposed deep CNN model uses transfer learning techniques, leveraging the outcomes of residual learning presented by He et al. [19]. The final model architecture, with its pipeline depicted in Fig. 2, consists of:

1. an initial pre-processing stage;
2. a sequence of many convolutional layers based on the first 49 layers of ResNet-50 and;
3. a top classifier replacing the original 1000 fully-connected softmax layer.

In a real word dataset, images can present different resolutions. However, as our model requires a constant input dimensionality, we resize the images to a fixed resolution of 224 × 224 and, for each pixel, we subtract the mean RGB value computed over the ImageNet dataset (as proposed by Krizhevsky et al. [16]). These two operations are performed by the pre-processing layer.

After pre-processing the dataset images, we apply the transfer learning techniques explained in Section 3.3. In our CNN model, after the pre-processing, we use the first 49 layers of ResNet-50 with their weights trained over ImageNet as a features extractor (red box named "Bottleneck Features Extractor" of Fig. 2) to generate a set of features with the correspondent label associated. These labeled features, also called bottleneck features[4] are the activation maps generated by the average pooling layer (the 49th layer of ResNet-50), ignoring the last 1000 fully-connected softmax layer.

These bottleneck features are used to train a top classifier that will make the final prediction of CG images. This classifier has the same role as the original softmax layer at the end of ResNet-50, adapting the network to the binary problem of CG image detection. The replacement of this softmax layer, with thousands of parameters, associated with the transfer learning techniques used (no learning happens at the convolutional layers), allowed us to deploy a very deep CNN for the CG image detection problem without the requirement of millions of CG/PG labeled images, besides significantly reducing the training time.

Once finished the training process, our final deep CNN model used for testing is made up of the pre-processing layer, the Bottleneck Features Extractor and a top classifier (network on the right of Fig. 2). Different types of classifiers were trained as top predictors in order to discover which one performs best for the task of CG image detection. Section 3.4 will delve into the details of each type used.

3.2. ResNet-50

Residual Networks (ResNet) [19] can be classified as convolutional neural networks (CNN). These CNN, in turn, can be defined as neural networks that have at least one layer using the convolution operation [13]. Mathematically speaking, the convolution operation can be view as a weighted average operation of two functions (x and w), where one of them (w) is a probability density function. More formally:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{1}$$

In practice, due to the commutative property, the convolution operation is usually implemented as the cross-correlation function [13]. For example, assuming the function x as a two-dimensional input image I and the probability density function w as a function K (usually called *kernel* in machine learning terminology), the function S below is the convolution of *kernel* K over the image I:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2}$$

In machine learning nomenclature, the function S is usually called a *feature map*. Among many interesting properties that convolution operation conveys, one of paramount importance is the robustness to translation in the recognition of patterns. If a kernel K is specialized in recognizing circles, the convolution of this kernel over the image will identify circles no matter where they occur in the image.

Besides convolutions, CNN usually have some non-linear activation functions (like sigmoid or rectified linear functions) and pooling layers, the latter having the effect of turning the representation invariant to small translations in the image [13].

Residual Networks (ResNets) [19] are deep convolutional networks where the basic idea is to skip a series of convolutional layers by using shortcut connections to form conceptual shortcut elements named residual blocks. The residual block can be formally described in the general form:

$$\mathcal{Y}_l(\mathbf{x}, W) = f(h(\mathbf{x}) + \mathcal{F}(\mathbf{x}, W)) \tag{3}$$

where $\mathbf{x}$ is the input of the block, h is the *shortcut* function (crucial to ResNet and better explained below), $\mathcal{F}$ is the mapping done by a series of one or more consecutive convolutional layers, W is the weights of these

[4] Bottleneck term refers to a neural network topology where the hidden layer has significantly lower dimensionality than the input layer, assuming that such layer –referred to as the bottleneck –compresses the information needed for mapping the neural network input to the neural network output, increasing the system robustness to noise and overfitting. Conventionally, bottleneck features are the output generated by the bottleneck layer.

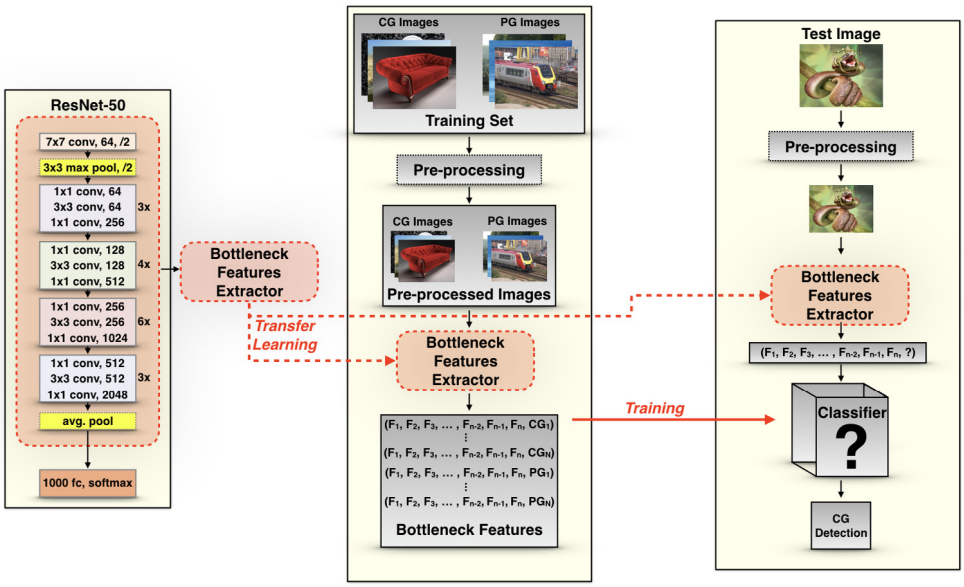


Fig. 2. Overview of proposed method. ResNet-50 parameters are transferred to our model to extract bottleneck features, which are used to train different classifiers.

convolutional layers, f is a rectified linear unit (ReLU) and $\mathcal{Y}_l$ is the mapping that describes the residual block l as a function of the input $\mathbf{x}$ and the weights W. The shortcut function is usually the identity function (or a very simple convolutional layer when a dimensional compatibility is needed) and its purpose is to speed up the learning of the mapping $\mathcal{F}$ as a perturbation of the input $\mathbf{x}$, starting this learning from a point near the identity function. This is faster to learn than if done from a random point near the zero-mapping (as done by previous architectures) and is one of the key contributions of Residual Networks.

In ResNet-50 architecture, the basic residual blocks, also called bottleneck blocks, are composed of a sequence of three convolutional layers with filters of size 1×1, 3×3 and 1×1 respectively. The down-sampling is performed directly by convolutional layers that have a stride of 2 and batch normalization [27] is calculated right after each convolution and before ReLU activation.

The identity shortcuts can be directly used when the input and output feature maps are of the same dimensions. When the dimensions change, two options are considered: (i) The shortcut still performs identity mapping, with extra zero entries padded for increasing filter dimensions (depth). This option introduces no extra parameter; (ii) A 1×1 convolution layer is used to match the dimensions. This is called *projection* shortcut. For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2. Fig. 3 tries to clarify this set-up.

The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax activation. The total number of weighted layers is 50.

3.3. Transfer learning

Deep learning techniques usually require a very large number of samples and demands a heavy computing effort. In this scenario, transfer learning [21] has gained a lot of attention. It represents the possibility to transfer the knowledge learned from one problem to another problem. For neural networks, the transfer learning process consists, in practice, in transferring the parameters of a (source) neural network that was previously trained over a particular dataset for a specific task, to another (target) network that will act over a different dataset to solve a different problem.

