Satellite Image Forgery Detection and Localization Using GAN and One-Class Classifier

Sri Kalyan Yarlagadda; School of Electrical and Computer Engineering, Purdue University; West Lafayette, IN, USA
David Güera; School of Electrical and Computer Engineering, Purdue University; West Lafayette, IN, USA
Paolo Bestagini; Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano; Milano, Italy
Fengqing Maggie Zhu; School of Electrical and Computer Engineering, Purdue University; West Lafayette, IN, USA
Stefano Tubaro; Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano; Milano, Italy
Edward J. Delp; School of Electrical and Computer Engineering, Purdue University; West Lafayette, IN, USA

Abstract

Current satellite imaging technology enables shooting high-resolution pictures of the ground. As any other kind of digital images, overhead pictures can also be easily forged. However, common image forensic techniques are often developed for consumer camera images, which strongly differ in their nature from satellite ones (e.g., compression schemes, post-processing, sensors, etc.). Therefore, many accurate state-of-the-art forensic algorithms are bound to fail if blindly applied to overhead image analysis. Development of novel forensic tools for satellite images is paramount to assess their authenticity and integrity. In this paper, we propose an algorithm for satellite image forgery detection and localization. Specifically, we consider the scenario in which pixels within a region of a satellite image are replaced to add or remove an object from the scene. Our algorithm works under the assumption that no forged images are available for training. Using a generative adversarial network (GAN), we learn a feature representation of pristine satellite images. A one-class support vector machine (SVM) is trained on these features to determine their distribution. Finally, image forgeries are detected as anomalies. The proposed algorithm is validated against different kinds of satellite images containing forgeries of different size and shape.

Introduction

Ever since the birth of the Internet, the accessibility to images has become easier over time. Internet has become an affordable and effective platform for distributing one’s own images. User friendly software like Photoshop and Gimp can be used to generate a variety of image manipulations such as inpainting, copy-forge, splicing, etc. A combination of the above two scenarios is a perfect environment for producing doctored images, which when treacherously used can cause substantial damage. Therefore, it is of paramount importance to develop forensic methods to validate the integrity of an image. For this reason, over the years, the forensic community has developed several techniques for image authenticity detection and integrity assessment [1, 2, 3].

In addition to photographs captured with cameras and smartphones, other types of imagery are starting to be circulated, posing new problems for the forensic community. Indeed, current satellite imaging technology enables shooting high-resolution pictures of the ground. Due to the increased availability of satellites equipped with imaging sensors, overhead images are becoming popular. It is now possible to easily gather overhead images of the ground through public websites [4] and to buy custom image sets of specific locations and times. As any other kind of digital images, overhead pictures can also be easily forged. One question that needs to be addressed is whether these images are authentic. Cases of malicious overhead image manipulations have already been reported [5], [6]. The development of forensic methods tailored to the analysis of this type of imagery is considered to be urgent.

However, common image forensic techniques are often developed for consumer cameras, which strongly differ in their nature from satellite sensors (e.g., compression schemes, post-processing, sensors, etc.). Therefore, many accurate state-of-the-art forensic algorithms are bound to fail if blindly applied to overhead image analysis. Development of novel forensic tools for satellite images is paramount to assess their authenticity and integrity.

To fill the lack of ad-hoc forensic techniques for satellite images, the authors of [7] proposed an active method based on watermark embedding. Watermarks can then be exploited to detect possible doctored image regions. Unfortunately, this method can only be used if watermark is inserted at image inception time. More recently, the authors of [8] proposed a passive forensic method for overhead image analysis. This algorithm is based on machine learning techniques, but it can only localize image regions that have been inpainted. To the best of our knowledge, no specific algorithms for other kinds of satellite image forgeries have been proposed in the literature.

In this paper, we propose an algorithm for satellite image forgery detection and localization. Specifically, we consider the situation in which pixels within a region of a satellite image are replaced to add or remove an object from the scene. Our algorithm works under the assumption that no forged images are available for training. Using a generative adversarial network (GAN), we learn a feature representation of pristine satellite images. A one-class support vector machine (SVM) is trained on these features to determine their distribution. Finally, image forgeries are detected as anomalies.

