Detection and Localization of Image Forgeries using Resampling Features and Deep Learning

Jason Bunk[1], Jawadul H. Bappy[2], Tajuddin Manhar Mohammed[1], Lakshmanan Nataraj[1], Arjuna Flenner[3], B.S. Manjunath[1,4], Shivkumar Chandrasekaran[1,4], Amit K. Roy-Chowdhury[2], and Lawrence Peterson[3]

[1]Mayachitra Inc., Santa Barbara, California , USA
[2]Department of Electrical and Computer Engineering, University of California, Riverside, USA
[3]Naval Air Warfare Center Weapons Division, China Lake, California, USA
[4]Department of Electrical and Computer Engineering, University of California, Santa Barbara, USA

Abstract

Resampling is an important signature of manipulated images. In this paper, we propose two methods to detect and localize image manipulations based on a combination of resampling features and deep learning. In the first method, the Radon transform of resampling features are computed on overlapping image patches. Deep learning classifiers and a Gaussian conditional random field model are then used to create a heatmap. Tampered regions are located using a Random Walker segmentation method. In the second method, resampling features computed on overlapping image patches are passed through a Long short-term memory (LSTM) based network for classification and localization. We compare the performance of detection/localization of both these methods. Our experimental results show that both techniques are effective in detecting and localizing digital image forgeries.

1. Introduction

The number of digital images has grown exponentially with the advent of new cameras, smartphones and tablets. Social media such as Facebook, Instagram and Twitter have further contributed to their distribution. Similarly, tools for digitally manipulating these images have evolved significantly, and software such as Photoshop, Gimp and smartphone apps such as Snapseed, Pixlr make it very trivial for users to easily manipulate images. Several methods have been proposed to detect digital image manipulations based on artifacts from resampling, color filter array, lighting, camera forensics, JPEG compression, and many more. In this paper, we consider artifacts that arise due to resampling which is common when creating digital manipulations such as scaling, rotation or splicing. We propose two methods to detect and segment image forgeries. In the first method, we present an end-to-end system to detect and localize these digital manipulations based on Radon transform and Deep Learning. In the second, we use a combination of resampling features based on probability-maps (p-maps) and Long short-term memory (LSTM) based modeling to classify tampered patches.

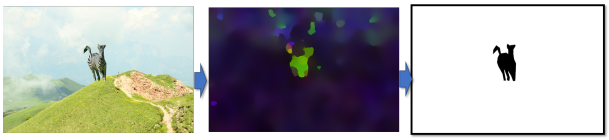


Figure 1: Illustration of our system to detect and localize digital manipulations. The manipulated image on the left is processed through a bank of resampling detectors to create a heat-map (middle), which is then used by a classifier to detect/localize the modified regions (right)

The rest of the paper is organized as follows. In Section 2, we detail some of the related work in the field of Digital Image Forensics. In Section 3, we describe our end-to-end system to detect and localize manipulations. Section 4 details the experimental results while the conclusion and future work is presented in Section 5.

2. Related Work

The field of image forensics comprises diverse areas to detect a manipulated image which includes resampling detection, detection of copy-move, splicing and object removal, JPEG artifacts, machine learning and deep learning techniques. We will briefly discuss some of them below.

In the past decade several techniques have been proposed to detect resampling in digital images [33, 34, 16, 28, 19, 15, 37]. In most cases, it is assumed to be done using linear or cubic interpolation. In [33], the authors

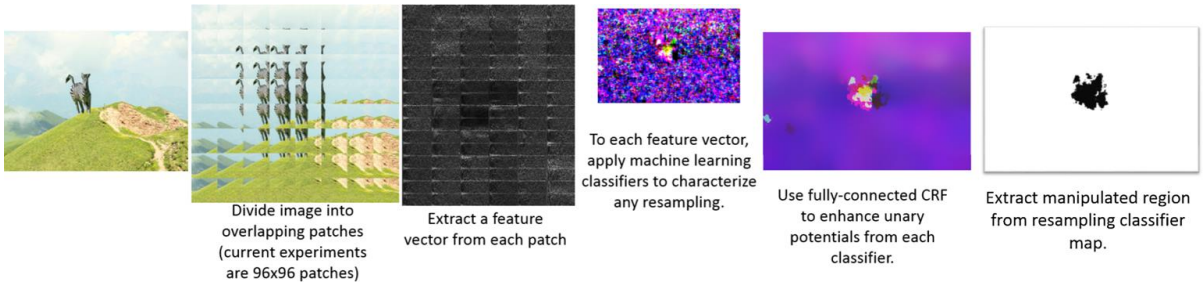


Figure 2: An end-to-end framework to detect and localize digital manipulations.