A typical transfer learning procedure implies in using an already trained base network and copying all the parameters from the first n layers to the first n layers of the target network. Then, a supervised training is performed only on the remaining layers that were not copied. Since the task is different and probably so is number of classes, the last layer has to be modified to contain the same amount of neuron as the number of classes or it can be replaced by another classifier. During this training, the copied layers can be left fixed, or it can be fine-tuned by allowing the backpropagation process into the copies parameters. Usually fine-tuning improves the accuracy, however, when the number of parameters is big and the numbers of samples is small, this may result in overfitting and fine-tuning should not be used.

In traditional supervised learning, the common sense was settled that the training should always be performed specifically for a given task and dataset and the transfer learning approach could sound senseless. However, in general, deep neural networks present a singular characteristic which is to learn features in the first layers that are more general and not specific to that particular dataset and problem. This turns that knowledge reusable for other problems and datasets. On the other hand, the knowledge learned on the last layers of the neural network are more specific for the associated task.

Transfer learning is very handy because it avoids the job of training the whole network. Since the architecture is very deep, training it represents an enormous computational effort, requiring expensive high processing computers usually using multiple high-end GPU units. Another problem in training deep networks from scratch is that it requires a vary large number of samples, otherwise it will overfit the data. Transfer learning significantly mitigates these two problems, enabling the use of deep learning even when the target dataset is small and/or when there is limited computing resources. Recent studies have taken advantage of this fact to obtain state-of-the-art results [28–30], which evidences the generality of the features learned in the first layers of the network.

In this work, we apply transfer learning techniques by using the same parameters of the ResNet-50 network trained over the ImageNet 2012 competition [15], which provided a dataset of 1.28 million images spread over 1000 classes. The first 49 layers of the network were copied to a new ResNet-50 network and we removed the last layer, replacing it by another classifier. We have not used fine-tuning during the training due to the relatively small dataset.

3.4. Top classifiers

In the proposed method, the last layer of the ResNet-50 is replaced by another classifier and we have evaluated four different classifiers for this task.

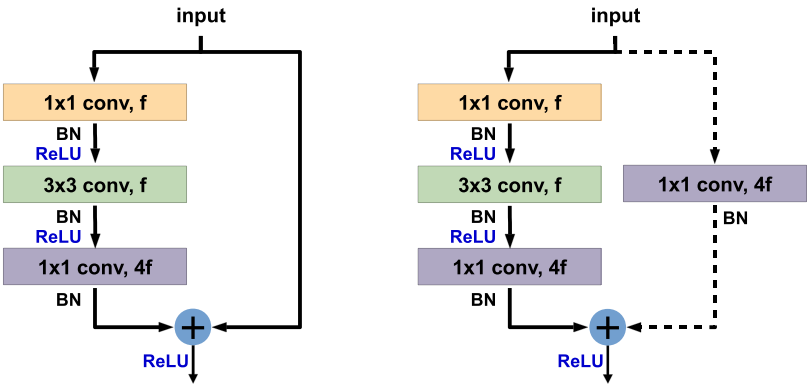


Fig. 3. Bottleneck Blocks for ResNet-50 (left: identity shortcut; right: projection shortcut).

3.4.1. Softmax

The softmax function [26] is widely used in deep learning architectures and consists in the generalization of the binary logistic regression to multiple classes.

This function is particularly interesting because it provides an intuitive output with probabilistic interpretation. The outcome of the function is a vector containing the probability for each class.

Typically, the softmax is used for classification as the activation function on the last fully-connected layer of CNN, which is the case in the ResNet-50. The function transforms a vector $\mathbf{z}$ with dimension K of real numbers z_k, to another vector $\sigma(\mathbf{z})$ of same dimension with the values ranging from 0 to 1. The sum of the output vector adds up to 1, therefore it can be interpreted as the probabilities for each class. The formula is given by:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4}$$

For training, we used the categorical cross-entropy loss function, which is given by:

$$H(p,q) = -\sum_{x} p(x).log(q(x)) \tag{5}$$

3.4.2. k-Nearest Neighbor (kNN)

The k-Nearest Neighbor (kNN) [26] classifier is one of the simplest and most popular supervised classifier. It consists in classifying a sample based on the k-nearest samples from the training set. Usually, the euclidean distance function is used, but other distance functions may be chosen. The sample is then classified by the majority voting or other similar function among the k-nearest samples. The advantage of kNN is that it is very simple, does not require an explicit training step and yet it is very effective for many applications, specially when the training set is large. The main drawbacks of this method are that (i) the method requires the distance computation to all samples from the training dataset, which makes it computationally heavy; and (ii) the fact that it also requires a lot of memory since the whole dataset has to be loaded for comparison.

3.4.3. XGBoost

Extreme Gradient Boosting (XGBoost) [31] is a recent work that has been gaining a lot of attention for impressive results in machine learning challenges like KDD Cup and Kaggle competitions.

XGboost consists of an open-source package that implements gradient tree boosting algorithm with the focus on being highly effective and scalable. It includes novel optimized algorithms related to: more efficient parallelization; a novel sparse-aware tree learning algorithm; out-of-core computation, cache-aware access; distributed weighted quantile sketch method; among other improvements. All these improvements allow the system to perform more than ten times faster than other tree boosting solutions.

The library was written in C++, however binding for other languages like Python, R and Java are available.

3.4.4. SVM

Support Vector Machines (SVM) [26] is one of the most popular supervised classifiers and basically operates by finding the optimum hyperplane that best separates two classes. Originally designed for binary classification, it can be extended for multi-class problems by reducing one multi-class task to multiple binary classification problems using techniques known as one-versus-one or one-versus-all, among other methods.

Although the original SVM is a linear classifier, it can be applied for non-linear problems by using kernel functions which nonlinearly maps the feature vector to a new space.

In the present work, we evaluate both the linear SVM and the SVM with a Radial Basis Function (RBF) kernel, for the top classifier of the architecture.

4. Experiments and results

To validate the proposed approach, we have performed different rounds of experiments which will be detailed in the next sections.

4.1. DSTok dataset

The first dataset in which the proposed method has been tested against is a public dataset proposed by Tokuda et al. [7]. It comprises 4850 CG images and 4850 PG images, depicting different kinds of scenarios as people, outdoor, objects, cars, animals, and others. The entire set of images has been collected from Internet and compressed in JPEG format, presenting images with sizes from 12 KB to 1.8 MB. Images in the dataset present different resolutions and, differently from Tokuda et al. [7], our proposed method works with the entire image (without cropping). Fig. 4 depicts some examples of images in DSTok dataset.

4.2. DSTokExt dataset

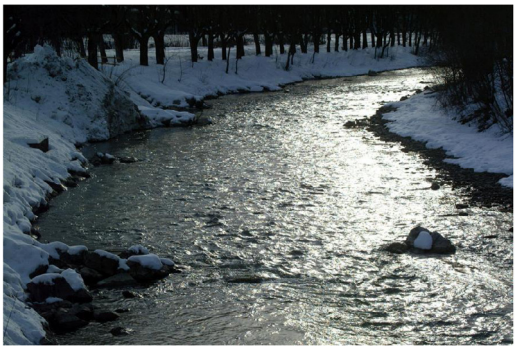
The second dataset used to validate the method is an extension of Tokuda et al. [7] dataset. It comprises 8394 CG images and 8002 PG images, also depicting different kinds of scenario. In the same way as Tokuda et al. [7], all the images have been collected from Internet and compressed in JPEG format, presenting images with sizes from 12 KB to 1.8 MB. Images in the dataset present different resolutions. Fig. 5 depicts some examples of images in DSTok dataset.