To validate the proposed method, we built a custom dataset of forged satellite images using different forgery sizes. Results in terms of forgery detection and localization are presented. Moreover, as the proposed algorithm works by analyzing images patch-wise, it is possible to strongly parallelize it to keep processing time at bay.
Problem Definition and Background

In this section we describe the problem formulation and notation used throughout the entire paper. Following this, we provide some background concepts on autoencoders and convolutional neural networks.

Problem formulation

Consider an image $I$ coming from a satellite. We can represent the pixel integrity associated with the image $I$, as a binary mask $M$ of the same size as the image in pixels. Each entry of $M$ is a binary label 0 or 1, such that a pixel belonging to a forged area is assigned the label 0 and a pixel from an untampered area is assigned a 1. As forgery, in this paper we consider an object insertion/removal through a copy-paste operation from a different source. This means that forged pixels do not belong to a satellite image but come from a different device (e.g., the picture of a plane acquired with a normal camera). Figure 1 shows an example of a pristine satellite image and a completely white (i.e., label 1) mask, as well as a forged image with the respective black and white mask localizing the forgery. Within this setup, our goal is twofold:

- **Tampering Detection**: given an image, detect whether it is pristine or forged.
- **Tampering Localization**: given a forged image, detect which are the forged pixels.

These two tasks can be accomplished by computing $\hat{M}$ (i.e., an estimate of $M$). If $\hat{M}$ contains any entry different from 1, the image is detected as forged. Entries of $\hat{M}$ whose values are 0 represent forged pixel positions.

Related Work

In this section we present a brief summary of autoencoders that are needed to follow this paper. For a thorough review, we recommend the readers to refer to Chapter 14 of [9].

Autoencoders are neural networks that are trained to attempt to obtain an output equal to the input through a set of linear and non-linear operations that expand or reduce data dimensionality at some point in the network. They consist of two parts: the encoder $A_e$ and decoder $A_d$. The output of the encoder is called feature vector or hidden representation, and we represent it as $h$. In this paper we work with autoencoders where the dimensionality of $h$ is lower than the dimensionality of the input. This kind of autoencoders are known as undercomplete autoencoders. From now on, whenever we refer to autoencoders we refer to undercomplete autoencoders. Such an architecture forces the autoencoder to capture a salient representation of the input in a reduced dimensionality space.

Autoencoders are trained by minimizing through iterative procedures a loss value defined as

$$L = \mathcal{L}(x, A_d(A_e(x))),$$

(1)

where $\mathcal{L}(\cdot, \cdot)$ is the loss function computing some distance between its two arguments, $x$ is the autoencoder input, and $A_d(A_e(x))$ is the output. In the special case where $\mathcal{L}$ is the mean squared error (MSE) loss, the autoencoder learns to perform a generalized non linear principal component analysis (PCA).

Figure 1: Example of pristine (a) and forged (c) images $I$ associated to their binary forgery masks $M$ (b) and (d), respectively.

In this paper we design our autoencoders using Convolutional Neural Networks (CNNs). CNNs have proven to be very successful in a variety of computer vision tasks such as object recognition [10], object detection [11], etc. They came to light in 2012 [12] when they produced stunning results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [13]. Since then there has been an explosion in the application of CNNs to various other computer vision tasks and often resulting in new state of the art results.

When it comes to image forensics, the use of CNNs has been on the rise. Many forensic problems deal with non-linear and often difficult to model pipelines. Therefore, CNNs have proven to be successful in this area. The first works using CNNs in this area were focused on steganalysis [14, 15, 16, 17]. Strictly concerning multimedia forensics, many other tasks have been considered. As a few examples, [18] deals with median filtering detection, [19] proposes the use of a constrained convolutional layer for forgery detection. In [20, 21, 22, 23], the problem of camera model identification and its possible forgeries is explored. Double JPEG compression is also considered in [24, 25].