discuss how resampling introduces specific statistical correlations and show that they can be automatically detected using an Expectation-Maximization (EM) algorithm. The algorithm estimates the periodic correlations (after first detecting whether such a correlation exists) among the interpolated pixels - the specific type of the correlation indicates the exact form of the resampling. However, the EM-based method is very susceptible to JPEG attacks - especially when the JPEG quality factor (QF) is 95 or lower. The reason is that the periodic JPEG blocking artifacts interfere with the periodic patterns introduced by resampling. For images that are scaled using linear/cubic interpolation, an algorithm was proposed by analyzing the variance of the second difference of interpolated signals [16]. Although this method can detect only up-scaling, it is very robust against JPEG compression and detection is possible even at very low QFs. (Downscaled images can be detected up to a certain extent but not as robustly as upscaled images.) This method was further improved to tackle other forms of resampling using a Radon transform and derivative filter based approach [28]. In this paper, we utilize the Radon transform based method and build a feature to detect manipulated regions (see Sec. 3.1). In [19], the author showed a simpler method to improve [33] by using a linear predictor residue instead of the computationally expensive EM algorithm. We combine this method with deep learning based models to detect tampered blocks (see Sec. 3.3). In [37], the authors exploit periodic properties of interpolation by the second-derivative of the transformed image for detecting image manipulation. To detect resampling on JPEG compressed images, the authors added noise before passing the image through the resampling detector and showed that adding noise aids in detecting resampling [31, 30]. In [14, 15], a feature is derived from the normalized energy density and then SVM is used to robustly detect resampled images. Some recent approaches [17, 21] have been proposed to reduce the JPEG artifact left by compression.

Many methods have been proposed to detect copy-move [22, 11], splicing [18, 29], seam carving [38, 25] and inpainting based object removal [41, 23]. Several approaches exploit JPEG blocking artifacts to detect tampered regions [13, 24, 27, 9]. In computer vision, deep learning shows outstanding performance in different visual recognition tasks such as image classification [44], and semantic segmentation [26]. In [26], two fully convolution layers have been exploited to segment different objects in an image. The segmentation task has been further improved in [43, 3]. These models extract hierarchical features to represent the visual concept, which is useful in object segmentation. Since, the manipulation does not exhibit any visual change with respect to genuine images, these models do not perform well in segmenting manipulated regions.

Recent efforts in detecting manipulations exploit deep learning based model in [6, 4, 36]. These include detection of generic manipulations [6, 4], resampling [5], splicing [36] and bootleg [10]. The authors tested existing CNN network for steganalysis [35]. In [35], the authors propose Gaussian-Neuron CNN (GNCNN) for steganalysis. A deep learning approach to identify facial retouching was proposed in [8]. In [42], image region forgery detection has been performed using stacked auto-encoder model. In[6], a new form convolutional layer is proposed to learn the manipulated features from an image. Unlike most of the deep learning based image tampering detection methods which use convolution layers, we present an unique network exploiting convolution layers along with LSTM network.

3. Detection and Localization

In this section, we describe our end-to-end system to detect and localize manipulations in digital images. Fig. 2 shows the block schematic of our system. The various steps in framework are as follows: We start by extracting small, overlapping patches. As the first step in the processing pipeline, we compute features on each patch. These are used to characterize any resampling applied to the patch. This produces a multi-channel heatmap, one channel per resampling characteristic, at every point at which the patches were extracted. By densely extracting overlapping patches (stride of 8), we can interpret a correspondence between a pixel from the original image and the point in the heatmap

representing the patch centered at that pixel. As final steps, we postprocess this heatmap and use it to produce an image-level detection score and binarized localization map. The manipulated regions are then extracted using image segmentation methods based on Otsu's thresholding and Random Walk segmentation.

