Title

Hybrid LSTM and Encoder—Decoder Architecture for Detection of Image Forgeries

Journal

IEEE Transactions on Image Processing, 28(7)

ISSN

1057-7149

Authors

Bappy, Jawadul H

Simons, Cody

Nataraj, Lakshmanan

et al.

Publication Date

2019-07-01

DOI

10.1109/tip.2019.2895466

Peer reviewed

Hybrid LSTM and Encoder-Decoder Architecture for Detection of Image Forgeries

Jawadul H. Bappy, Cody Simons, Lakshmanan Nataraj, B.S. Manjunath, and Amit K. Roy-Chowdhury

Abstract—With advanced image journaling tools, one can easily alter the semantic meaning of an image by exploiting certain manipulation techniques such as copy-clone, object splicing, and removal, which mislead the viewers. In contrast, the identification of these manipulations becomes a very challenging task as manipulated regions are not visually apparent. This paper proposes a high-confidence manipulation localization architecture which utilizes resampling features, Long-Short Term Memory (LSTM) cells, and encoder-decoder network to segment out manipulated regions from non-manipulated ones. Resampling features are used to capture artifacts like JPEG quality loss, upsampling, downsampling, rotation, and shearing. The proposed network exploits larger receptive fields (spatial maps) and frequency domain correlation to analyze the discriminative characteristics between manipulated and non-manipulated regions by incorporating encoder and LSTM network. Finally, decoder network learns the mapping from low-resolution feature maps to pixel-wise predictions for image tamper localization.
With predicted mask provided by final layer (softmax) of the proposed architecture, end-to-end training is performed to learn the network parameters through back-propagation using ground-truth masks. Furthermore, a large image splicing dataset is introduced to guide the training process. The proposed method is capable of localizing image manipulations at pixel level with high precision, which is demonstrated through rigorous experimentation on three diverse datasets.

Index Terms—Image Forgery, Tamper Localization, Segmentation, Resampling, LSTM, CNN, Encoder, Decoder.

1 Introduction

The detection of image forgery has become very difficult as manipulated images are often visually indistinguishable from real images. With the advent of high-tech image editing tools, an image can be manipulated in many ways. The types of image manipulation can broadly be classified into two categories: (1) content-preserving, and (2) content-changing [40]. The first type of manipulation (e.g., compression, blur and contrast enhancement) occurs mainly due to post-processing, and they are considered as less harmful since they do not change any semantic content. The latter type (e.g., copy-move, splicing, and object removal) reshapes image content arbitrarily and alters the semantic meaning significantly [40]. The content-changing manipulations can convey false or misleading information. As the number of tampered images grows at an enormous rate, it becomes crucial to detect the manipulated images to prevent viewers from being presented with misleading information. Recently, the detection of content-changing manipulation from an image or a video has become an area of growing interest in diverse scientific and security/surveillance applications. In this paper, we present a novel architecture to localize manipulated regions at pixel level for content-changing manipulation.

Over the past decades, there have been many works to classify image manipulation, i.e., whether an image is tampered or not [13], [30], [39], [47], [65], [74], [78]. However, only few works [7], [14] attempt to localize image manipulation at pixel level. Some recent works [17], [26], [54] address the localization problem by classifying patches as manipulated. The localization of image tampering is a very challenging task as well-manipulated images do not leave any visual clues, as shown by the following examples in Fig. 1. In Fig. 1(a), copy-move manipulation is illustrated where one object is copied to another region of the same image leading to two similar objects, one originally present, and another manipulated. However, only the latter needs to be identified. Fig. 1(b) illustrates object splicing manipulation, where an object from a donor image has been spliced into other image. As another example, if an object is removed as shown in Fig. 1(c), the region may visually blend into the background, but needs to be identified as manipulated.

Most of the state-of-the-art image tamper classification approaches utilize the frequency domain characteristics and/or statistical properties of an image [46], [56], [58], [85]. The analysis of artifacts by multiple JPEG compressions is also utilized in [20], [85] to detect manipulated images, which are applicable only to the JPEG formats. In [67], [68], noise has been added to the JPEG compressed image in order to improve the performance of resampling detection. In computer vision, deep learning has shown promising performance in different visual recognition tasks such as object detection [32], scene classification [90], and semantic segmentation [55]. Some recent deep learning based methods such as stacked auto-encoders (SAE) [88] and convolutional neural networks (CNN) [8], [21], [77] have also been applied to detect/classify image manipulations. In media forensics, most of the existing forgery detection approaches focus on identifying a specific tampering method, such as copy-move [19], [36], [48], and splicing [61]. Thus, one approach might not do well on other types of tampering. Moreover, it seems unrealistic to assume that the type of manipulation will be known beforehand. Our recent paper [7], upon which this particular work builds, presents a general detection architecture for different content-changing

- *J. Bappy was with the ECE department at University of California, Riverside at the time of this work. He is currently at JD.com. E-mail: mbappy@ece.ucr.edu*
- *C. Simons, and A. Roy-Chowdhury are with the Department of Electrical and Computer Engineering, University of California, Riverside, CA. E-mail: mbappy,csimons,amitrc@ece.ucr.edu*
- *L. Nataraj is with Mayachitra Inc., Santa Barbara, CA. E-mail: nataraj@mayachitra.com*
- *B.S. Manjunath is with Mayachitra Inc. and University of California, Santa Barbara, CA. E-mail: manj@ucsb.edu*

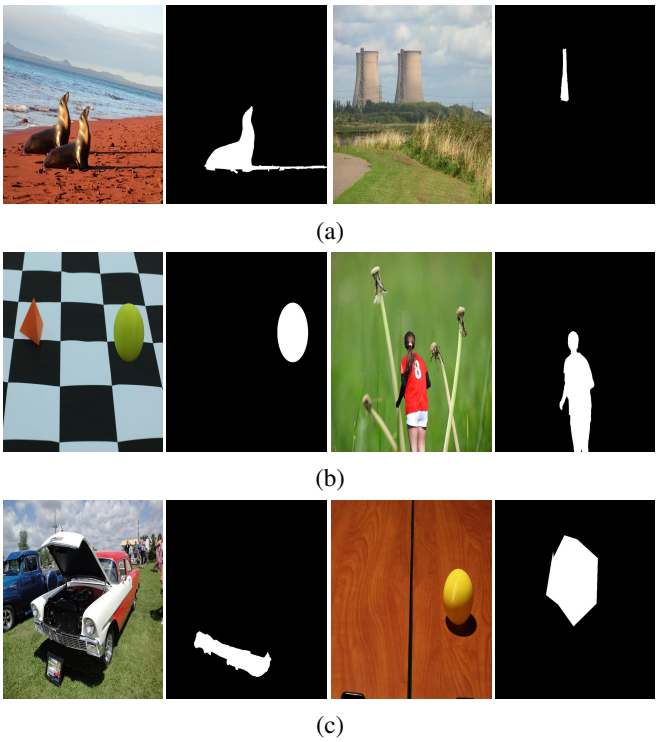


Fig. 1. The figure demonstrates some examples of content-changing manipulations. (a), (b), (c) illustrate copy-clone, splicing and object removal techniques to manipulate an image. First and third columns are tampered images and corresponding ground-truth masks are shown in second and fourth columns.

manipulations.

Unlike semantic object segmentation where all meaningful regions (objects) are segmented, the localization of image manipulation focuses only the possible tampered region which makes the problem even more challenging. In computer vision, recent advances in semantic segmentation methods [6], [55], [89] are based on convolutional neural networks (CNN). In [89], a fully convolutional network is utilized to analyze the content of the objects and shape of a region by extracting the hierarchical features at different levels. In object detection [32] and segmentation [6], [55], CNN based architectures demonstrate very promising performance in understanding visual concepts by analyzing the content of different regions. In contrast to semantic segmentation, manipulated regions could be removed objects, or copied objects from other parts of the image. Well-manipulated images usually show strong resemblance between fake and genuine objects/regions (i.e. content is similar) [77]. Even though CNN generates spatial maps for different regions of an image, it can not generalize some other artifacts created by different manipulation techniques. Thus, the localization of manipulated regions with only CNN based architecture may not be the best strategy. In our earlier work [7], we compared with some recent semantic segmentation approaches [55], [89] that do not perform well for copy-clone and object removal type of manipulations.

Image tampering creates some artifacts, e.g., resampling, compression, shearing, which are better captured by resampling features [17], [29], [79]. In [17], a Long short-term memory (LSTM) based network is presented in order to classify manipulated patches where resampling features are utilized as an important signature. The authors trained six classifiers to detect six different types of resampling (e.g., JPEG quality thresholded above or below 85, upsampling, downsampling, rotation clockwise, rotation counterclockwise, and shearing. Resampling introduces periodic correlations among pixels due to interpolation. As convolutional neural networks exhibit robust translational invariance to generate spatial maps for the different regions of the image, and certain artifacts are well-captured in resampling features, both can be exploited in order to localize manipulated regions.

Towards the goal of localizing manipulated regions in an image, *we present a unified architecture that exploits resampling features, LSTM network, and encoder-decoder architectures in order to learn the pixel level localization of manipulated image regions*. Given an image, we divide into several blocks/patches and then resampling features (as discussed in Sec. 3.1.1) are extracted from each block. LSTM network is utilized to learn the correlation between manipulated and non-manipulated blocks at frequency domain. We utilize and modify encoder-decoder network as presented in [6] to capture spatial information. Each encoder generates feature maps with varying size and number. The feature maps from LSTM network and the encoded feature maps from encoders are embedded before going through the decoder. We perform end-to-end training to learn the parameters of the network through back-propagation using ground-truth mask information. As deep networks are data hungry, a large number of images are synthesized to augment the training data.. The proposed model shows promising results in localizing manipulated regions at the pixel level, which is demonstrated on different challenging datasets.

1.1 Approach Overview

Given an image, our goal is to localize the manipulated regions at a pixel level. The proposed framework is shown in Fig. 2. Our network can be divided into three parts- (1) LSTM network with resampling features, and (2) convolutional encoder, and (3) decoder network.