4.3. Validation protocol

In order to compare the results achieved by the proposed method with the results reported by Tokuda et al. [7], we perform the same five fold cross-validation protocol, reporting the accuracy by fold and the average accuracy for each round of experiments.

(a) CG.

(b) CG.



(c) PG.

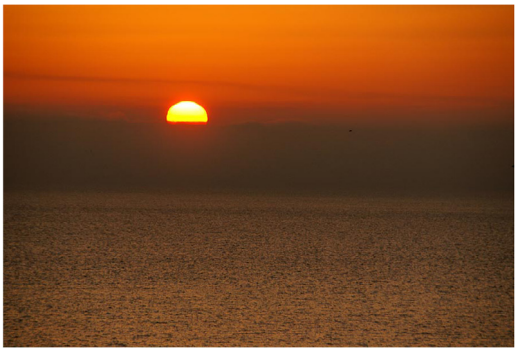
(d) PG.

Fig. 4. Examples of images in DSTok dataset.

(a) CG.

(b) CG.

(c) PG.



(d) PG.

Fig. 5. Examples of images in DSTokExt dataset.

Table 1
Accuracy by fold of ResNet-50 trained from Glorot uniform initialization.

Fold	Accuracy	Time (s)
0	0.79	17,387.85
1	0.72	17,352.25
2	0.80	17,357.33
3	0.74	17,353.05
4	0.74	17,399.03
Average	0.76	17,369.90

Table 2
Accuracy by fold of ResNet-50 fine-tuned from ImageNet initialization.

Fold	Accuracy	Time (s)
0	0.86	16,888.32
1	0.82	16,895.00
2	0.85	16,897.95
3	0.77	16,879.80
4	0.80	16,884.62
Average	0.82	16,889.14

Table 3
Accuracy by fold of 2 fully-connected softmax layer trained with bottleneck features and ResNet-50 fine-tuned from ImageNet initialization.

Fold	Pre-train Accuracy	Model Accuracy	Time (s)
0	0.91	0.93	16,716.76
1	0.89	0.90	16,618.89
2	0.91	0.92	16,596.87
3	0.89	0.91	16,620.85
4	0.90	0.93	16,619.18
Average	0.90	0.92	16,634.51

4.4. Implementation details

The proposed methods have been implemented using Python 3.5, Keras 2.0.3,[5] and TensorFlow 1.0.1.[6] All performed tests have been executed in a machine with an Intel(R) Xeon(R) CPU E5-2620 2.00 GHz processor with 96 GB of RAM and two Nvidia Titan Xp GPUs.

4.5. Round #1: ResNet-50 trained from Glorot uniform initialization over DSTok

In the first round of experiments we classify samples from DSTok using a deep CNN architecture similar to the original ResNet-50. Since we have only 2 classes (CG and PG), we have adapted the ResNet-50 architecture to the CG detection task replacing its last 1000 fully-connected softmax layer by a 2 fully-connected softmax layer. The weights of the network have been initialized using Glorot uniform approach [32] and the bias terms were initialized to zero. All layers of the model have been trained for 200 epochs with categorical cross-entropy cost function and Adam optimizer ($lr = 0.001$, $beta_1 = 0.9$, $beta_2 = 0.999$, $epsilon = 1e-08$ and $decay = 0.0$).

Fig. 6 presents the average accuracy and loss of ResNet-50 trained from Glorot uniform initialization. Solid lines represent the average performance in the training (red) and testing (green) set while the shadows represent the standard deviation in the 5-fold cross-validation.

The results are presented in Table 1. This table shows that the 75.85% average accuracy was achieved with a training time around 17,369.90 s after 200 training epochs.

4.6. Round #2: ResNet-50 fine-tuned from ImageNet initialization over DSTok

In the second round of experiments, we evaluate the impact of transfer learning as a strategy to initialize the weights of the convolutional layers in the proposed model. We transfer the weights of ResNet-50 convolutional layers pre-trained on ImageNet dataset to our deep CNN model, replacing the last 1000 fully-connected softmax layer by a 2 fully-connected softmax layer.

In the first round of experiments, all network parameters (including the last layer) have been initialized using Glorot uniform approach and the bias terms were initialized to zero. At this round of experiments, we use ImageNet parameters as initial weights, except in the last layer where, again, we apply Glorot uniform initialization. Then, all layers have been trained for 200 epochs with categorical cross-entropy cost function and Adam optimizer ($lr = 0.001$, $beta_1 = 0.9$, $beta_2 = 0.999$, $epsilon = 1e-08$ and $decay = 0.0$).

Fig. 7 presents the average accuracy and loss of ResNet-50 fine-tuned from ImageNet initialization. Solid lines represent the average performance in the training (red) and testing (green) set while the shadows represent the standard deviation in the 5-fold cross-validation.

The results by fold are presented in Table 2. This table shows that the 82.12% average accuracy was achieved with a training time around 16,889.14 s after 200 training epochs.

[5] https://keras.io.
[6] https://www.tensorflow.org.

Given the improvement of 7% over the average accuracy obtained in the experiments of Round #1, we can conclude that the knowledge transferred from ImageNet dataset to CG image detection problem produced good results.

4.7. Round #3: ResNet-50 fine-tuned from ImageNet initialization and pre-trained softmax layer over DSTok

In Round #2 of experiments, we showed that the transfer learning in fact helps to improve the model accuracy of CG detection task. However, the random initialization of the last fully-connected softmax layer could cause an undesired drawback, backpropagating the error from the last layer to the ImageNet transferred weights during the fine-tune step along the entire network, degrading the model accuracy.

Therefore, in this round of experiments, we pre-train the last fully-connected softmax layer before the fine-tuning step along the entire network. To perform this, we initialize the convolutional layers with ImageNet weights and freeze them. Then, the weights of the last layer are initialized using Glorot uniform approach, the bias terms are initialized to zero and the network is pre-trained for 200 epochs with categorical cross-entropy cost function and Adam optimizer ($lr = 0.001$, $beta_1 = 0.9$, $beta_2 = 0.999$, $epsilon = 1e-08$ and $decay = 0.0$). This procedure results in the training only of the softmax layer‘. After this step, we unfreeze convolutional layers, training all layers of the model for 200 epochs with categorical cross-entropy cost function and SGD optimizer ($lr = 0.0001$, $momentum = 0.9$, $decay = 0.0$ and $nesterov = False$), using a smaller learning rate to train the network. Since we expect the pre-trained weights to be quite good already as compared to randomly initialized weights, we do not want to distort them too quickly and too much.

Fig. 8 presents the average accuracy and loss of ResNet-50 fine-tuned from ImageNet initialization and pre-trained softmax layer. Solid lines represent the average performance in the training (red) and testing (green) set while the shadows represent the standard deviation in the 5-fold cross-validation.

The results by fold are presented in Table 3. As can be seen in the table, the 2 fully-connected softmax layer pre-trained with bottleneck features (obtained freezing ImageNet weights in convolutional layers) achieved an average accuracy of 89.90% after 200 pre-training epochs and the model fine-tuned after this pre-training step achieved an average accuracy of 91.96% with a training time around 16,634.51 s after 200 training epochs.