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. We choose 64x64 as a small size that we can detect reasonably well, as will be discussed in section 3.1. We extract patches with a stride of 8 for computational efficiency and to have a consistent response w.r.t. 8x8 JPEG blocks. The features in the first step start with the absolute value of a 3x3 Laplacian filter, which produces an image of the magnitude of linear predictive error. To look for periodic correlations in the linear predictor error, we apply the Radon transform to accumulate errors along various angles of projection, and then take the FFT to find the periodicity. This was proposed in [28]. We performed some additional experiments on this feature extraction stage to try to explore any potential variations that could improve these features. One improvement we found was to take the square root of the magnitude after Laplacian filtering. We also compared with using the residual from a 3x3 Median filter and found that the 3x3 Laplacian produces 4.6% better results (mean AUC over 6 binary classification tasks).

The second step is to characterize any resampling detected in the patch. We train a set of six binary classifiers that check for different types of resampling. The six resampling characteristics are: JPEG quality thresholded above or below 85, upsampling, downsampling, rotation clockwise, rotation counterclockwise, and shearing (in an affine transformation matrix). To train a model for each task, we build a dataset of about 100,000 patches extracted from about 8,000 images from two publicly available datasets of raw uncompressed images, UCID [39] and RAISE [12] datasets. Some of the patches are transformed with a set of randomly generated parameters, like multiple JPEG compressions and affine transformations, but one half of the dataset must include the transformation specified, and the other half must not. The classifiers are not mutually exclusive, and are trained individually. The best performing classifier we found for this task was an artificial neural network with two hidden layers. Using cross-validation this beat a Bayesian quadratic classifier by 6.5% on mean AUC (0.82 vs 0.77 mean AUC averaged over the six classifiers). The filtering in the third step uses bilateral filtering, which has been shown to improve region detection accuracy in other domains like semantic segmentation when fusing adjacent noisy local patch-based classifiers [20]. The fourth and final step is to obtain a mask showing manipulated regions from the filtered resampling heatmaps using proposed selective segmentation model.

3.1. Deep Neural Networks for Resampling Detection

As described in the previous section, we found that an artificial neural network with two hidden layers performed best as a binary classifier characterizing resampling in a patch for step 2 in our detection pipeline. We also explored using a newly proposed convolutional layer [6] as the first layer of an end-to-end patch classification network to learn to extract resampling features itself, rather than using our hand-crafted resampling features described before as step 1 of our pipeline. These two neural network architectures are graphed visually in Fig. 3.

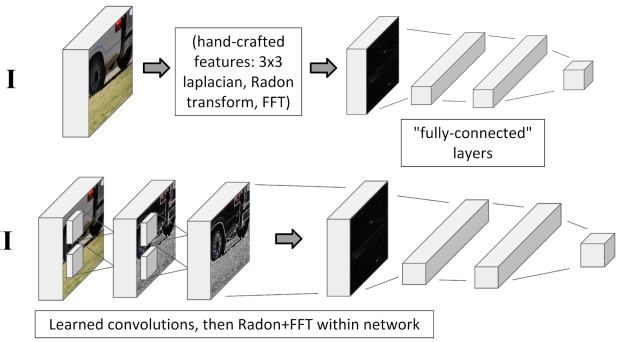


Figure 3: Deep Neural networks for detecting resampling in small patches.

We experimented with using an FFT within the network of model II, made possible by Tensorflow [2], and inspired by model I. Compared to a model that did not include the FFT, we found that the inclusion of the FFT stabilized training, reducing noise in the training loss, and caused the model to converge faster and to a slightly higher (1%) final score.

We found that the first architecture (I) using hand-crafted features performed slightly better at detecting rotation, shearing, and downsampling, but the second architecture (II) performed significantly better at detecting upsampling and differences in JPEG compression quality factors. The best ROC AUC scores of the methods are listed in the table below, and the model from which the best score was taken is listed in the rightmost column. Model II performed 1% to 3% worse on the four lower rows, which was a small but consistent difference that could be due to optimization difficulties. Model I's best ROC AUC scores for the first two rows, JPG quality and rescaling up, were 0.87 and 0.89 respectively.