For the first part, we divide image into patches. For each patch, resampling features [17] have been extracted. With extracted resampling features, we use Hilbert curve (discussed in Sec. 3.1.2) to determine the ordering of the patches to feed into LSTM cells. We allow LSTM cells to learn the transition between manipulated and non-manipulated blocks in the frequency domain. Finally, feature maps are generated from the LSTM cell output, which will be combined with the feature maps from the encoder. An encoder consists of residual block, batch normalization and activation function. At each residual block, two convolutions are performed with shortcut connection. After each residual unit, max-pooling operation is performed which gives translation invariance.

Our next step is to design a decoder that can provide finer representation of different regions in a mask. We combine both spatial features from encoder and output features from LSTM to understand the nature of manipulation. Then, these features are taken as input to the decoder. Each decoder follows basic operations like upsampling, convolution, batch normalization and activating feature maps (using activation function). The decoders help learn the finer details of the manipulated and non-manipulated classes. Finally, a softmax layer is used to predict manipulated pixels against non-manipulated ones. With the ground-truth mask of manipulated regions we perform end-to-end training to classify each pixel. We compute cross entropy loss, which is then minimized by utilizing back-propagation algorithm. After optimization, we find the optimal set of parameters for the network, that will be used to predict manipulated regions given a test set.

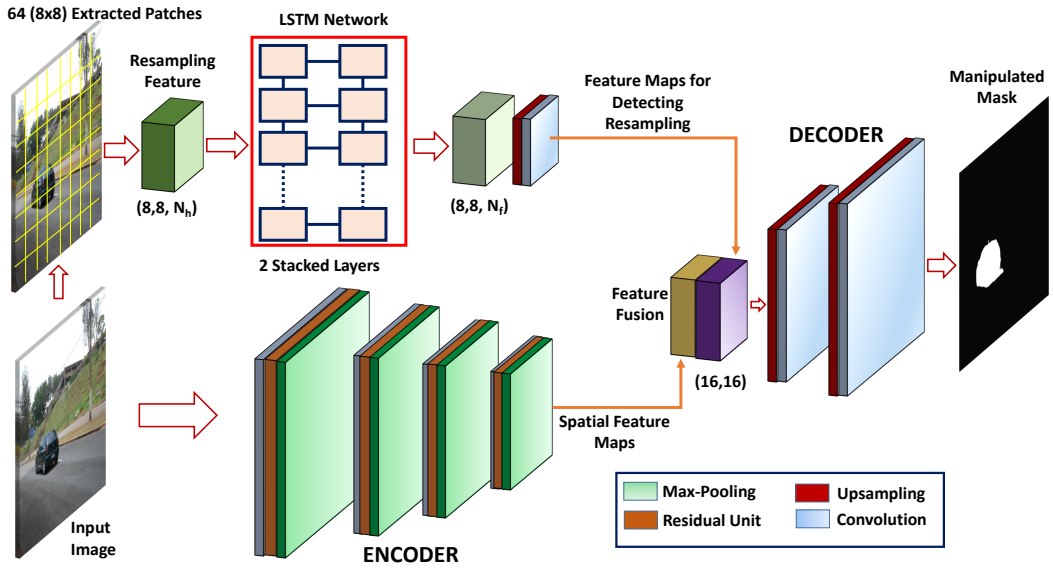


Fig. 2. Overview of proposed framework for localization of manipulated image regions.

1.2 Main Contributions

Our *main contributions* are as follows.

• We propose a novel localization framework that exploits both frequency domain features and spatial context in order to localize manipulated image regions, which makes our work significantly different than other state-of-the-art methods.

• Unlike most of the existing works where patches are used as input, we consider image as input so that we can utilize global context. Our architecture is able to localize manipulated region with high confidence as demonstrated on three datasets.

• We present a new dataset for image tamper localization task that includes a large number of images with ground-truth binary mask. This dataset is larger than current publicly available datasets such as IEEE Forensics [1] and COVERAGE [86]. It will also help train deeper networks for image tamper classification or localization tasks.

This work builds upon our earlier paper [7], but with significant differences. First, the method presented in the paper [7] exploits low level features such as tampered edges, as evidence of tampering, which cannot always detect the entire tampered regions. The proposed method exploits an encoder and an LSTM network to extract spatial feature maps and frequency domain features respectively in order to localize manipulated regions. In the proposed method, the encoder provides larger receptive fields by exploiting multiple convolutional layers which allow the network to identify large manipulated regions. Second, we consider the image as the input, instead of patches, which helps the network to learn more meaningful context, i.e., intra-patch and inter-patch correlation. Third, unlike [7], we utilize resampling features in our network that captures the characteristics of different artifacts due to image transformation such as JPEG quality above or below a threshold, upsampling, downsampling, rotation clockwise, rotation counterclockwise, and shearing. Fourth, we present a large image splicing dataset which can be used to train a deep neural network for the task of manipulation. We show the comparison against existing dataset in experimental analysis.

2 Related Work

In media forensics, there have been lot of efforts to detect different types of manipulations such as resampling, JPEG artifacts, and content-changing manipulations. In this section, we will briefly discuss some of the existing works for detecting image forgeries.

In last few years, several methods have been proposed to detect resampling in digital images [29], [59], [69], [73], [79]. Most of the approaches exploit linear or cubic interpolation. In [79], periodic properties of interpolation by the second-derivative of the transformed image have been utilized for detecting image manipulation. In [69], an approach was presented to identify resampling on JPEG compressed images where noise was added before passing the image through the resampling detector; it was shown that adding noise aids in detecting resampling. In [28], [29], a feature was generated from the normalized energy density and then SVM was used to robustly detect resampled images. Some recent approaches [34], [45] have been proposed to reduce JPEG artifacts produced by compression techniques. In [4], [82], feature based methods have been presented in order to detect manipulation in an image. Many methods have been proposed to detect seam carving [31], [53], [80] and inpainting based object removal [20], [50], [87]. Several approaches exploit JPEG blocking artifacts to detect tampered regions [12], [13], [27], [52], [57]. Some recent works [3], [39], [41], [48] focus on identifying and localizing copy-move manipulation. In [48], the authors used an interesting segmentation based approach to detect copy-move forgeries. They first divided an image into semantically independent patches and then performed keypoint matching among these patches. In [24], a patch match algorithm was used to efficiently compute an approximate nearest neighbor field over an image. They further used invariant features such as Circular Harmonic transforms and showed robustness over duplicated blocks that have undergone geometrical transformations. In [61], an image splicing technique was presented using visual artifacts. In [64], the steerable pyramid transform (SPT) and the local binary pattern (LBP) were utilized to detect image forgeries. The paper [35] highlights the recent advances in image manipulation and also discusses the process of restoring missing or damaged areas in an image. In [5], a review on different image forgery detection techniques is presented.

Recently, there has been a growing interest to detect image manipulations by applying different computer vision and machine learning algorithms [70], [76]. In semantic segmentation, many deep learning architectures [6], [55], [89] have been proposed, which surpass previous state-of-the-art approaches by a large margin in terms of accuracy. Most of the deep networks [6], [55] are based on Convolutional Neural Networks (CNNs), where hierarchical features are exploited at different layers in order to learn the spatial map for semantic region. In [55], a classification-purposed CNN is transformed into fully convolutional one by replacing fully connected layers to produce spatial heatmaps. Finally, a deconvolution layer is used to upsample the heatmaps to generate dense per-pixel labeling. SegNet [6] designs a decoder to efficiently learn the low-resolution heatmaps for pixel-wise predictions for segmentation. In [22], [42], the fully connected pairwise CRF is utilized as a post-processing step to refine the segmentation result. In [72], skip connection is exploited to perform late fusion of feature maps for making independent predictions for each layer and merging the results. In ReSeg [83], Gated Recurrent Units (GRUs) and upsampling have been used to obtain the segmentation mask.

Recent efforts, including [8], [9], [17], [62], [77] in the manipulation detection task, exploit deep learning based models. These tasks include detection of generic manipulations [8], [9], resampling [10], splicing [77], and bootleg [16]. In [75], the authors propose Gaussian-Neuron CNN (GNCNN) for steganalysis. A deep learning approach to identify facial retouching was proposed in [11]. In [88], image region forgery detection has been performed using stacked auto-encoder model. In [8], a new form of convolutional layer is proposed to learn the manipulated features from an image. In computer vision, deep learning has led to significant performance gain in different visual recognition tasks such as image classification [90], and semantic segmentation [55]. The deep networks extract hierarchical features to represent the visual concept, which is useful in object segmentation. Most of the architectures are based on Convolutional Neural Network (CNN), which provides spatial maps relevant to manipulated regions. However, we can also exploit resampling features that distinguish other artifacts. Since both spatial context and resampling are important attributes to localize manipulated regions from image, we present an unique network that exploits both of the features.

3 Network Architecture Overview

Our main goal of this work is to localize image manipulations at pixel level. Fig. 2 shows our overall framework. The whole network can be divided into three parts - (1) LSTM network with resampling features, and (2) Encoder, and (3) Decoder network. Convolutional neural network (CNN) architectures extract meaningful spatial features for object segmentation, which could also be useful to localize manipulated objects. Even though spatial feature maps are crucial to classify each pixel, solely using CNNs in the image domain does not usually perform well in identifying image manipulations. It is simply because there are certain manipulations like upsampling, downsampling, compression, which are well-captured in the frequency domain. Thus, we use resampling features from the extracted patches of an image. These resampling features are considered as input to the LSTM network which learns the correlation between different patches. An encoder architecture is also utilized to understand the spatial location of manipulated region. Before decoder network, we utilize the meaningful features by exploiting both spatial and frequency domain. Finally, we use decoder network to obtain finer representation of binary mask to localize tampered region from low-resolution feature maps. In order to develop encoder-decoder network, we utilize convolutional layers, batch normalization, max-pooling and upsampling. Next, we will discuss the technical details of our proposed architecture for image tamper localization.