Convolutional neural networks usually consist of operations such as convolutions, batch normalization [26], local pooling, thresholding and non linear activations. These operations are stacked together and are tuned by minimizing a cost function at the output. Following, we describe some of the most commonly used layers:

- **Convolutional**: the input of this layer is convolved with a bank of filters whose response is learned through training. The input is typically a 3D structure, i.e., it has two spatial coordinates plus depth (e.g., an RGB image). The output is known as feature map.
- **Max pooling**: given an input $x$, a sliding window is used to extract the maximum value over it.
it is composed by the following steps: particular, the pipeline of our method is reported in Figure 2, and detect object insertion / deletion attacks in satellite images. In anomaly with respect to the learned model.

- Batch Normalization: given an input $x$, this layer normalizes $x$ by imposing zero mean and unit variance. Details about this are explained in [26]
- Deconvolutional: this layer is the transpose of a convolutional layer. The output is obtained by convolving a zero-padded version of the input with a filter bank learned through training. The spatial dimensions of the output are greater than that of the input.

**Method**

In this section we elaborate on the details of our method to detect object insertion / deletion attacks in satellite images. In particular, the pipeline of our method is reported in Figure 2, and it is composed by the following steps:

- The color image under analysis is split into patches (either overlapping or not) of size $64 \times 64$ pixels.
- A adversarially trained autoencoder encodes the patches into a low dimensional representation called feature vector $h$.
- A one-class SVM fed with $h$ is used to detect forged patches as anomalies with respect to features distribution learned from pristine patches.
- Once all patches are classified, a label mask for the entire image is obtained by grouping together all the patch labels.

The rationale behind the proposed solution is that autoencoders are able to capture a reduced dimensionality representation of the input data, still retaining important characteristic information, as shown in [27, 28] for forensic purposes. Therefore, by training an autoencoder only on pristine data, we expect it to learn to extract features specific of original satellite images. Conversely, when it is tested on forged data, the extracted features should be strongly different from those obtained from pristine images. A one-class SVM trained on pristine features only can then be used to discriminate between features coming from pristine and forged images. Following, we report a detailed explanation of each step of the proposed pipeline.

**Patch Extraction**

The given image $I$ is split into regular patches $P_k$, where $k \in [1, K]$ is the patch index, and $K$ is the total amount of patches. Patches can be either overlapped or not depending on the selected trade-off between detection accuracy and computational complexity (i.e., overlapping leads to more patches to analyze but more accurate results).

**Feature Extraction**

Every patch $P_k$ is fed to the autoencoder $A$ which consists of two parts: the encoder $A_e$ and decoder $A_d$. Both $A_e$ and $A_d$ are made of convolutional and deconvolutional neural networks respectively. They are symmetric in terms of the number of layers. The architecture of $A_e$ has been selected following the same rationale of [29] and it is as follows:

- **conv1**: convolution layer with 16 filters each of size (6,6) with stride 1.
- **conv2**: convolution layer with 16 filters each of size (5,5) with stride 2.
- **conv3**: convolution layer with 32 filters each of size (4,4) with stride 2.
- **conv4**: convolution layer with 64 filters each of size (3,3) with stride 2.
- **conv5**: convolution layer with 128 filters each of size (2,2) with stride 2.

All convolutional layers except $conv5$ are followed by batch normalization. All the convolution layers are activated using a linear function. The output of $conv5$ is the feature vector $h$, a 2048 dimensional vector and has a much lower dimension than that of the input which is 12288 dimensional. The architecture of $A_d$ is as follows:

- **deconv1**: deconvolution layer with 64 filters each of size (2,2) with stride 2.
- **deconv2**: deconvolution layer with 32 filters each of size (3,3) with stride 2.
- **deconv3**: deconvolution layer with 16 filters each of size (4,4) with stride 2.
- **deconv4**: deconvolution layer with 16 filters each of size (5,5) with stride 2.
- **deconv5**: deconvolution layer with 3 filters each of size (6,6) with stride 1.

Every deconvolutional layer is followed by batch normalization except $deconv5$. $deconv5$ has a hyperbolic tangent activation where all other deconvolution layers have linear activations. The output of $deconv5$ is the output of the autoencoder. Once the autoencoder is trained on pristine image patches, we use it as feature extractor to compute the feature vector $h_k = A_e(P_k)$ from each image patch $P_k$.