These results confirm our hypothesis that, despite the disparity between object detection and CG detection tasks, ResNet-50 comprehensively trained on the large-scale well-annotated ImageNet may still

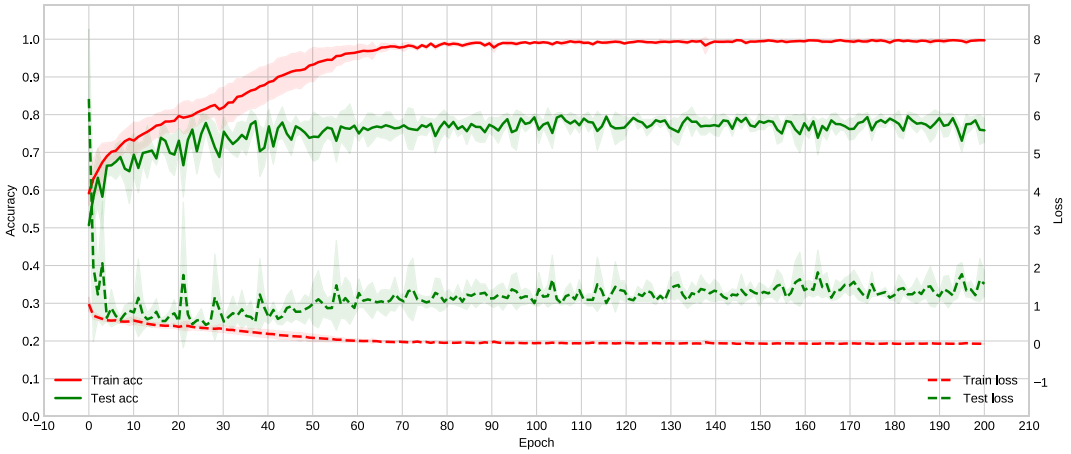


Fig. 6. Train/test average accuracy/loss of ResNet-50 trained from Glorot uniform initialization.

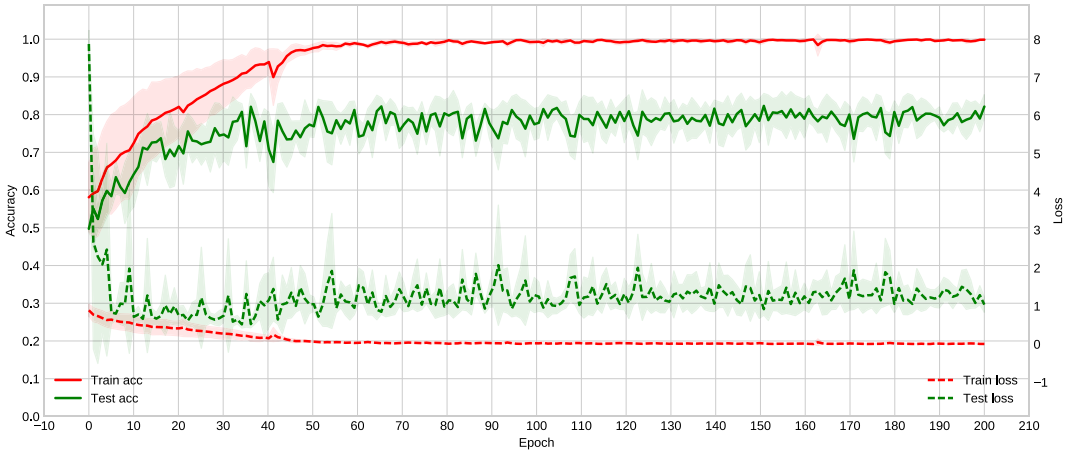


Fig. 7. Train/test average accuracy/loss of ResNet-50 fine-tuned from ImageNet initialization.

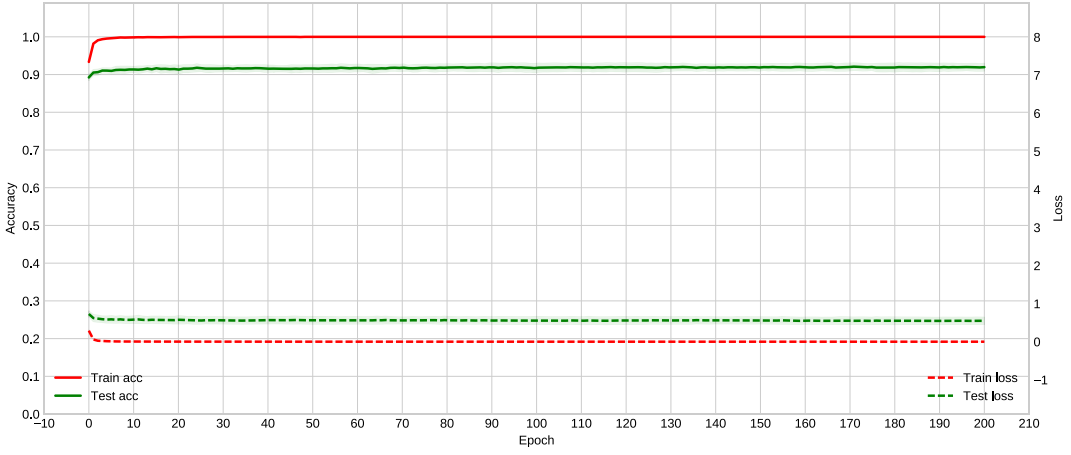


Fig. 8. Train/test average accuracy/loss of ResNet-50 fine-tuned from ImageNet initialization and pre-trained softmax layer.

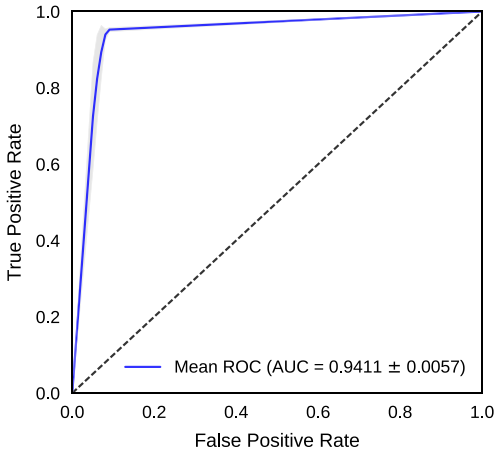


Fig. 9. ROC curve for the best result in DSTok dataset using ResNet-50 + SVM RBF Kernel.

be transferred to make CG detection task more effective. Furthermore, it is important to highlight that ResNet-50 bottleneck features provide a very discriminative image descriptor for CG detection problem.

4.8. Round #4: ResNet-50 bottleneck features with Shallow Classifiers over DSTok

Based on the promising results obtained with the knowledge transfer of ResNet-50 convolutional layers pre-trained on ImageNet dataset, in this round of experiments we evaluate the performance of transfer learning combined with shallow classifiers.

Therefore, we replace the last fully-connected softmax layer of ResNet-50 by shallow classifiers in order to classify images represented by bottleneck features. We evaluate the performance of proposed method replacing the top layer by three different classifiers: Support Vector Machine (SVM) [26], k-Nearest Neighbor (kNN) [26], and Extreme Gradient Boosting (XGBoost) [31].

For the SVM classifier we use two different kernels: a linear kernel, where the parameter C has been obtained through a grid search process with $C \in [10^{-2}, 10^{-1}, \ldots, 10^{10}]$, and a Radial Basis Function (RBF) kernel, where the parameters C and $gamma$ (γ) have been obtained through a gridsearch process with $C \in [10^{-2}, 10^{-1}, \ldots, 10^{10}]$ and $\gamma \in [10^{-9}, 10^{-8}, \ldots, 10^{3}]$.

The best C obtained for linear kernel was 0.01 and for RBF kernel the best C obtained was 10.0 with a γ of 0.001. For kNN classifier, we use a $k = 1$ and for XGBoost the learning rate (lr) was 0.1, maxdepth (md) was 3 and the number of estimators (ne) was 100.

Table 4 summarizes the results for each round of experiments, including the shallow classifiers proposed at this round. The best average accuracy achieved was 0.94 using SVM with RBF kernel, with a standard deviation of 0.0065 and a variance of 0.00003.