3.1 LSTM Network with Resampling Features

3.1.1 Resampling Features

The typical content-changing manipulations are copy-clone, splicing and object removal, which are difficult to detect. In general, these manipulations distort the natural statistics at the boundary of tampered regions. In [59], the method of resampling detection using Radon transform is presented. Laplacian filter along with Radon transform is exploited in order to extract resampling features given a patch. We will also follow a similar procedure for extracting resampling features. To illustrate how Radon transform captures resampling characteristics, we provide two examples to highlight the difference in statistics between manipulated and non-manipulated patches as shown in Fig. 3 (c), with the top row containing no manipulation and the bottom row containing some manipulations due to resampling. Fig. 3 (d,e) illustrates the radon transform of two patches from the manipulated and non-manipulated regions and their sum along the columns. Though the differences are subtle, we can see that there is a pattern for manipulated patches which is different from those of non-manipulated patches. Given an image, we first extract 64 (8×8) non-overlapping patches. As input image has size of 256x256x3, the dimension of each patch would be 32x32x3. Then, the square root of magnitude of 3×3 Laplacian filter is used to produce the magnitude of linear predictive error for each extracted patch as presented in [17]. As resampling signal has periodic correlations in the linear predictor error, we apply the Radon transform to accumulate errors along various angles of projection. In our experiment, we use 10 angles. Finally, we apply Fast Fourier Transform (FFT) to find the periodic nature of the signal. In general, these resampling features are capable of capturing different resampling characteristics- JPEG quality thresholded above or below a threshold, upsampling, downsampling, rotation clockwise, rotation counterclockwise, and shearing (in an affine transformation matrix). In order to reduce computational burden, we resize images to 256×256 which might introduce some additional artifacts such as degradation in image quality factor, shearing, upsampling, downsampling. In [17], resampling features are used to classify these artifacts. In this work, we also utilize resampling features, which gives us robust performance. Unlike [17], where resampling features are considered for patch classification, we perform localization at pixel level. There is a tradeoff in selecting the patch size: resampling is more detectable in larger patch sizes because the resampling signal has more repetitions, but small manipulated regions will not be localized that well. In [17], resampling features are extracted from 8×8 block. On the other hand, we choose 32×32 small patches from an image to extract resampling features that capture more information. The major motivation of utilizing the resampling features for patches is to characterize the local artifacts due to different types of manipulations.

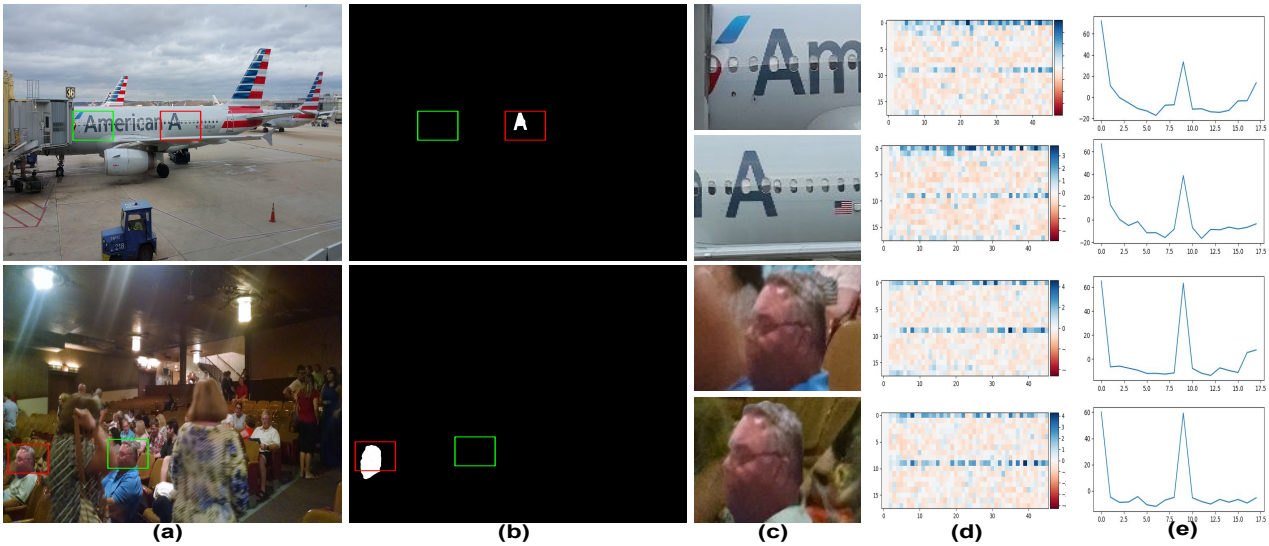


Fig. 3. (a) Examples of manipulated images from the NIST dataset [2]. (b) Corresponding ground-truth masks for the manipulated images in column (a), green for non-manipulated patches and red for manipulated ones. (c) The patches extracted from the corresponding image, with the top containing no manipulation and the bottom one containing some manipulations. (d) Radon transform of two patches from the manipulated and non-manipulated images. (e) Sum along the columns of the radon transform. Here, we can see that the non-manipulated patch exhibits high magnitude at the right of the curve, whereas low value is observed in the same region for the manipulated patch. Differences such as these are seen across other manipulated and non-manipulated patches.

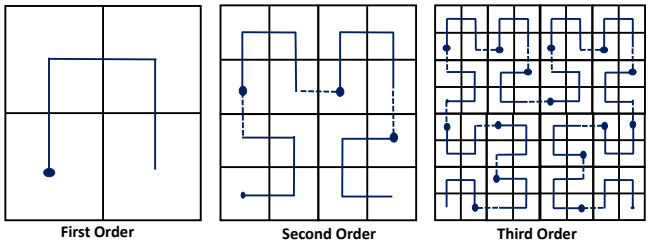


Fig. 4. The figure illustrates Hilbert curves for different orders. In this work, third order curve has been exploited.

3.1.2 Hilbert Curve

Long-Short Term Memory (LSTM) is commonly used in tasks where sequential information exists. The performance of LSTM highly depends on the ordering of the patches (sequence of the extracted patches). One can consider horizontal or vertical directions, but these orderings do not capture local information well. For example, if we were to iterate horizontally over the rows of patches, then patches that neighbor each other vertically will be separated by an entire row of patches. Due to this long time lag, LSTM can not correlate well between these patches. If we were to iterate vertically over the columns we would face similar issues.

In order to better preserve the spacial locality of the patches, we use space-filling curve which is commonly used to reduce multi-dimensional problem to a one-dimensional [15]. The Hilbert curve has been shown to outperform many other curves in maintaining the spatial locality, when transforming from a multi-dimensional space to a one-dimensional space [63]. The major advantage of Hilbert curve is in applications where the coherence between neighboring patches/blocks is important [84]. Fig. 4 shows the process of how Hilbert curve works. The basic elements of the Hilbert curves can be divided into 'cups' (a square with one open side) and 'joins' (a vector that joins two cups) [84]. Every cup has two end-points - (a) entry point, and (b) exit point. From Fig. 4(left), we can see that a single cup represents a first order Hilbert curve which fills a 2×2 space. The second order Hilbert curve contains four cups, which are linked together by three joins as shown in Fig. 4(middle). A third order Hilbert curve repeats the process by dividing into four parts, each of these parts contains second order Hilbert curve. Finally, the four parts are connected by three joins. So, the main mechanism is to divide a plane into four parts, each of these parts into four parts, and so on. As we have total 64 (8×8) blocks extracted from an image, we require three recursive dividing of the plane. After ordering the patches with Hilbert curve, LSTM network is utilized. We empirically observe that this ordering technique helps improve the performance of localization.

3.1.3 Long-Short Term Memory (LSTM) Network

LSTM network is well-known for processing sequential data in different applications such as language modeling, machine translation, image captioning, and hand writing generation. In computer vision, LSTM network has been successfully used to capture the dependency among a series of pixels [18], [71]. The key insight of using LSTM for detecting image manipulations is *to learn the boundary transformation between different blocks, which provides discriminative features between manipulated and non-manipulated regions*.

In [7], [17], LSTM network is utilized in order to learn the transition (change) between manipulated vs non-manipulated blocks by feeding the blocks into an LSTM network. In [17], the authors propose a patch classification framework where frequency domain features are extracted from 8×8 block before LSTM network. The method could be more effective by considering larger block size. Unlike these approaches, we divide an image into several patches, and extract rasampling features as discussed in Sec. 3.1.1 from 32×32 size of patch that are taken as input to the LSTM network.

After extracting resampling features for each patch, we use Hilbert curve (discussed in Sec. 3.1.2) to determine the ordering of the patches. Then, we feed the resampling features extracted from

patches into LSTM cells in a sequential manner. LSTM network computes the logarithmic distance of patch dependency by feeding each patch to each cell. The LSTM cells learn the correlation among neighboring patches. In this paper, we use 2 stacked layers, and 64 time steps in LSTM network. We obtain 64 dimensional feature vector from each time step in the last layer. Then, we project the vector generated by LSTM network to N_f features maps. Let us denote a feature vector $F_l \in \mathcal{R}^{1\times N_h}$ produced by l^{th} time step of LSTM network. We represent the projected vector as F_l' with N_f dimension. In order to obtain the output O_l, we introduce a weight matrix W_l ($\in \mathcal{R}^{N_h \times N_f}$) which transforms from F_l to F_l'. The vector F_l' can be written as

$$F_l' = F_l.W_l + B_l. \tag{1}$$

Here, B_l is bias with N_f dimension. Each time step of LSTM network actually provides the transformed feature for each of the extracted patches from input image. Finally, we obtain $64 \times N_f$ size matrix for 64 patches. In our experiment, we choose $N_h = 128$ and $N_f = 64$. Next, we carefully choose the ordering of the cell outputs in order to preserve the spatial information. Then, we reshape the $64 \times N_f$ matrix to $8 \times 8 \times N_f$, where first two dimensions represent the location of the patch as shown in Fig. 2.

LSTM Cell Overview. Information flow between the LSTM cells is controlled by three gates: (1) input gate, (2) forget gate, and (3) output gate. Each gate has a value ranging from zero to one, activated by a sigmoid function. Let us denote cell state and output state as C_t and z_t for current cell t. Each cell produces new candidate cell state $\bar{\mathcal{C}}_t$. Using the previous cell state $\mathcal{C}_{t-1}$ and $\bar{C}_t$, we can write the updated cell state $\mathcal{C}_t$ as

$$\mathcal{C}_t = f_t \circ \mathcal{C}_{t-1} + i_t \circ \bar{\mathcal{C}}_t \tag{2}$$

Here, $\circ$ denotes the $pointwise$ multiplication. Finally, we obtain the output of the current cell h_t, which can be represented as

$$z_t = o_t \circ tanh(\mathcal{C}_t) \tag{3}$$

In Eqns. 2 and 3, i, f, o represent input, forget and output gates.