Conventionally $A$ can be trained using stochastic gradient descent to minimize mean squared loss between input (i.e., $P_k$) and output (i.e., $A_d(A_e(P_k))$). However better results can be achieved when we follow an adversarial framework for training the autoencoder. In [30] the authors established a framework of min-max
adversarial game between two neural networks, namely the generator and discriminator, and such networks are called Generative Adversarial Networks. As shown in Figure 3, the discriminator aims to accurately discriminate between patches from real satellite images and patches created by the generator. The generator on the other hand aims to mislead the discriminator by trying to generate data closer and closer to the real one. Such frameworks have proven to be extremely effective.

The architecture of the discriminator $D$ we use is as follows:

- **conv1**: convolution layer with 16 filters each of size (5,5) with stride 1 followed by Leaky ReLU and batch normalization.
- **conv2**: convolution layer with 16 filters each of size (2,2) with stride 2.
- **conv3**: convolution layer with 32 filters each of size (4,4) with stride 1 followed by Leaky ReLU and batch normalization.
- **conv4**: convolution layer with 32 filters each of size (2,2) with stride 2.
- **conv5**: convolution layer with 64 filters each of size (2,2) with stride 2 followed by Leaky ReLU and batch normalization.
- **conv6**: convolution layer with 64 filters each of size (3,3) with stride 1 followed by Leaky ReLU and batch normalization.
- **fc1**: A 128-neuron fully connected dense layer followed by a Leaky ReLU activation.
- **fc2**: A single neuron followed by a sigmoid activation.

After training the autoencoder using the GAN strategy on pristine images only, the encoder $A_e$ is used to extract the feature vector $h_k$ from each patch $P_k$ under analysis.

**One-Class SVM**

The autoencoder $A$ is trained only on pristine patches and hence it learns to encode them very well. So when $A$ sees a patch containing a forgery, it encodes it quite differently. In order to capture this difference without any knowledge on forged data, we use a one-class SVM trained on feature vectors $h$ extracted from pristine images only. The used one class SVM learns pristine feature distribution. It then outputs a soft value which represents the likelihood of the feature vector $h$ under analysis being pristine. We define the soft mask $\tilde{M}$ as a matrix the same size of the image, where each entry contains the soft SVM output relative to the image patch in the same position. This soft mask $\tilde{M}$ can be used to obtain the final detection binary mask $\hat{M}$ by simply thresholding.

**Experimental Validation**

In this section we report the experimental validation of the proposed technique. We first discuss how we built the used dataset. We then provide details about the considered experimental setup for reproducible research. Finally, we show the achieved numerical results.

**Dataset**

We tested our algorithm using overhead images obtained from the Landsat Science program [31, 32]. The Landsat Science is a program run jointly by NASA [33] and the US Geological Survey (USGS) [34]. It was first launched in 1972 and has produced the longest, continuous record of Earth’s land surface as seen from Space. NASA is responsible for the remote sensing equipment, launching satellites and validating their performance. USGS operates the satellites and manages data reception, archiving and distribution. Since late 2008 these images have been made available free of charge. The Landsat Program obtains overhead images from a series of satellites. We have created our dataset $D$ using images from one satellite. $D$ consists of 130 color images each cropped at a resolution of $650 \times 650$ pixels. This dataset is further divided into three parts namely training $D_{\text{train}}$, validation $D_{\text{val}}$ and testing $D_{\text{test}}$.

Out of the 130 images, 30 of them are used to create patches for training and validation. Patches of size $64 \times 64 \times 3$ are extracted from every image with a patch stride of $32 \times 32$ generating a total of 10830 patches. Out of these patches, 20 percent have been used for validation and the remaining for training. So $D_{\text{train}}$ consists of 8664 patches and $D_{\text{val}}$ 2166 patches.