The ROC curve is presented in Fig. 9. Additionally, in Fig. 10 we also provide the learning curve for the SVM with RBF Kernel

Analyzing the learning curve, it is possible to observe that the training score is around the maximum and the validation score could be increased with more training samples. This observation led us to the next round of experiments, where we evaluate the transfer learning combined with SVM RBF performance in DSTokExt Dataset, which is an extended version of DSTok dataset proposed by Tokuda et al. [7].

4.9. Round #5: ResNet-50 bottleneck features with SVM over DSTokExt

The transfer of ResNet-50 convolutional layers trained on ImageNet dataset to CG image detection problem provided results comparable to the best literature methods. Furthermore, as showed in Round #4, the

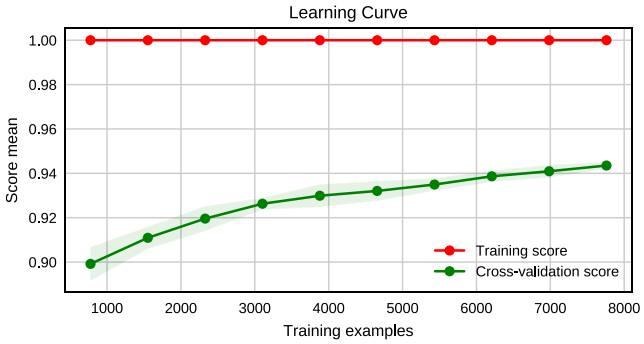


Fig. 10. SVM Learning Curve.

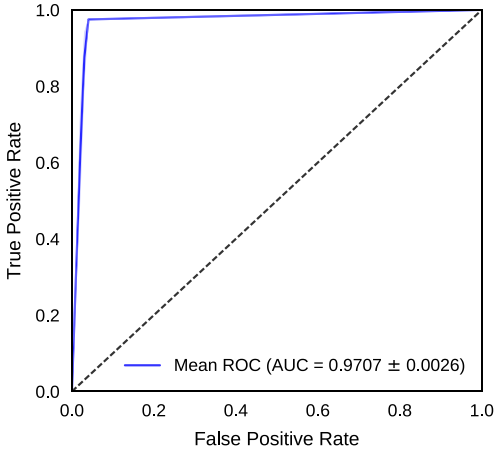


Fig. 11. ROC curve for the best result in DSTokExt dataset using ResNet-50 + SVM RBF Kernel.

learning curve of ResNet 50 + SVM RBF suggests that increasing the number of samples could improve the model accuracy.

At this round of experiments, we use an extended version of DSTok dataset, named DSTokExt and described in Section 4.2, to improve methods accuracy. We used the best model (ResNet-50 + SVM RBF) obtained in Round #4 and performed a gridsearch to find the parameters C and $gamma$ of the SVM RBF classifier with $C \in [10^{-2}, 10^{-1}, \ldots, 10^{10}]$ and $\gamma \in [10^{-9}, 10^{-8}, \ldots, 10^{3}]$. The best C was 10.0 with a γ of 0.001, the same values obtained in Round #4. The average accuracy achieved was 0.97 with an standard deviation of 0.003 and a variance of 6.85E−06. The ROC curve is depicted in Fig. 11. The area under the curve (AUC) is 0.97 and Table 5 presents the accuracy for each fold.

Fig. 12 presents the learning curve for this round, with the red curve representing the training score, green curve representing the average test score and the green shadow representing the standard deviation across the folds. Again, as depicted in learning curve of Round #4, it is possible to observe that the validation score could still be increased with more training samples.

Fig. 13 shows a comparison of confusion matrix (left) and the normalized confusion matrix (right) obtained with ResNet-50 + SVM RBF on DSTok and DSTokExt datasets.

4.10. Round #6: Visualization of bottleneck features

As described in Section 3, our method takes advantage of transfer learning process to generate ResNet-50 bottleneck features, projecting the 150,528 input features (224 × 224 × 3 RGB values of the pixels of each image) in a lower-dimensional space of 2048 features. This process intends to generate a set of features with a better degree of separability,

Table 4
Summary of proposed approaches along 4 rounds of experiments.

Architecture	Train	Epochs	Transfer	Avg Acc	Std Dev	Variance
ResNet-50 + 2fc softmax	from scratch	200	no	0.76	0.035	9.81E−04
ResNet-50 + 2fc softmax	fine tune	200	yes	0.82	0.035	9.73E−04
ResNet-50 + 2fc softmax	pre-train top + fine tune	200	yes	0.92	0.011	9.79E−05
ResNet-50 + kNN	$k=1$		yes	0.89	0.006	4.41E−05
ResNet-50 + XGBoost	$lr=0.1$, $md=3$, $ne=100$		yes	0.90	0.007	3.56E−05
ResNet-50 + SVM Linear	$C=0.01$		yes	0.92	0.007	4.39E−04
ResNet-50 + SVM RBF	$C=10$, $\gamma=0.001$		yes	0.94	0.007	3.38E−05

Table 5
Accuracy by fold of ResNet-50 + SVM RBF over DSTokExt.

Fold	Accuracy
0	0.97
1	0.98
2	0.97
3	0.97
4	0.96
Average	0.97

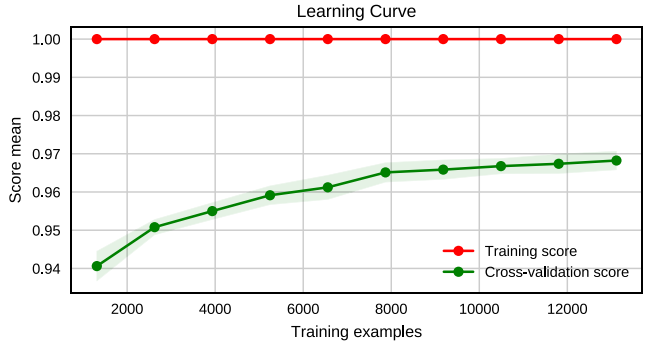


Fig. 12. SVM Learning Curve over DSTokExt dataset.

which could allow the top classifier to achieve a higher classification accuracy.

To evaluate if the bottleneck features would, in fact, produce the desired boost in classification accuracy, we applied the t-Distributed Stochastic Neighbor Embedding (t-SNE) [33] dimensionality reduction technique to visualize our high-dimensional features. We projected the 150,528 input features and the 2,048 bottleneck features in 2D, and plot them as points colored according to their class, as depicted in Fig. 14 for the images in DSTok dataset and Fig. 15 for the images in DSTokExt dataset. Green circles represent CG samples while blue squares represent PG samples.

It is possible to observe in the figures that the operations performed by ResNet-50 convolutional layers projected the raw pixels into a better separable feature space.

4.11. Round #7: comparative analysis

Along five rounds of experiments, we exposed how transfer learning can be used to take advantage of a DNN trained for object recognition task to generate discriminative features for CG images. These features can be used to train accurate classifiers for CG detection problem. The best accuracy of 0.97 was achieved using bottleneck features with an SVM classifier with RBF kernel in an extended version of DSTok dataset (containing DSTok images plus additional images).

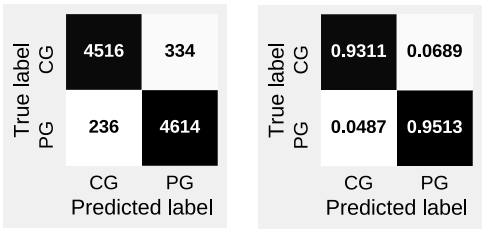
In Tokuda et al. [7], the authors present an extensive comparison of several literature approaches dedicated to solve the problem of detecting CG and PG images. The main characteristics of each method investigated by the authors are reported in Table 6.