3.2 Encoder Network

Our main objective is to design an efficient architecture for pixel-wise tamper region segmentation. We use convolutional layers to design the encoder which allows the network to understand appearance, shape and the spatial-relationship (context) between manipulated and non-manipulated classes. In [6], [22], [55], some deep architectures are presented where convolutional layers are utilized in order to produce spatial heatmaps for semantic segmentation. As spatial information is very important to localize manipulated regions, we also incorporate convolutional layers into our framework. We exploit and modify encoder-decoder architecture as presented in [6]. The encoder component is similar to CNN architecture except the fully connected layers.

Convolutional Network (ConvNet) consists of different layers, where each layer of data is a three-dimensional array of size $h \times w \times c$, where h and w are height and width of the data, and c is the dimension of the channels. Each layer of convolution involves learnable filters with varying size. The filters in convolutional layer will create feature maps that are connected to the local region of the previous layer. In the first layer, image is taken as input with dimension of $256 \times 256 \times 3$ (width, height, color channels).

The basic building block of each encoder utilizes convolution, pooling, and activation functions. We use residual unit [37] for each encoder. Residual block takes advantage of shortcut connections that are parameter free. The main advantage of using residual unit is that it can easily optimize the residual mapping and more layers are trainable. Let us consider an input to the residual unit is y, and the mapping from input to output of the unit is $\mathcal{T}(.)$. The output of residual unit would be $\mathcal{T}(y) + y$ in the forward pass. In each convolutional layer, we use kernel size of $3 \times 3 \times d$, where d is the depth of a filter. We use different depth for different layers in the network. In encoder network, the number of filters are generally in increasing order. In this work, we utilize $32, 64, 128$, and 256 feature maps in first, second, third and fourth layer of encoder architecture respectively.

Each residual unit in the encoder produces a set of feature maps. We utilize batch normalization [38] at each convolutional layer. Batch normalization is robust to covariance shift. As an activation function, we choose rectified linear unit (ReLU) [66] that can be represented as $max(0, x)$. At the end of each residual unit, max-pooling with stride 2 is performed, which reduces the size of feature maps by a factor of 2. Unlike [7], we exploit max-pooling [44] at each layer as it provides translation invariance. Each max-pooling operation introduces a loss of spatial resolution (i.e., boundary details) of the feature maps. The loss in boundary detail can be compensated by using decoder which is introduced in [6], and discussed next.

3.3 Decoder Network

In [55], a decode technique is proposed that requires encoder feature maps to be stored during prediction. This process might not be applicable in real-life as it requires intensive memory. In this paper, we follow a decoding technique that is presented in [6]. In [6], the advantage of using decoder has been discussed in details. The key part is the decoder which replaces the fully connected layers. The decoder decodes the feature output from encoder. As encoder-decoder is primarily developed for semantic object segmentation [6], we exploit and tune this network in order to segment manipulated objects. In the upsampling step, no learnable parameters are involved. Different multi-channel filters are utilized which are convolved with the upsampling heatmaps (coarse representation) to create dense maps. Each decoder follows basic operations - upsample, convolution, and batch normalization. Each decoder first performs upsampling of the feature maps learned at previous layer. Following that, convolutional operation and batch normalization are performed. We employ 3×3 size kernel for decoder network. In our decoder, 64 and 16 feature maps are exploited in first and second layer respectively. Finally, 2 heat maps are used for the prediction of manipulated and non-manipulated class at the end of decoder network. Fig. 2 shows the decoder operation of the network. At the end of network, we obtain finer representation of spatial maps that indicates the manipulated regions in an image.

3.4 Training the Network

Soft-max Layer. In order to predict the pixel-wise classification, softmax layer is used at the end of the network.

Let us denote the probability distribution over various classes as $P(\mathcal{Y}_k)$ which is provided by softmax classifier. Now, we can predict label by maximizing $P(\mathcal{Y}_k)$ with respect to k. The predicted label can be obtained by $\hat{\mathcal{Y}} = \arg\max_k \; P(\mathcal{Y}_k)$. As we are only interested to predict manipulated pixels against non-manipulated pixels, the value of k would be 2. Given the predicted

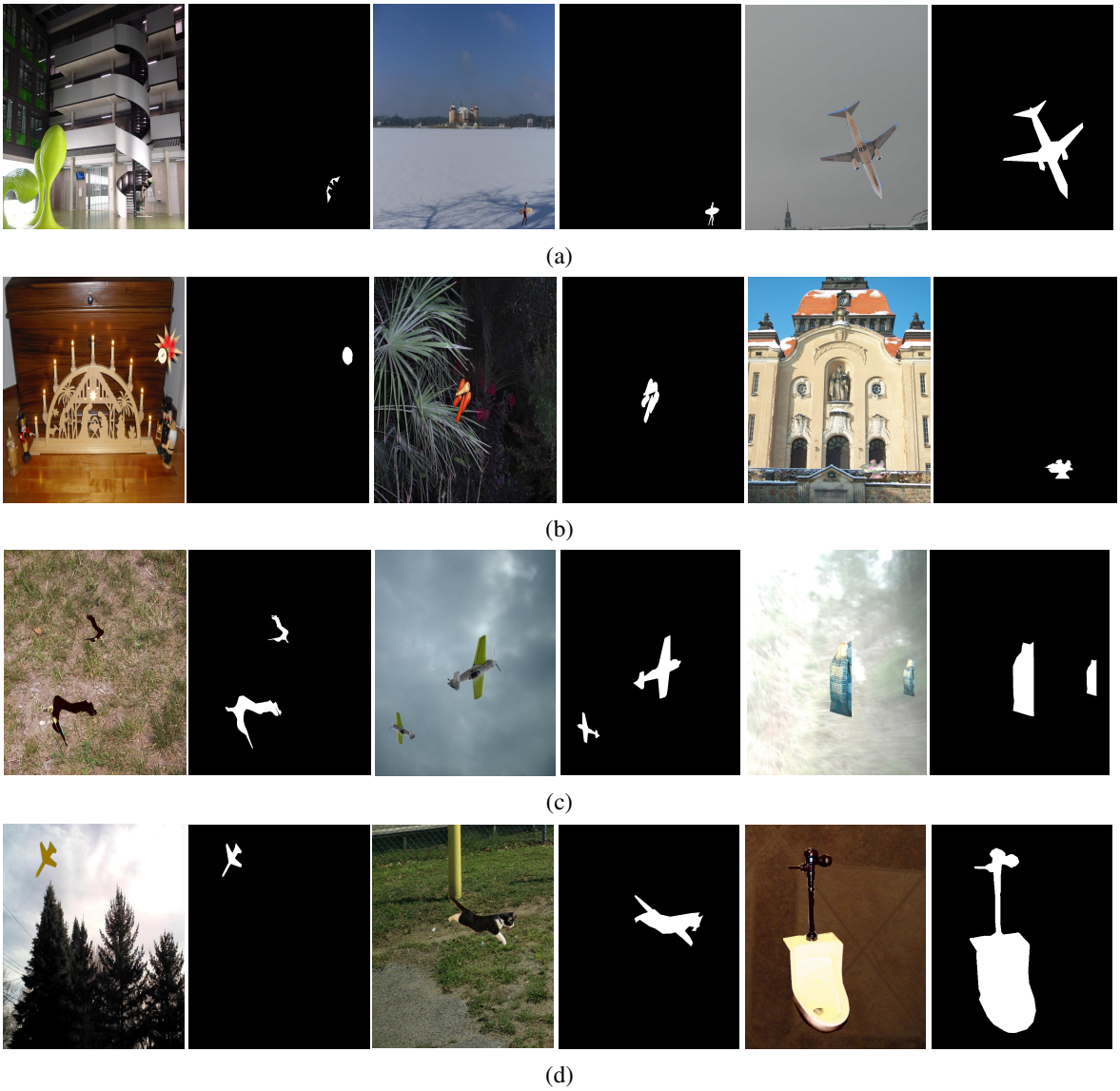


Fig. 5. The figures show some manipulated images with corresponding ground-truth masks from synthetic dataset. (a) and (b) show images created from DRESDEN [33] dataset. (c) and (d) are the manipulated images created from NIST [2] dataset.

mask provided by softmax layer, we can compute the loss that will be used to learn the parameter through back-propagation.

Training Loss. During training, we use cross entropy loss, which is minimized to find the optimal set of parameters of the network. Let θ be the parameter vector corresponding to image tamper localization task. So, the cross entropy loss can be computed as

$$\mathcal{L}(\theta) = -\frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbb{1}(\mathcal{Y}^m = n)\log(\mathcal{Y}^m = n|y^m;\theta) \quad (4)$$

Here, M and N denote the total number of pixels, and the number of class. y represents the input pixel. $\mathbb{1}(.)$ is an *indicator function*, which equals to 1 if $m = n$, otherwise it equals 0. In our experiment, we observe that weighted cross entropy loss provides better result. It is simply because the imbalance between the number of non-manipulated and manipulated pixels. We put more weight on manipulated pixels over non-manipulated pixels. In this work, the class weights are inversely proportional to their frequency in the training set. The weights are normalized to lie in between 0 and 1. We use *adaptive moment estimation (Adam)* [43] optimization technique in order to minimize the loss of the network, shown in Eqn. 4. At each iteration, one mini-batch is processed to update the parameters of the network. In order to learn the parameters effectively, we choose the mini-batch very carefully which will be discussed in details in Sec. 4. After optimizing the loss function over several epochs, we learn the optimal set of parameters of the network. With these optimal parameters, the network is able to predict pixel-wise classification given a test image.