The remaining 100 images are used for creating $D_{\text{test}}$. Half of $D_{\text{test}}$ is used to generate forgeries, the remaining half is kept

![Image with small sized splicing](b) Forged mask

![Image with medium sized splicing](c) Forged mask

![Image with large sized splicing](e) Forged mask

Figure 4: Examples of forged images with forgeries of different sizes. Ground truth masks $M$ are also reported.
as pristine testing data. In order to create forged images, credible objects such as airplanes, clouds, etc. are spliced at random positions onto the 50 selected images from $D_{test}$. During the splicing operation, the size of spliced objects relative to the used analysis patch size is controlled. Therefore, we define three sizes namely:

- **Small** - Object size is smaller than the patch size (approximately 32 pixel per side).
- **Medium** - Object size is comparable to patch size (approximately 64 pixel per side).
- **Large** - Object size is larger than patch size (approximately 128 pixel per side).

Objects of each size are forged onto the 50 images at random positions to create 150 forged images, i.e., we have 50 images with Small objects spliced onto them ($D_{32}$), 50 images with Medium objects spliced onto them ($D_{64}$), and at last 50 images with Large objects spliced onto them ($D_{128}$). Examples of pristine images and forged images with different size forgeries are shown in Figure 4.

**Experimental Setup**

Our model consists of two important components, namely the Autoencoder $A$ and the one-class SVM. In this section we describe the policies we used to choose the best model and the various hyper parameters for each of these.

**Autoencoder**

The autoencoder is fed with patches from pristine images and its task is to capture their distribution. An autoencoder can be trained on its own but we choose to couple it with a Discriminator $D$ to further push its training. The job of $D$ is to be able to discriminate between patches produced by $A$ and the actual patches from pristine images. By coupling $A$ with $D$ we form a Generative Adversarial Network (GAN). We can judge the performance of $A$ using the mean squared error metric and the performance of $D$ by binary cross entropy. Both $A$ and $D$ are CNNs and, in order to choose the right CNN architecture, we use the following approach:

- The architecture of $D$ is fixed and it is described in the Feature Extraction section.
- For $A$, a variety of CNN architectures are tested and the one with the lowest mean squared error loss is chosen as the best model.

The various architectures tested to choose the best model for $A$ are detailed in Table 1.

**Training Strategy**

We adopt two different training strategies, With GAN and Without GAN.

- **Without GAN**: we train the autoencoder with patches from $D_{train}$ and $D_{val}$. We use the Adam optimizer for a total of 100 epochs. The model weights with the lowest MSE loss on $D_{val}$ over the 100 epochs are chosen as the final model weights.
- **With GAN**: we first train the autoencoder using patches from $D_{train}$ and $D_{val}$ using the Adam optimizer for 100 epochs. The weights corresponding to the lowest loss on $D_{val}$ over the 100 epochs are locked. We then use these weights to initialize the weights of the generator in the GAN. The GAN is then trained for 100 epochs with the SGD optimizer and a learning rate of 0.001 for the discriminator, and the Adam optimizer with a learning rate of 0.001 for the generator. The GAN training is carried out on batches of 128 patches. The weights with the lowest MSE loss are chosen for the final generator.

For the SVM, we used a radial basis function with $\gamma = 1/2048$ as kernel, and small value of nu-parameter ($\nu = 0.00001$).

**Results**

Using the With GAN training strategy, the mean squared error (MSE) loss over $D_{val}$ for the various architectures of the autoencoder are reported in Table 2. Architectures $A_2$, $A_3$ and $A_4$ have similar number of parameters (about 100k) while $A_1$ has almost a million parameters. Note that $A_1$, $A_2$ and $A_3$ perform similarly despite the huge difference in the number of parameters. Among all of them $A_4$ provides the lowest MSE and hence better patch reconstruction. Therefore, we decided to select architecture $A_4$ for our system.

In order to visualize the forged-vs-pristine discriminability power of feature vectors $h$ extracted from the proposed autoencoder, we applied the t-SNE algorithm [35]. T-SNE can be used for unsupervised feature dimensionality reduction in order to check whether it is possible to cluster some data. In our particular case, the feature vector $h$ is a high dimensional vector that is very difficult to visualize, whereas, by applying t-SNE we have some visual clues on feature behavior. Figure 5 shows the distribution of features in a reduced dimensionality space of three dimensions using t-SNE. It is possible to notice that patches not containing forged pixels have features that cluster together (i.e., red dots). Conversely, features belonging to patches containing forged pixels (i.e., blue dots) are spread in the three-dimensional space far from the pristine cluster. This confirms that the proposed feature vector is able to capture forged-vs-pristine information.