Additionally, we included the characteristics of all methods proposed in this work: (1) ResNet-50 trained from Glorot uniform initialization (DNN1); (2) ResNet-50 fine-tuned from ImageNet initialization (DNN2); (3) ResNet-50 fine-tuned from ImageNet initialization and pre-trained softmax layer (DNN3); (4) ResNet-50 bottleneck features with kNN (DNN4); (5) ResNet-50 bottleneck features with XGBoost (DNN5); (6) ResNet-50 bottleneck features with SVM Linear (DNN6); and (7) ResNet-50 bottleneck features with SVM RBF (DNN7).

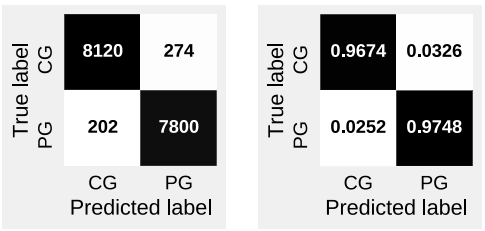
Considering that our experimental protocol is exactly the same one adopted by Tokuda et al. [7], we compared our method with other literature methods. Table 7 presents these results. From the table, we see that the accuracies of literature methods have a large range of values going from 0.51 (lowest) to 0.97 (highest). Proposed method DNN7 overcomes all literature methods based on raw and simple features and it is better than FUS4 proposed by Tokuda et al. [7]. This fact shows the expression power of transfer learning approach in features extraction process.

Table 8 show the results when comparing DNN7 against all the methods in Tokuda et al. [7], but now using DSTokExt. Here we also report execution time.

When increasing the number and complexity of images in dataset, most of the methods present a decay in accuracy. The exception is FUS4, which keeps result stable with an accuracy of 0.97. By other hand, proposed method increases its accuracy, going from 0.94 to 0.97 and also, as previously depicted in Fig. 12, present a learning rate curve not stable, meaning that the method can still be improved with more samples. Proposed method also has two main advantages when compared against FUS4: the absence of laborious hand-craft feature extraction work and it is extremely faster (around 155 times faster).

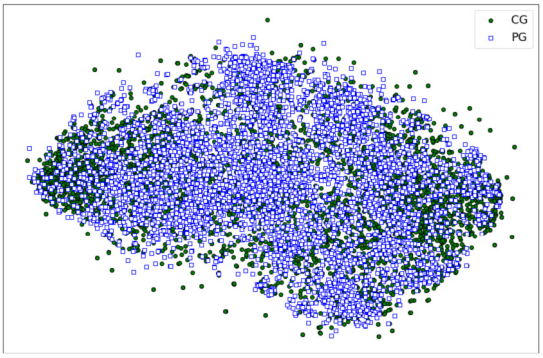


(a) DSTok dataset.

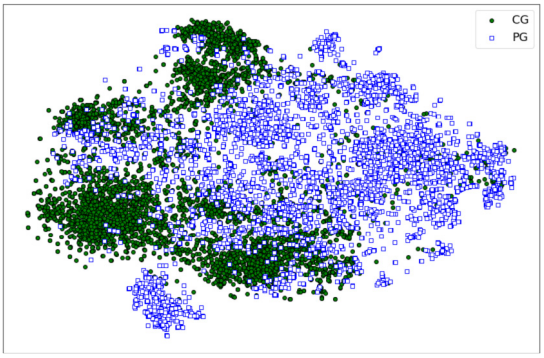


(b) DSTokExt dataset.

Fig. 13. Confusion matrix of the SVM classifier.

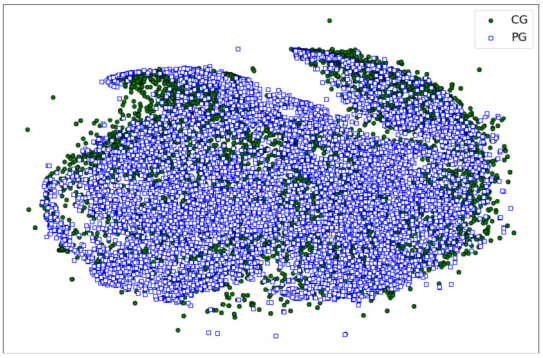


(a) Raw image pixels.

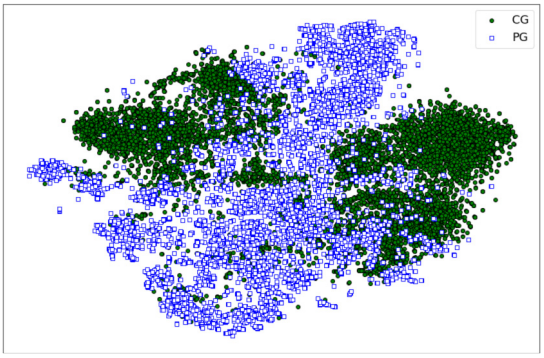


(b) Bottleneck features.

Fig. 14. t-SNE visualization of DSTok dataset using (a) raw image pixels and (b) ResNet-50 bottleneck features.



(a) Raw image pixels.



(b) Bottleneck features.

Fig. 15. t-SNE visualization of DSTokExt dataset using (a) raw image pixels and (b) ResNet-50 bottleneck features.

Table 6
Methods evaluated by et al.Tokuda [7] and methods proposed here. For each of one of the methods, it is shown the identifiers, the main concepts and the related features used by the methods.

Method	Main Concept	Feature
Li [34]	Second order differences	Edges/Texture
LSB [35]	Camera noise	Acquisition
LYU [36]	Wavelet transform	Edges/Texture
POP [37]	Interpolator predictor	Acquisition
BOX [38]	Boxes counting	Auto-similarity
CON [39]	Contourlet transform	Edges/Texture
CUR [40]	Curvelet transform [40]	Edges/Texture
GLC [41]	Cooccurrence matrix	Texture
HOG [42]	Histogram of oriented grads	Shape
HSC [43]	Histogram of shearlet coeff	Curves
LBP [44]	Local binary patterns	Edges/Texture
SHE [45]	Shearlet transform	Edges/Texture
SOB [46]	Sobel operator	Edges
FUS1 [7]	Concatenation	Combination
FUS2 [7]	Simple voting	Combination
FUS3 [7]	Weighted voting	Combination
FUS4 [7]	Meta-classification	Combination
DNN1	Deep CNN + Softmax (from scratch)	Raw image pixels
DNN2	Deep CNN transfer + Softmax (from ImageNet weights)	Raw image pixels
DNN3	Deep CNN transfer + Softmax (fine-tuning)	Raw image pixels
DNN4	Deep CNN transfer + kNN	Raw image pixels
DNN5	Deep CNN transfer + XGBoost	Raw image pixels
DNN6	Deep CNN transfer + SVM Linear	Raw image pixels
DNN7	Deep CNN transfer + SVM RBF	Raw image pixels

Table 7
Comparison among approaches for distinguishing CGs and PGs in DSTok dataset. Table is sorted from highest to lowest average accuracy. For each of the methods, it is shown the number of dimensions of the feature space (m), the average accuracy for each class and the variance.