TABLE 1
A comparison of common image tampering datasets

Data Set	# image pairs	Avg. Image Size
CoMoFod [81]	260	512×512
Manip [23]	48	2305×3020
GRIP [25]	100	1024×786
COVERAGE [86]	100	400×486
Synthesized	$65k$	1024×1024

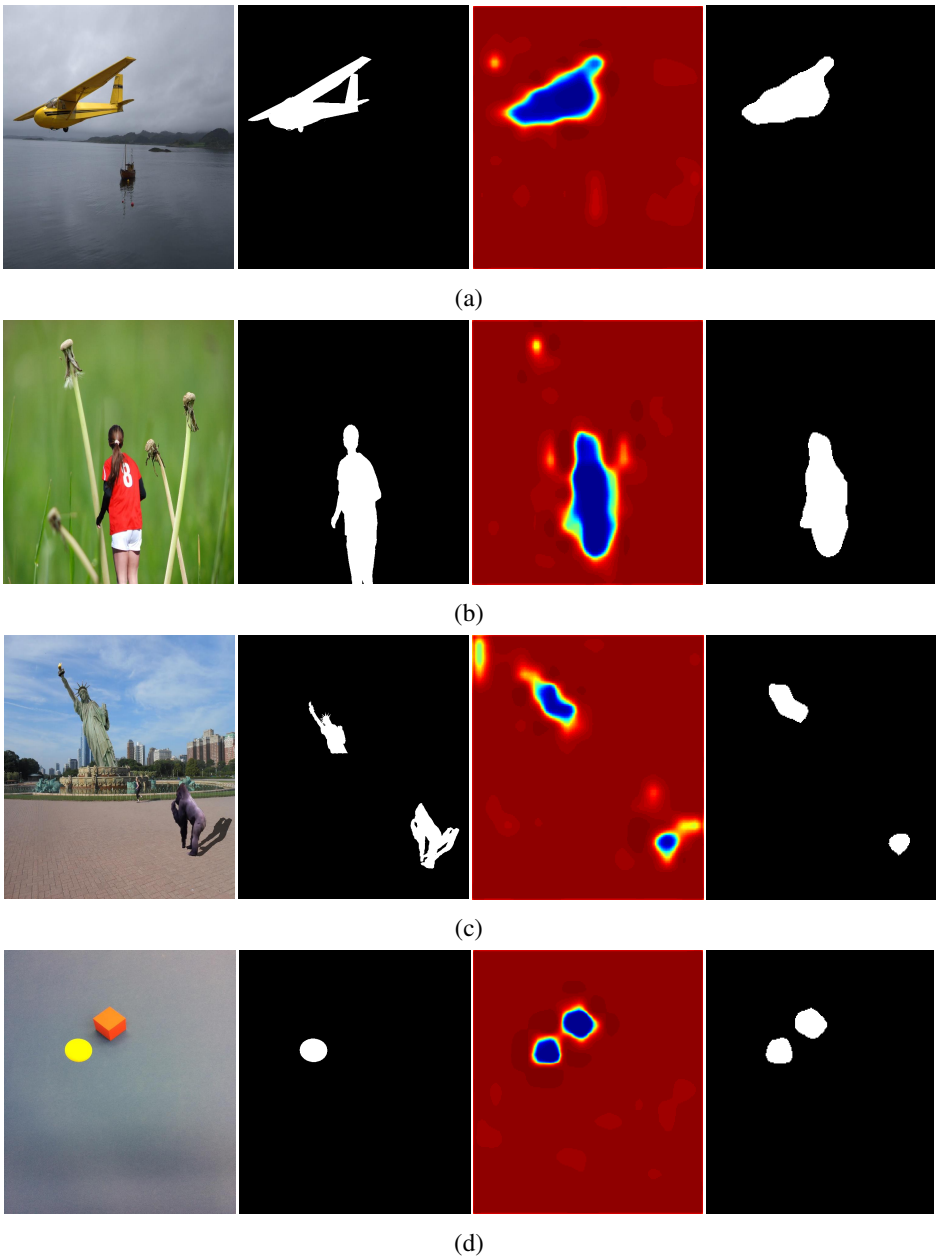


Fig. 6. This figure illustrates some segmentation results on NIST'16 [2] dataset. First and second columns represent input image and ground-truth mask for tampered region. Third and fourth columns delineate probability heat map and predicted binary mask.

4 Experiments

In this section, we demonstrate our experimental results for segmentation of manipulated regions given an image. We evaluate our proposed model on two challenging datasets- NIST'16 [2], IEEE Forensics Challenge [1] and COVERAGE [86] datasets.

4.1 Datasets

4.1.1 Creation of Synthesized Data

As deep learning networks are extremely data hungry, there is a need to collect images for training and testing the networks. For training, we will need plentiful examples (usually tens of thousands) of both manipulated and non-manipulated images. Towards this goal, we create approximately $65k$ manipulated images in order to train the proposed network discussed in Sec. 3. This network will be referred to as 'Base-Model'. The 'Base-Model' will then be fine-tuned with the NIST'16 [2] and IEEE Forensics Challenge [1] datasets. Below we explain the innovation in the collection of the manipulated image set.

In the synthesized dataset, we have focused on mainly object splicing (additions/subtractions) manipulation. The major challenge of creating manipulated images was to obtain segmented objects to insert into an image. For this we used the MS-COCO [51], which is largely used for object detection and semantic segmentation, to obtain segmented objects across a variety of categories. We extracted the objects from MS-COCO [51] images using image masks provided in ground-truth. Finally, these objects are used to create manipulation from the images of DRESDEN [33] and NIST'16 [2]. To attempt to emulate a copy-move attack in some cases we spliced multiple version of the same object onto an image, however the difficulty in obtaining segmented object automatically makes it infeasible to perform automated synthesis of copy-move attacks. Please note that we use only non-

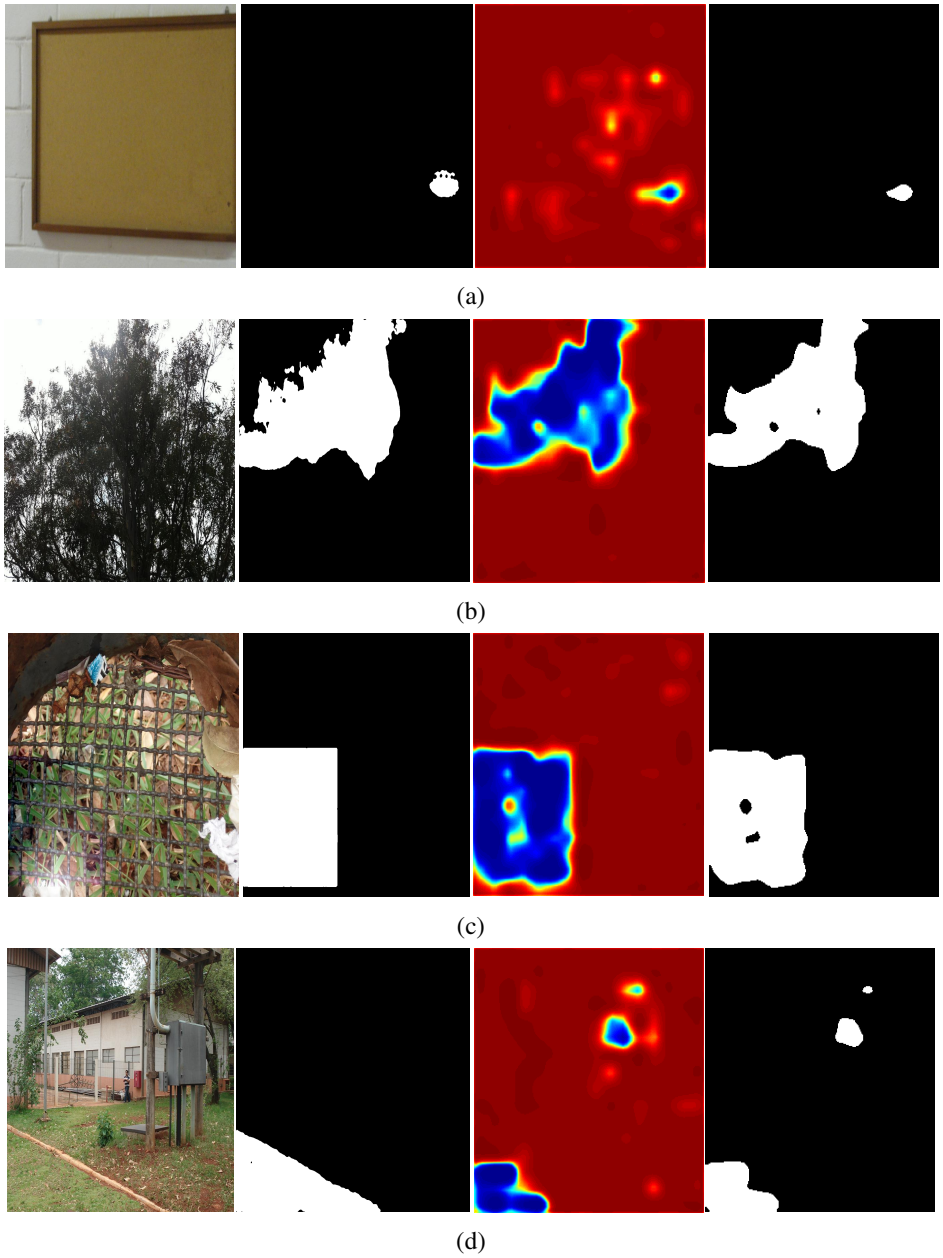


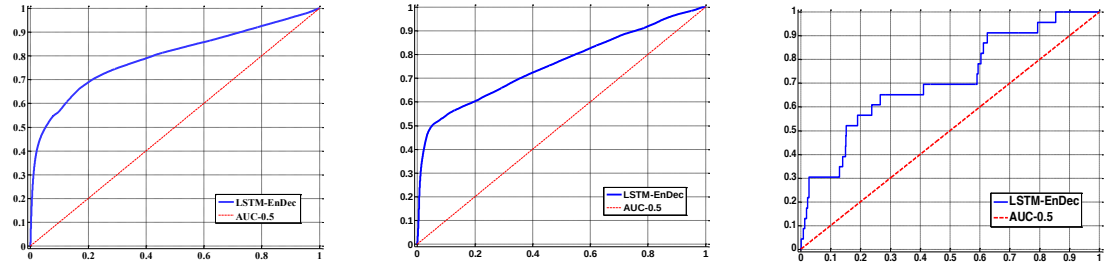
Fig. 7. Some segmentation examples on IEEE Forensics Challenge [1] dataset are shown in this figure. First and second columns are input images and ground-truth masks for manipulated regions. Third and fourth columns demonstrate the probability heatmap and predicted binary mask.

manipulated images from NIST'16 dataset to create manipulation.