In order to evaluate forgery detection performance, we estimated the soft mask $\mathbf{M}$ for each image in the dataset. For each mask, we selected as pristine confidence the minimum $\mathbf{M}$ value (i.e., the SVM output associated to the least probable pristine patch). By thresholding this confidence score, we obtained a receiver operating characteristic (ROC) curve. Figure 6 shows some
Table 1: The four proposed autoencoder architectures (shown in columns). The number of filters for each convolutional layer, its size and the used stride is shown in parenthesis followed by the activation function. If no activation is specified for any layer its assumed to be linear activation.

<table>
<thead>
<tr>
<th>Encoder and Decoder Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{1e}$</td>
</tr>
<tr>
<td>Conv1 (16, (6,6), Stride 1) + ReLU</td>
</tr>
<tr>
<td>Conv2 (32, (5,5), Stride 2) + ReLU</td>
</tr>
<tr>
<td>Conv3 (64, (4,4), Stride 2) + ReLU</td>
</tr>
<tr>
<td>Conv4 (128, (3,3), Stride 2) + ReLU</td>
</tr>
<tr>
<td>Conv5 (256, (2,2), Stride 2) + ReLU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$A_{1d}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deconv1 (256, (2,2), Stride 2) + ReLU</td>
</tr>
<tr>
<td>Deconv2 (128, (3,3), Stride 2) + ReLU</td>
</tr>
<tr>
<td>Deconv3 (64, (4,4), Stride 2) + ReLU</td>
</tr>
<tr>
<td>Deconv4 (32, (5,5), Stride 2) + ReLU</td>
</tr>
<tr>
<td>Deconv5 (3, (6,6), Stride 1) + tanh</td>
</tr>
</tbody>
</table>

BN after each convolutional layer except conv5

Table 2: MSE of the various autoencoder architectures. The one with the lowest MSE loss is selected for the proposed method.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Trainable Parameters</th>
<th>MSE loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>997299</td>
<td>0.00131671</td>
</tr>
<tr>
<td>$A_2$</td>
<td>84547</td>
<td>0.00131675</td>
</tr>
<tr>
<td>$A_3$</td>
<td>124883</td>
<td>0.00130047</td>
</tr>
<tr>
<td>$A_4$</td>
<td>135939</td>
<td>0.00125511</td>
</tr>
</tbody>
</table>

Table 3: Detection results in terms of AUC for the different datasets. AUCs are reported in two different cases: autoencoder trained with or without the GAN. Best results are reported in italics.

<table>
<thead>
<tr>
<th>Forgery Size</th>
<th>AUC (without GAN)</th>
<th>AUC (with GAN)</th>
<th>AUC Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>0.784</td>
<td>0.797</td>
<td>+0.013</td>
</tr>
<tr>
<td>Medium</td>
<td>0.904</td>
<td>0.920</td>
<td>+0.016</td>
</tr>
<tr>
<td>Large</td>
<td>0.950</td>
<td>0.972</td>
<td>+0.022</td>
</tr>
</tbody>
</table>

Table 4: Localization results in terms of AUC for the different datasets. AUCs are reported in two different cases: autoencoder trained with or without the GAN. Best results are reported in italics.

<table>
<thead>
<tr>
<th>Forgery Size</th>
<th>AUC (without GAN)</th>
<th>AUC (with GAN)</th>
<th>AUC Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>0.913</td>
<td>0.902</td>
<td>-0.009</td>
</tr>
<tr>
<td>Medium</td>
<td>0.963</td>
<td>0.961</td>
<td>-0.002</td>
</tr>
<tr>
<td>Large</td>
<td>0.970</td>
<td>0.974</td>
<td>-0.004</td>
</tr>
</tbody>
</table>
Figures 6: Examples of forged images with ground truth forged mask $M$ and estimated soft mask $\tilde{M}$. It is possible to notice the correlation between ground truth and estimated soft mask.