Method	m	Average accuracy	Variance
FUS4	13	0.97	6.06E−04
FUS3	13	0.96	3.86E−04
FUS2	13	0.95	2.82E−04
DNN7	**150528**	**0.94**	**3.38E−05**
FUS1	4011	0.93	9.60E−02
Li	144	0.93	8.27E−05
DNN6	**150528**	**0.92**	**4.39E−04**
DNN3	**150528**	**0.92**	**9.79E−05**
LYU	216	0.92	2.26E−04
DNN5	**150528**	**0.90**	**3.56E−05**
CON	696	0.90	3.03E−04
DNN4	**150528**	**0.89**	**4.41E−05**
LBP	78	0.87	3.68E−04
DNN2	**150528**	**0.82**	**9.73E−04**
CUR	2328	0.80	9.39E−04
HSC	96	0.80	6.23E−04
DNN1	**150528**	**0.76**	**9.81E−04**
HOG	256	0.74	5.20E−04
SHE	60	0.71	7.84E−04
LSB	12	0.66	7.53E−04
GLC	12	0.63	1.01E−03
POP	12	0.57	5.95E−04
BOX	3	0.55	1.45E−03
SOB	150	0.55	1.05E−03

Table 8
Comparison among approaches for distinguishing CGs and PGsin DSTokExt. Table is sorted from highest to lowest average accuracy. For each of the methods, it is shown the number of dimensions of the feature space (m), the average accuracy for each class and the variance.

Method	m	Average accuracy	Variance	Time (s)
DNN7	**150,528**	**0.97**	**6.85E−06**	915
FUS4	13	0.97	2.58E−06	142,060
FUS1	4011	0.92	2.95E−05	102,723
FUS3	13	0.90	1.20E−05	125,783
FUS2	13	0.89	1.47E−05	122,783
LYU	216	0.86	2.98E−05	38,914
GLC	12	0.86	2.45E−05	2889
CON	696	0.85	1.76E−06	21,449
Li	144	0.85	7.71E−05	23,997
LBP	78	0.84	4.09E−05	2894
HOG	256	0.84	4.49E−05	22,143
CUR	2328	0.81	4.71–05	37,158
SHE	60	0.78	1.20E−05	5171
HSC	96	0.75	4.29E−05	1251
LSB	12	0.66	2.36E−04	559
SOB	150	0.58	6.67E−05	30,148
POP	12	0.53	2.15E−05	5728
BOX	3	0.51	7.10E−09	380

4.12. Round #8: Evaluation of method robustness against image processing operations

In a real world scenario, images can be submitted to different image processing operations as compression, noise addition, blur, etc. At this round of experiment, we evaluate the robustness of the proposed method against different processing operations. The results are computed on DSTokExt. Apart from the additional preprocessing steps, the evaluation protocol was identical to the one described in Section 4.9. Using OpenCV[7] to modify DSTokExt images, we examined the impact of different operations:

- **JPEG compression:** we re-compress original images with JPEG quality levels of 50, 70, 80 and 90 to check method robustness against compression;

[7] https://opencv.org.

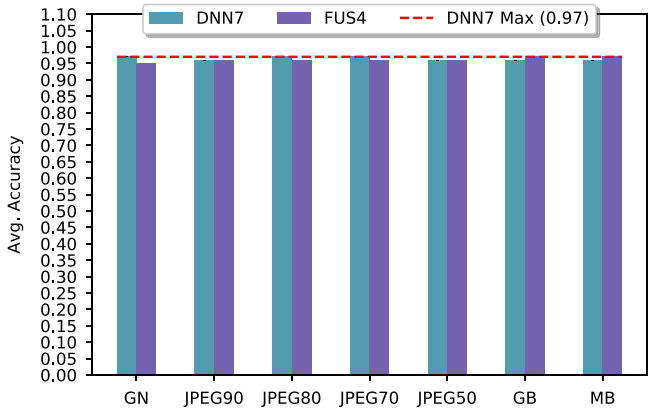


Fig. 16. Average accuracy for each variation of DSTokExt dataset.

- **Gaussian Noise (GN) Addition:** at each image in the dataset, we added a random amount of Gaussian noise. The amount of noise varies between 1% and 5%.
- **Blur:** we use a Gaussian (GB) and a Median (MB) filter to blur the image. Kernel size is $k = 5$ and, for Gaussian kernel, σ is calculated according Eq. (6)

$$\sigma = 0.3 \times ((k-1) \times 0.5 - 1) + 0.8 \tag{6}$$

Fig. 16 depicts the average accuracy for each processed dataset.

By Fig. 16 we can observe that DNN7 and FUS4 are very robust against image processing operations. Methods performance decay no more than 2% in average accuracy. Its interesting to observe that DNN7 performs better in noise and compression operations while FUS4 performs better in blur operations. Given that CNN use a sequence of convolutional operations, similar to filters, to extract specialized features, this small accuracy decay on datasets with blur operations was expected.

4.13. Round #9: Evaluation of Method Robustness in a Cross-Dataset Scenario

To evaluate the generalization of proposed method with respect to the training data, we followed an experimental design similar to other literature works [47,48]. We performed a cross-dataset experiment using DSTok images and the additional images used to create DSTokExt, called here DSTokExt*, which is composed by difference between DSTokExt and DSTok (DSTokExt* = DSTokExt - DSTok). In the first scenario (cross-dataset 1) we train DNN7 architecture using DSTok and test trained model using DSTokExt*, obtaining an average accuracy rate of 0.92. In the second scenario (cross-dataset 2) we train DNN7 architecture using DSTokExt* and test trained model using DSTok, achieving an average accuracy rate of 0.88. Fig. 17 shows the ROC curve for this experiment, and compare it with ROC curve of DSTokExt experiment. Obtained results indicate that the proposed method offers a degree of generalization to images from different sources.

5. Conclusions and research directions

In this paper we have presented a new method for CG images detection using a deep convolutional neural network model based on ResNet-50 and transfer learning concepts. After a simple pre-processing, each image in our dataset is fed into our deep CNN model and, as result, we obtain a 2048 dimension feature vector, here called bottleneck features. Exploring different approaches looking for achieving the most effective problem solution, we evaluate different approaches, since train ResNet-50 architecture from scratch (just changing the 1000fc softmax

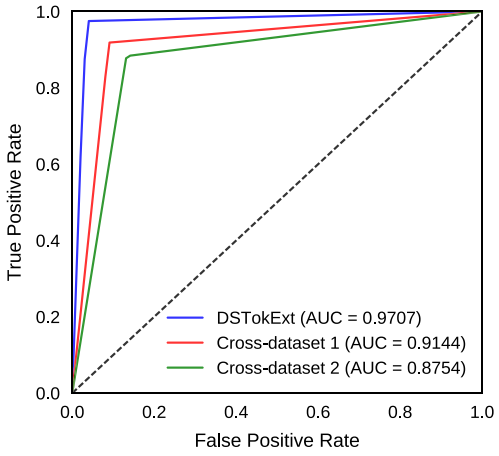


Fig. 17. ROC curve of cross dataset experiment.

from original architecture to a 2fc softmax in top layer), using our dataset, for CG detection process, until full transfer learning, where ImageNet weights for ResNet-50 are totally frozen in a way to produce bottleneck features, which are used to train different machine learning classifiers to detect if an image is, or not, produced by computer graphics methods.

Conducting different rounds of experiments, we evaluate the efficiency and effectiveness of using a deep CNN architecture proposed for an object recognition task in a CG detection problem, where we are looking for distinguishing between a CG and a PG image, involving different kinds of objects and context. Results showed that proposed approach performs as good as the top state-of-the-art methods in the same dataset, achieving more than 0.97 accuracy rate. These results highlight three main advantages of the proposed method: (i) no requirement for hand-craft feature extraction, (ii) robustness against image processing operations and, (iii) extremely lower execution time when compared against state-of-the-art method with similar accuracy.

In special, in Round #6 of experiments (Section 4.10) we showed for DSTok dataset, using t-SNE dimensionality reduction, the expression power of bottleneck features generated by ResNet-50 transfer layers, which increases the classes separability when compared against raw input features. The same behavior is kept in DSTokExt with more and different kinds of images.

Furthermore, it is important to realize that, as showed in Section 4.9, even with a extended dataset, the learning curve from the SVM classifier it is still not stable, which suggest that training score is not still around the maximum. Since deep learning models needs an astonishing number of images to achieve a satisfactory accuracy, we conclude that if we keep increasing the number of images, it can lead to an accuracy even better.