To create a new manipulated image, we followed the steps below.
(1) For each raw image in the DRESDEN [33], we cropped each of the image's corners to extract a 1024×1024 patch. This method avoids resizing which introduces additional image distortions.
(2) For each of these image patches we spliced on six different objects, from the MS-COCO, to create six splice manipulated images.
(3) In order to create diverse splicing data, we spliced the same object onto the patch twice with different scaling and rotation factor, while ensuring no overlap as shown in Fig. 5.
This entire process was automated allowing us to generate tens of thousands of images in less than a day with no human interaction. Using the DRESDEN image database as the source of non-manipulated images we were able to produce approximately $40k$ images and an additional $25k$ using the DRESDEN and NIST'16 datasets respectively. The scale of our data is a hundred fold increase over most datasets that offer similar types of manipulations, which allows us to train a deep learning model. Our synthesized data also has a relatively high resolution. We can see how our dataset compares to similar datasets in table 1. With this newly generated data, we trained the 'Base-Model'. The base model predicts manipulated region at pixel level given an image.

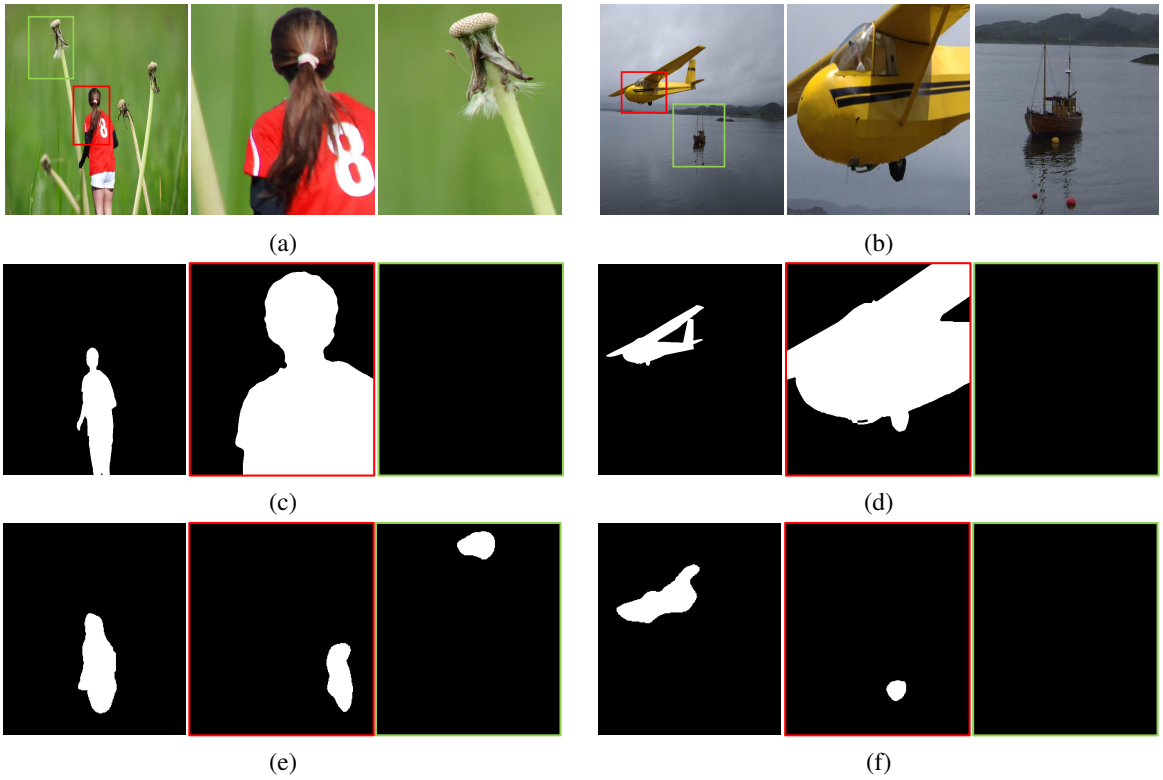
4.1.2 *Dataset Preparation*

In order to evaluate our model, we chose three datasets which provided ground-truth mask for manipulated regions. NIST'16 [2] is a very challenging dataset, which includes three main types of manipulation - (a) copy-clone, (b) removal, and (c) splicing. This



(a) NIST'16 [2] (b) IEEE Forensics Challenge [1] (c) COVERAGE [86]

Fig. 8. The figures demonstrate ROC plots on NIST'16 [2], IEEE Forensics Challenge [1] and COVERAGE [86] datasets respectively. Each curve has area under the curve (AUC), which are provided in Table 3.



(a) (b) (c) (d) (e) (f)

Fig. 9. This figure demonstrates the segmentation performance with patches as input on NIST'16 [2] dataset. First column of (a) represents the input image. Second and third columns of (a) delineate the patches as shown in the bounding boxes of input image (first column). Figures (c,d) and (e,f) are corresponding ground-truth mask and predicted binary mask.

recently released dataset includes images, which are tampered in a sophisticated way to beat current state-of-the-art detection techniques. We also show our results on the IEEE Forensics Challenge [1] dataset which provides ground-truth mask for manipulation. As manipulated regions are small in number compared to non-manipulated regions, we also perform data augmentation in order to get rid of bias in training. In addition, we choose COVERAGE [86] to demonstrate the performance of our proposed model on copy-move manipulation.

In data preparation, we first split the whole image dataset into three subsets- training (70%), validation (5%) and testing (25%). These subsets are chosen randomly. In order to increase the training data, we extract bigger patches from the four corners of the image. One additional patch is also extracted from center location of the image. We crop patches with size 1024×1024 from NIST'16 [2] training images to optimize the parameters of our architecture. The spatial resolution of IEEE Forensics Challenge [1] dataset is comparatively low. So, we extract 512×512 size of patches for IEEE Forensics Challenge [1] dataset. These newly generated images usually contain partial manipulated objects when compared to original images. We only perform data augmentation on training set, not in validation and test set. As the image and corresponding ground-truth mask are the same size, we can easily generate the ground-truth masks for the extracted image patches. With these newly generated ground-truth masks and patches, we train the whole network end-to-end. For COVERAGE [86] dataset, we do not train the proposed network due to small number of samples.

4.2 Experimental Analysis

In this section, we will discuss the implementation and evaluation criterion of our model. We also compare our model with different state-of-the-art methods for segmentation of manipulated regions.

Implementation Details. We implement our proposed framework in TensorFlow. In order to expedite our computational load, we utilize multi-GPU setting. We use two NVIDIA Tesla $K80$

TABLE 2
The table shows the pixel-wise accuracy on NIST'16 [2], IEEE Forensics Challenge [1] and COVERAGE [86] datasets for image tamper segmentation.

Methods	NIST'16 [2]	IEEE [1]	COVERAGE [86]
FCN [55]	74.28%	–	–
Encoder-Decoder [6]	82.96%	–	–
J-Conv-LSTM-Conv [7]	84.60%	77.67%	81.14%
LSTM-EnDec-Base	91.36%	88.24%	88.76%
LSTM-EnDec	**94.80%**	**91.19%**	–

TABLE 3
AUC Comparison against existing approaches on NIST'16 [2], IEEE [1] and COVERAGE [86] datasets.

Methods	NIST'16 [2]	IEEE [1]	COV. [86]
DCT Histograms [52]	0.545	–	–
ADJPEG [13]	0.5891	–	–
NADJPEG [13]	0.6567	–	–
PatchMatch [24]	0.6513	–	–
Error level analysis [57]	0.4288	–	–
Block Features [49]	0.4785	–	–
Noise Inconsistencies [60]	0.4874	–	–
J-Conv-LSTM-Conv [7]	0.7641	0.7238	0.6137
LSTM-EnDec	**0.7936**	**0.7577**	**0.7124**

GPUs to perform different sets of experiments, which will be discussed next.

Evaluation Criterion. In order to evaluate our model, we use pixel-wise accuracy and receiver operating characteristic (ROC) curve. ROC curve measures the performance of binary classification task by varying the threshold on prediction score. The area under the ROC curve (AUC) is computed from the ROC curve that measures the distinguishable ability of a system for binary classification. The AUC value typically lies in between 0 and 1.0. The AUC with 1.0 is sometimes referred as perfect system (no false alarm).

Experimental Setup. In this paper, we setup few experiments to evaluate our proposed architecture. They are (1) performance of the proposed model, (2) performance with different baseline methods, (3) comparison against existing state-of-the-art approaches, (4) ROC curve, (5) qualitative analysis, and (6) impact of global context.

Baseline Methods: In this section, we will introduce some baseline methods. We implement and compare against these methods. The various baseline methods are described below.
◇ *FCN* : Fully convolutional network as presented in [55].
◇ *J-Conv-LSTM-Conv*: This method utilizes LSTM network and convolutional layers for segmentation as in [7].
◇ *Encoder-Decoder*: This method utilizes convolutional network as encoder and deconvolution as decoder, proposed in [6].
◇ *EnDec*: Similar to encoder-decoder [6] with upsampling factor of 4 in deconvolution.
◇ *LSTM-EnDec-Base*: Proposed architecture as shown in Fig. 2 trained on Synthesized dataset discussed in Sec. 5
◇ *LSTM-EnDec*:Finetuned model of proposed architecture as shown in Fig. 2

4.2.1 Performance of the Proposed Model.

We test our proposed model on three datasets- NIST'16 [2], IEEE Forensics Challenge [1] and COVERAGE [86]. We first train our model with synthesized data (discussed in Sec. 5). We refer this model as 'LSTM-EnDec-Base' model. The LSTM-EnDec-Base model is finetuned with training sets from NIST'16 [2], IEEE Forensics Challenge [1] datasets. We obtain two finetuned models for NIST'16 and IEEE Forensics Challenge datasets respectively. As the number of images in COVERAGE [86] dataset is small, we do not perform any finetuning. Table 2 shows pixel-wise classification accuracy on segmentation task. 'LSTM-EnDec-Base' model learns good discriminative properties between manipulated vs non-manipulated pixels. Finally, finetuning this 'LSTM-EnDec-Base' model provides a boost in performance for labeling tamper class at pixel level. From the table, we can see that proposed model 'LSTM-EnDec' outperforms 'LSTM-EnDec-Base' model by 3.44%, and 2.95% on NIST'16 [2], IEEE Forensics Challenge [1] datasets respectively.