In this paper we proposed a solution for satellite imagery forgery detection and localization. The rationale behind the proposed method is that it is possible to train an autoencoder to obtain a compact representation of image patches coming from pristine satellite pictures. This autoencoder can then be used as a feature extractor for image patches. During testing, a one-class SVM is used to detect whether feature vectors come from pristine images or not, thus representing forgeries.

A final consideration is devoted to computational time. We tested the proposed algorithm on a workstation equipped with an Intel Core i7-5930K CPU, 128 GB of RAM and a NVIDIA GeForce Titan X GPU. The processing time needed for a $64 \times 64$ pixel patch (considering both the autoencoder and the SVM) was around 500μs for testing. As each patch processing is independent, the algorithm allows for strong parallelization, thus making processing of high resolution images not an issue.

Conclusions
In this paper we proposed a solution for satellite imagery forgery detection and localization. The rationale behind the proposed method is that it is possible to train an autoencoder to obtain a compact representation of image patches coming from pristine satellite pictures. This autoencoder can then be used as a feature extractor for image patches. During testing, a one-class SVM is used to detect whether feature vectors come from pristine images or not, thus representing forgeries.

The solution proposed in this paper makes use of generative adversarial networks to train the autoencoder for the forgery detection task. Moreover, it is worth noting that the whole system is trained only on pristine data. This means that no prior knowledge on the forgeries is assumed to be available.

Tests on copy-paste attacked images with different forgery size show promising accuracy in both detection and localization. Future work will be devoted to study system robustness to different kinds of forgeries as well.

Acknowledgments
This material is based on research sponsored by DARPA and Air Force Research Laboratory (AFRL) under agreement number FA8750-16-2-0173. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclu-
sions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA and Air Force Research Laboratory (AFRL) or the U.S. Government.

References


[32] National Aeronautics and Space Administration, Landsat Science,
Author Biography

Sri Kalyan Yarlagadda received his btech in Electrical Engineering from Indian Institute of Technology Madras in July 2015. He is currently pursuing a Ph.D. at the School of Electrical and Computer Engineering, Purdue University, USA. His research interests are video/image processing and deep learning.

David Güera received his B.E. (Hons.) in Telecommunications Systems Engineering from Polytechnic University of Catalonia - BarcelonaTech, Spain in July 2016. He is currently pursuing a Ph.D. at the Video and Image Processing Laboratory (VIPER), School of Electrical and Computer Engineering, Purdue University, USA. His research interests are video/image processing and deep learning.

Paolo Bestagini received the M.Sc. in Telecommunications Engineering and the Ph.D. in Information Technology at the Politecnico di Milano, Italy, in 2010 and 2014, respectively. Since 2016 he is Assistant Professor at the Department of Electronics, Informatics and Bioengineering, Politecnico di Milano. His research activity is focused on multimedia forensics and acoustic signal processing for microphone arrays.

Fengqing Zhu is an Assistant Professor of Electrical and Computer Engineering at Purdue University, West Lafayette, IN. Dr. Zhu received her Ph.D. in Electrical and Computer Engineering from Purdue University in 2011. Prior to joining Purdue in 2015, she was a Staff Researcher at Huawei Technologies (USA), where she received a Huawei Certification of Recognition for Core Technology Contribution in 2012. Her research interests include image processing and analysis, video compression, computer vision and computational photography.

Stefano Tubaro completed his studies in Electronic Engineering at the Politecnico di Milano, Italy, in 1982. He joined the Politecnico di Milano, first as a researcher and in 1991 as an Associate Professor. Since 2004 he has been appointed as Full Professor of Telecommunications. His current research interests are on advanced algorithms for video and sound processing and for image and video tampering detection. He authored over 180 scientific publications and more than 15 patents.

Edward J. Delp was born in Cincinnati, Ohio. He is currently The Charles William Harrison Distinguished Professor of Electrical and Computer Engineering and Professor of Biomedical Engineering at Purdue University. His research interests include image and video processing, image analysis, computer vision, image and video compression, multimedia security, medical imaging, multimedia systems, communication and information theory. Dr. Delp is a Life Fellow of the IEEE, a Fellow of the SPIE, a Fellow of IS&T, and a Fellow of the American Institute of Medical and Biological Engineering.