A limitation of this method is its difficulty in dealing with CG images with a high degree of realism. We conducted an experiment with two datasets involving images very similar and with high degree of realism, one proposed by Holmes et al. [1] and a second one proposed by Carvalho et al. [49]. This left a door open for the improvement of this technique or development of a new one that could cope with this difficult scenario.

As research directions, our proposal is to explore different architectures for bottleneck features extraction and to perform fusion of these architectures in a way to construct an ensemble of deep architectures.

Acknowledgments

The authors would like to thank the financial support of IFSP-Campinas, FAPESP (grant 2017/12631-6), Fapesp DéjàVu (grant 2017/12646-3) and CNPq (grants 302923/2014-4, 313152/2015-2 and 423797/2016-6). We also would like to thank the authors Tokuda et al. [7] who helped us with dataset acquirement and we gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

References

[1] O. Holmes, M.S. Banks, H. Farid, Assessing and improving the identification of computer-generated portraits, ACM Trans. Appl. Perception 13 (2) (2016) 12.
[2] V. Schetinger, M.M. Oliveira, R. da Silva, T.J. Carvalho, Humans are easily fooled by digital images, Comput. Graph. 68 (2017) 142–151.
[3] K. Schulten, A.C. Brown, Evaluating sources in a "post-truth" world: Ideas for teaching and learning about fake news, The New York Times, 2017, http://tinyurl.com/h3w7rp8 Accessed on November 17th.
[4] R. Keyes, The Post-Truth Era: Dishonesty and Deception in Contemporary Life, St. Martin's Press, 2004.
[5] F. Davey-Attlee, I. Soares, The Fake News Machine. Inside a Town Gearing Up for 2020, CNN, http://money.cnn.com/interactive/media/the-macedonia-story/, Accessed on November 17th 2017.
[6] S. Shane, Mystery of Russian fake on facebook solved, by a Brazilian, The New York Times https://www.nytimes.com/2017/09/13/us/politics/russia-facebook-election.html, accessed on November 17th 2017.
[7] E. Tokuda, H. Pedrini, A. Rocha, Computer generated images vs. digital photographs: A synergetic feature and classifier combination approach, J. Vis. Commun. Image Represent. 24 (8) (2013) 1276–1292.
[8] D. Dang-Nguyen, G. Boato, F.G.B.D. Natale, Discrimination between computer generated and natural human faces based on asymmetry information, in: European Signal Processing Conference, 2012, pp. 1234–1238.
[9] D. Dang-Nguyen, G. Boato, F.G.B.D. Natale, Identify computer generated characters by analysing facial expressions variation, in: IEEE International Workshop on Information Forensics and Security, 2012, pp. 252–257.
[10] H. Farid, M.J. Bravo, Perceptual discrimination of computer generated and photographic faces, Digit. Investigat. 8 (2012) 226–235.
[11] V. Conotter, E. Bodnari, G. Boato, H. Farid, Physiologically-based detection of computer generated faces in video, in: IEEE International Conference on Image Processing, 2014, pp. 248–252.
[12] Y. Bengio, et al., Learning deep architectures for AI, Found. Trends® Mach. Learning 2 (1) (2009) 1–127.
[13] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016 http://www.deeplearningbook.org.
[14] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[16] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances In Neural Information Processing Systems, 2012, pp. 1097–1105.
[17] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556, 2014.
[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[20] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
[21] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Advances in Neural Information Processing Systems, 2014, pp. 3320–3328.
[22] H. Farid, Creating and detecting doctored and virtual images: implications to the child pornography prevention act TR2004-518, 2004.
[23] D.-T. Dang-Nguyen, G. Boato, F.G. De Natale, 3D-model-based video analysis for computer generated faces identification, IEEE Trans. Inform. Forensics Secur. 10 (8) (2015) 1752–1763.
[24] D.Q. Tan, X.J. Shen, J. Qin, H.P. Chen, Detecting computer generated images based on local ternary count, Pattern Recognit. Image Anal. 26 (4) (2016) 720–725.
[25] Computer Vision Laboratory - Columbia University, 2017 Natural image library, http://www.cs.columbia.edu/CAVE/, accessed on May 19th.
[26] C.M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag, 2006.
[27] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv preprint arXiv:1502.03167, 2015.
[28] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, DeCAF: A deep convolutional activation feature for generic visual recognition, in: International Conference on Machine Learning, Vol. 32, 2014, pp. 647–655.
[29] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision, Springer, 2014, pp. 818–833.
[30] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: Integrated recognition, localization and detection using convolutional networks, in: arXiv preprint arXiv:1312.6229, 2013.

[31] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, arXiv preprint arXiv:1603.02754, 2016.
[32] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: International Conference on Artificial Intelligence and Statistics, Vol. 9, 2010, pp. 249–256.
[33] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.
[34] W. Li, T. Zhang, E. Zheng, X. Ping, Identifying photorealistic computer graphics using second-order difference statistics, in: International Conference on Fuzzy Systems and Knowledge Discovery, Vol. 5, 2010, pp. 2316–2319.
[35] T.-T. Ng, S.-F. Chang, Identifying and prefiltering images distinguishing between natural photography and photorealistic computer graphics, IEEE Signal Process. Mag. 26 (2) (2009) 49–58.
[36] S. Lyu, H. Farid, How realistic is photorealistic? IEEE Trans. Signal Process. 53 (2) (2005) 845–850.
[37] H.F.A.C. Popescu, Exposing digital forgeries in color filter array interpolated images? IEEE Trans. Signal Process. 53 (10) (2005) 3948–3959.
[38] L. Liebovitch, T. Toth, A fast algorithm to determine fractal dimensions by box counting, Phys. Lett. A 141 (1989) 386–390.
[39] M. Do, M. Vetterli, Contourlets: a directional multiresolution image representation, in: IEEE International Conference on Image Processing, 2002, pp. 357–360.
[40] E. Candes, D. Donoho, Curvelets: A Surprisingly Effective Nonadaptive Representation for Objects with Edges, Vanderbilt University Press, 2000.
[41] R. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, IEEE Trans. Syst. Man Cybern. 3 (6) (1973) 610–621.
[42] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 886–893.
[43] W. Schwartz, R. da Silva, L. Davis, H. Pedrini, A novel feature descriptor based on the shearlet transform, in: IEEE International Conference on Image Processing, 2011, pp. 1053–1056.
[44] T. Ojala, M. Pietikainen, T. Maenpaa, (2001) A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification, in: International Conference on Advances in Pattern Recognition, pp. 399–408.
[45] G. Kutyniok, W.-Q. Lim, Compactly supported shearlets are optimally sparse, J. Approx. Theory (2011) 1564–1589.
[46] R. Gonzalez, R. Woods, Digital Image Processing, Prentice-Hall, 2007.
[47] T.J.d. Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, A.d.R. Rocha, Exposing digital image forgeries by illumination color classification, IEEE Trans. Inform. Forensics Secur. 8 (7) (2013) 1182–1194.
[48] A. Rocha, T. Carvalho, H.F. Jelinek, S. Goldenstein, J. Wainer, Points of interest and visual dictionaries for automatic retinal lesion detection, IEEE Trans. Biomed. Eng. 59 (8) (2012) 2244–2253.
[49] T. Carvalho, E.R.S. de Rezende, M.T.P. Alves, F.K.C. Balieiro, R.B. Sovat, Exposing computer generated images by eye's region classification via transfer learning of VGG19 CNN, in: IEEE International Conference On Machine Learning And Applications, ICMLA, 2017.