4.2.2 Performance with Different Baseline Methods.

In semantic segmentation, some recent architectures such as fully convolutional netowork (FCN) [55] and Encoder-Decoder (SegNet) [6] have successfully exploited. In this paper, we implement and train these deep architectures with image manipulation data to compare the performance of our model. We can see from Table. 2 that convolutional neural network based model such as FCN, and SegNet does not perform well compared to proposed architecture for tamper localization. It is because these models try to learn the visual concept/feature from an image whereas manipulation of an image does not leave any visual clue. We empirically observe that FCN and SegNet prone to misclassify for copy-clone and object removal type of manipulations. LSTM-EnDec surpasses FCN and Encoder-Decoder network by 20.52% and 11.84% on NIST'16 [2] as shown in Table. 2. We also compare against the segmentation framework for tamper localization (J-Conv-LSTM-Conv) presented in [7]. The proposed network outperforms J-Conv-LSTM-Conv by large margin. The advantage of our proposed model over J-Conv-LSTM-Conv is that proposed model can learn larger context by exploiting correlation between patches. On the other hand, J-Conv-LSTM-Conv is limited to correlate between different blocks of a patch. The exploitation of both LSTM network with resampling features and spatial features using encoder, helps the overall architecture to learn manipulations better.

4.2.3 Comparison against Existing Approaches.

Some of the tamper localization techniques include DCT Histograms [52], ADJPEG [13], NADJPEG [13], PatchMatch [24], Error level analysis [57], Block Features [49], and Noise Inconsistencies [60]. Table. 3. shows the performance of these state-of-the-art methods for image tamper localization. From the table, we can observe that our framework outperforms other existing methods by large margin on NIST'16 [2] dataset. In our proposed network, resampling features are exploited to predict manipulated regions. To understand the effect of resampling features in the proposed architecture, we run an experiment without LSTM network and resampling features, which is represented as Encoder-Decoder network in Table. 2. As can be seen in Table. 2, the proposed model LSTM-EnDec outperforms Encoder-Decoder by large margin (11.84%) on NIST'16 [2] dataset.

We also compare against [91] where a two-stream Faster R-CNN network has been exploited to detect manipulated regions. [91] utilized bounding box to coarsely localize manipulated ob-

jects. In contrast, we segment out manipulated regions by classifying a pixel (manipulated/non-manipulated). Since our model does not provide bounding boxes, we exploit contour approximation method to predict a bounding box on the segmentation maps produced by the proposed model. We evaluate the performance of our method in terms of average precision (AP) on NIST'16 [2] dataset. We also generate ground-truth bounding boxes on ground-truth binary masks using contour approximation method. In some cases, the proposed method falsely classifies manipulated pixels, and the contour approximation method puts a bounding box around these small false positive pixels. In order to reduce false positive bounding boxes, we use a threshold on the area of rectangle box. In our case, we eliminate the bounding box which has area under 64. As a result, we observe significant improvement in AP score. The AP score rises from 0.825 to 0.923. The AP score of [91] is 0.934. From the above discussion, we can see that the proposed model achieves comparable results to [91] even though the network do not predict a bounding box as output.

4.2.4 ROC Curve.

Figs. 8(a,b) show the ROC plots for image tamper localization, on NIST'16 [2], IEEE Forensics Challenge [1], and COVERAGE [86] datasets respectively. These ROC curves measure the performance of binary pixel classification whether a pixel is manipulated or not. We also provide the area under the curve (AUC) results in Table 3. Our model achieves AUC of **0.7936**, **0.7577** and **0.7124** on NIST'16, IEEE Forensics, and COVERAGE datasets respectively. From the ROC curves as shown in Figs. 8(a), 8(b) and 8(c), we can see that the proposed network classifies tampered pixels with high confidence.

4.2.5 Qualitative Analysis of Segmentation.

In Figs. 6 and 7, we provide some examples showing segmentation results produced by the proposed network. Fig. 6 shows segmentation results on NIST'16 [2] dataset. Segmentation results for IEEE Forensics Challenge [1] dataset are illustrated in Fig. 7. We also provide probability heat map for localizing tampered region as shown in third column of Figs. 6 and 7. As we can see from the Figs. 6 and 7, the predicted mask can locate manipulated regions from an image with high probability. The boundary of tampered objects is affected in the segmentation results as shown in Fig. 6 (third column), the underlying reason being that image boundaries are smooth (blurred) for NIST'16 [2] dataset. However, our proposed network can still localize precisely with higher overlap compared to ground-truth mask.

4.2.6 Impact of Global Context.

In our framework, we consider images as input so that the network can exploit global context. In order to observe the effectiveness of global context, we run an experiment where we consider patches as input to the network instead of images. Fig. 9 illustrates the segmentation results with respond to the input patches. From the figure, we can see that the network can localize more precisely given an image. On the other hand, the precision of localization degrades for smaller patch as the patch misses the broader context. In case of manipulated patch as shown in Figs. 9(a) and 9(b) (middle column), proposed network detects the part of the manipulated objects. For example, *digit of the person's dress* and *wheel of a plane* are identified as manipulated as shown in Figs. 9(e) and 9(f) respectively. For the patch with non-manipulated pixels, the network may provide false alarm sometimes as demonstrated in Figs. 9(e) (third column). From this study, we can conclude that global context helps analyzing the manipulated images.

5 Conclusion

In this paper, we present a deep learning based approach to semantically segment manipulated regions in a tampered image. In particular, we employ a hybrid CNN-LSTM model that effectively classifies manipulated and non-manipulated regions. We exploit CNN architecture to design an encoder network that provides spatial feature maps of manipulated objects. Resampling features of the patches are incorporated in LSTM network to observe the transition between manipulated and non-manipulated patches. Finally, a decoder network is used to learn the mapping from encoded feature maps to binary mask. Furthermore, we also present a new synthesized dataset which includes large number of images. This dataset could be beneficial to media forensics community, especially if one wants to train a deep network. Our detailed experiments showed that our approach could efficiently segment various types of manipulations including copy-move, object removal and splicing.

6 Acknowledgement

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. The paper is approved for public release, distribution unlimited.

References

[1] IEEE IFS-TC Image Forensics Challenge Dataset. http://ifc.recod.ic.unicamp.br/fc.website/index.py.

[2] NIST Nimble 2016 Datasets. https://www.nist.gov/sites/default/files/documents/2016/11/30/should_i_believe_or_not.pdf.

[3] O. M. Al-Qershi and B. E. Khoo. Passive detection of copy-move forgery in digital images: State-of-the-art. *Forensic science international*, 231(1):284–295, 2013.

[4] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra. A sift-based forensic method for copy–move attack detection and transformation recovery. *IEEE Transactions on Information Forensics and Security*, 6(3):1099–1110, 2011.

[5] M. D. Ansari, S. P. Ghrera, and V. Tyagi. Pixel-based image forgery detection: A review. *IETE journal of education*, 55(1):40–46, 2014.

[6] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[7] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. Manjunath. Exploiting spatial structure for localizing manipulated image regions. In *ICCV*, 2017.

[8] B. Bayar and M. C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 5–10, 2016.

[9] B. Bayar and M. C. Stamm. Design principles of convolutional neural networks for multimedia forensics. In *IS&T International Symposium on Electronic Imaging: Media Watermarking, Security, and Forensics*, 2017.

[10] B. Bayar and M. C. Stamm. On the robustness of constrained convolutional neural networks to jpeg post-compression for image resampling detection. In *Proceedings of The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017.

[11] A. Bharati, R. Singh, M. Vatsa, and K. W. Bowyer. Detecting facial retouching using supervised deep learning. *IEEE Transactions on Information Forensics and Security*, 11(9):1903–1913, 2016.

[12] T. Bianchi, A. De Rosa, and A. Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

[13] T. Bianchi and A. Piva. Image forgery localization via block-grained analysis of jpeg artifacts. *IEEE Transactions on Information Forensics and Security*, 7(3):1003–1017, 2012.

[14] L. Bondi, S. Lameri, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro. Tampering detection and localization through clustering of camera-based cnn features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1855–1864, 2017.

[15] G. Breinholt and C. Schierz. Algorithm 781: generating hilbert's space-filling curve by recursion. *ACM Transactions on Mathematical Software (TOMS)*, 24(2):184–189, 1998.

[16] M. Buccoli, P. Bestagini, M. Zanoni, A. Sarti, and S. Tubaro. Unsupervised feature learning for bootleg detection using deep learning architectures. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014.

[17] J. Bunk, J. H. Bappy, T. M. Mohammed, L. Nataraj, A. Flenner, B. Manjunath, S. Chandrasekaran, A. K. Roy-Chowdhury, and L. Peterson. Detection and localization of image forgeries using resampling features and deep learning. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1881–1889, 2017.

[18] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[19] Y. Cao, T. Gao, L. Fan, and Q. Yang. A robust detection algorithm for copy-move forgery in digital images. *Forensic science international*, 214(1):33–43, 2012.

[20] I.-C. Chang, J. C. Yu, and C.-C. Chang. A forgery detection algorithm for exemplar-based inpainting images using multi-region relation. *Image and Vision Computing*, 31(1):57–71, 2013.

[21] J. Chen, X. Kang, Y. Liu, and Z. J. Wang. Median filtering forensics based on convolutional neural networks. *IEEE Signal Processing Letters*, 22(11):1849–1853, 2015.

[22] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[23] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou. An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on Information Forensics and Security*, 7(6):1841–1854, Dec 2012.

[24] D. Cozzolino, G. Poggi, and L. Verdoliva. Efficient dense-field copy–move forgery detection. *IEEE Transactions on Information Forensics and Security*, 10(11):2284–2297, 2015.

[25] D. Cozzolino, G. Poggi, and L. Verdoliva. Efficient dense-field copy-move forgery detection. *IEEE Transactions on Information Forensics and Security*, 10(11):2284–2297, Nov 2015.

[26] W. Fan, K. Wang, and F. Cayre. General-purpose image forensics using patch likelihood under image statistical models. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6, 2015.

[27] H. Farid. Exposing digital forgeries from jpeg ghosts. *IEEE transactions on information forensics and security*, 4(1):154–160, 2009.

[28] X. Feng, I. J. Cox, and G. Doërr. An energy-based method for the forensic detection of re-sampled images. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2011.

[29] X. Feng, I. J. Cox, and G. Doerr. Normalized energy density-based forensic detection of resampled images. *IEEE Transactions on Multimedia*, 14(3):536–545, 2012.

[30] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva. Image forgery localization via fine-grained analysis of cfa artifacts. *IEEE Transactions on Information Forensics and Security*, 7(5):1566–1577, 2012.

[31] C. Fillion and G. Sharma. Detecting content adaptive scaling of images for forensic applications. In *Media Forensics and Security*, 2010.

[32] R. Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, 2015.

[33] T. Gloe and R. Böhme. The dresden image database for benchmarking digital image forensics. *Journal of Digital Forensic Practice*, 3(2-4):150–159, 2010.

[34] S. A. Golestaneh and D. M. Chandler. Algorithm for jpeg artifact reduction via local edge regeneration. *Journal of Electronic Imaging*, 23(1):013018–013018, 2014.

[35] C. Guillemot and O. Le Meur. Image inpainting: Overview and recent advances. *Signal processing magazine*, 31(1):127–144, 2014.

[36] M. F. Hashmi, V. Anand, and A. G. Keskar. Copy-move image forgery detection using an efficient and robust method combining un-decimated wavelet transform and scale invariant feature transform. *AASRI Procedia*, 9:84–91, 2014.

[37] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[39] M. Jaberi, G. Bebis, M. Hussain, and G. Muhammad. Accurate and robust localization of duplicated region in copy–move image forgery. *Machine vision and applications*, 25(2):451–475, 2014.

[40] R. M. Joseph and A. Chithra. Literature survey on image manipulation detection. *International Research Journal of Engineering and Technology (IRJET)*, 2(04), 2015.

[41] P. Kakar and N. Sudha. Exposing postprocessed copy–paste forgeries through transform-invariant features. *IEEE Transactions on Information Forensics and Security*, 7(3):1018–1028, 2012.

[42] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

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

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

[45] Y. Kwon, K. I. Kim, J. Tompkin, J. H. Kim, and C. Theobalt. Efficient learning of image super-resolution and compression artifact removal with semi-local gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1792–1805, 2015.

[46] G. Li, Q. Wu, D. Tu, and S. Sun. A sorted neighborhood approach for detecting duplicated regions in image forgeries based on dwt and svd. In *IEEE International Conference on Multimedia and Expo*, 2007.

[47] H. Li, W. Luo, X. Qiu, and J. Huang. Image forgery localization via integrating tampering possibility maps. *IEEE Transactions on Information Forensics and Security*, 12(5):1240–1252, 2017.

[48] J. Li, X. Li, B. Yang, and X. Sun. Segmentation-based image copy-move forgery detection scheme. *IEEE Transactions on Information Forensics and Security*, 10(3):507–518, 2015.

[49] W. Li, Y. Yuan, and N. Yu. Passive detection of doctored jpeg image via block artifact grid extraction. *Signal Processing*, 89(9):1821–1829, 2009.

[50] Z. Liang, G. Yang, X. Ding, and L. Li. An efficient forgery detection algorithm for object removal by exemplar-based image inpainting. *Journal of Visual Communication and Image Representation*, 30:75–85, 2015.

[51] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[52] Z. Lin, J. He, X. Tang, and C.-K. Tang. Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis. *Pattern Recognition*, 42(11):2492–2501, 2009.

[53] Q. Liu and Z. Chen. Improved approaches with calibrated neighboring joint density to steganalysis and seam-carved forgery detection in jpeg images. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):63, 2015.

[54] Y. Liu, Q. Guan, X. Zhao, and Y. Cao. Image forgery localization based on multi-scale convolutional neural networks. *arXiv preprint arXiv:1706.07842*, 2017.

[55] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[56] W. Luo, J. Huang, and G. Qiu. Robust detection of region-duplication forgery in digital image. In *18th International Conference on Pattern Recognition*, 2006.

[57] W. Luo, J. Huang, and G. Qiu. Jpeg error analysis and its applications to digital image forensics. *IEEE Transactions on Information Forensics and Security*, 5(3):480–491, 2010.

[58] B. Mahdian and S. Saic. Detection of copy–move forgery using a method based on blur moment invariants. *Forensic science international*, 171(2):180–189, 2007.

[59] B. Mahdian and S. Saic. Blind authentication using periodic properties of interpolation. *Information Forensics and Security, IEEE Transactions on*, 3(3):529–538, Sept. 2008.

[60] B. Mahdian and S. Saic. Using noise inconsistencies for blind image forensics. *Image and Vision Computing*, 27(10):1497–1503, 2009.

[61] V. Manu and B. Mehtre. Visual artifacts based image splicing detection in uncompressed images. In *IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, 2015.

[62] T. M. Mohammed, J. Bunk, L. Nataraj, J. H. Bappy, A. Flenner, B. Manjunath, S. Chandrasekaran, A. K. Roy-Chowdhury, and L. Peterson. Boosting image forgery detection using resampling detection and copy-move analysis. 2018.

[63] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):124–141, Jan 2001.

[64] G. Muhammad, M. H. Al-Hammadi, M. Hussain, and G. Bebis. Image forgery detection using steerable pyramid transform and local binary pattern. *Machine Vision and Applications*, 25(4):985–995, 2014.

[65] G. Muhammad, M. Hussain, and G. Bebis. Passive copy move image forgery detection using undecimated dyadic wavelet transform. *Digital Investigation*, 9(1):49–57, 2012.

[66] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[67] L. Nataraj, A. Sarkar, and B. S. Manjunath. Adding gaussian noise to denoise jpeg for detecting image resizing. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 1493–1496. IEEE, 2009.

[68] L. Nataraj, A. Sarkar, and B. S. Manjunath. Improving re-sampling detection by adding noise. In *Media Forensics and Security II*, volume 7541, page 75410I. International Society for Optics and Photonics, 2010.

[69] L. Nataraj, A. Sarkar, and B. S. Manjunath. Improving re-sampling detection by adding noise. In *SPIE, Media Forensics and Security*, volume 7541, 2010.

[70] H. H. Nguyen, T. Tieu, H.-Q. Nguyen-Son, V. Nozick, J. Yamagishi, and I. Echizen. Modular convolutional neural network for discriminating between computer-generated images and photographic images. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, page 1. ACM, 2018.

[71] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning*, 2014.

[72] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

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

[74] C.-M. Pun, X.-C. Yuan, and X.-L. Bi. Image forgery detection using adaptive oversegmentation and feature point matching. *IEEE Transactions on Information Forensics and Security*, 10(8):1705–1716, 2015.

[75] Y. Qian, J. Dong, W. Wang, and T. Tan. Deep learning for steganalysis via convolutional neural networks. In *SPIE/IS&T Electronic Imaging*, 2015.

[76] N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen. Distinguishing computer graphics from natural images using convolution neural networks. In *Information Forensics and Security (WIFS), 2017 IEEE Workshop on*, pages 1–6. IEEE, 2017.

[77] Y. Rao and J. Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2016.

[78] S. Ryu, M. Kirchner, M. Lee, and H. Lee. Rotation invariant localization of duplicated image regions based on zernike moments. *IEEE Transactions on Information Forensics and Security*, 8(8):1355–1370, 2013.

[79] S.-J. Ryu and H.-K. Lee. Estimation of linear transformation by analyzing the periodicity of interpolation. *Pattern Recognition Letters*, 36:89–99, 2014.

[80] A. Sarkar, L. Nataraj, and B. S. Manjunath. Detection of seam carving and localization of seam insertions in digital images. In *Proceedings of the 11th ACM workshop on Multimedia and security*, 2009.

[81] D. Tralic, I. Zupancic, S. Grgic, and M. Grgic. Comofodnew database for copy-move forgery detection. In *ELMAR, 2013 55th international symposium*, pages 49–54. IEEE, 2013.

[82] L. Verdoliva, D. Cozzolino, and G. Poggi. A feature-based approach for image tampering detection and localization. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014.

[83] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville. Reseg: A recurrent neural network-based model for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016.

[84] D. Voorhies. Space-filling curves and pace-filling curves and a measure of coherence. *Graphics Gems II*, page 26, 1991.

[85] W. Wang, J. Dong, and T. Tan. Exploring dct coefficient quantization effects for local tampering detection. *IEEE Transactions on Information Forensics and Security*, 9(10):1653–1666, 2014.

[86] B. Wen, Y. Zhu, R. Subramanian, T.-T. Ng, X. Shen, and S. Winkler. Coverage - a novel database for copy-move forgery detection. In *IEEE International Conference on Image processing (ICIP)*, 2016.

[87] Q. Wu, S. Sun, W. Zhu, G. Li, and D. Tu. Detection of digital doctoring in exemplar-based inpainted images. In *International Conference on Machine Learning and Cybernetics*, 2008.

[88] Y. Zhang, L. L. Win, J. Goh, and V. L. Thing. Image region forgery detection: A deep learning approach. In *Proceedings of the Singapore Cyber-Security Conference (SG-CRC)*, 2016.

[89] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *IEEE International Conference on Computer Vision*, 2015.

[90] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, 2014.

[91] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis. Learning rich features for image manipulation detection. *CVPR*, 2018.