The scores on patch resampling classification depend on the patch size, particularly the JPEG quality level detection and scaling up. We compared against another state-of-the-art deep learning model [7], which was trained on larger

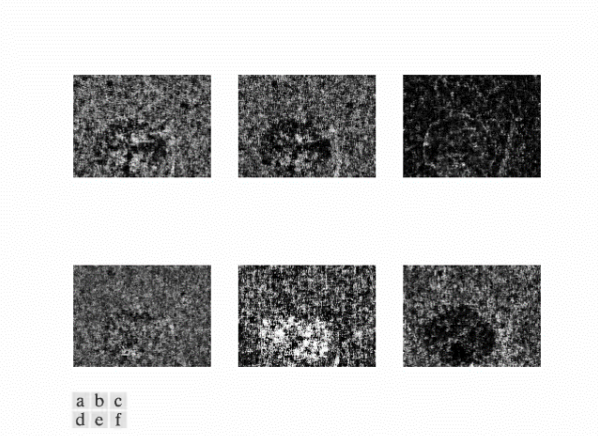


Figure 4: (a)-(f) Outputs from the neural network based resampling detector of a manipulated image

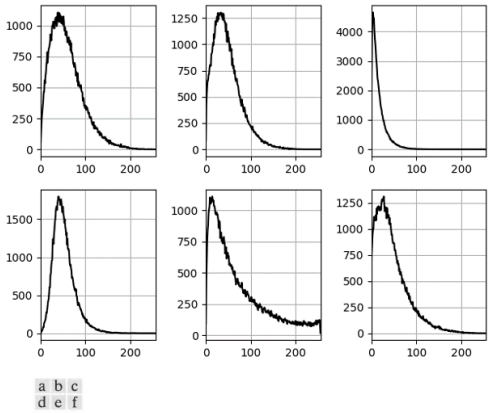


Figure 5: (a)-(f) Corresponding histograms of images in Fig. 4

	AUC	Model
JPG quality	0.93	II.
Rescale up	0.92	II.
Rescale down	0.77	I.
Rotate CW	0.81	I.
Rotate CCW	0.81	I.
Shearing	0.76	I.

Table 1: Experiments on patch classification

256x256 patch sizes. The evaluation metric they use is accuracy, while we use AUC which is a more robust evaluation metric. Averaged over JPEG quality factors from 60 to 100, the authors report 96.8% accuracy for detecting upscaling. In our experiments, we report the trend in AUC of 0.922 on 64x64 patches and AUC 0.950 on 128x128 patches. Following the trend in increasing accuracy with increasing patch size, we are at a similar level of accuracy. In addition, a fair and direct comparison would require tuning each method to use the best patch size and on the same dataset.

3.2. Mask Filtering and Segmentation

Here we describe how we filter the heatmaps and segment the manipulated regions.

The noisy feature maps (Fig. 4) can be filtered to enhance segmentation. As shown in the Fig. 5, the gray level histograms of these maps have nice properties with respect to their distribution (unimodal). This distribution can be compared to that of a special case of adding excessive gaussian-like noise to the binary images. As seen in Fig. 6, the third image from the left that was obtained after adding excessive white gaussian noise and has a similar histogram distribution to that of our feature maps. The goal is to obtain a bimodal-like distribution for the histogram of the feature maps after filtering (similar to that of the second image in Fig. 6) that will be useful for the segmentation task.

We used edge-preserving bilateral filters as inspired by their successful application in the semantic segmentation field [20, 43], a mathematically analogous task of filtering a set of noisy classifier outputs predicted on overlapping patches. The bilateral filter parameters were empirically determined from the noise distribution of various feature maps. As shown in the Fig. 8, only a subset of these six feature maps have a bimodal-like distribution in their histograms after filtering. Therefore, the ones which does not

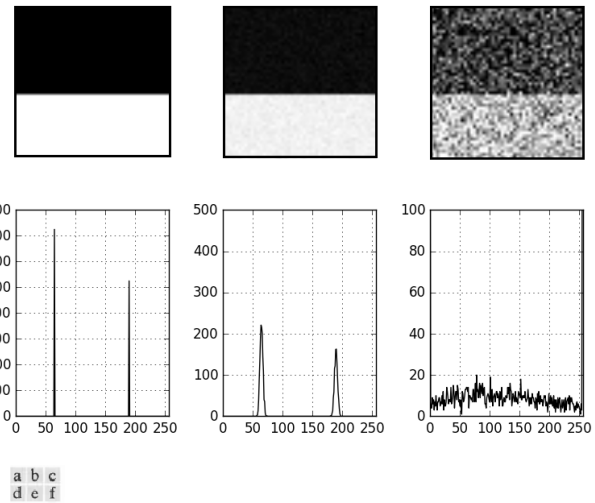


Figure 6: (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and variance 0.0001. (c) Image with additive Gaussian noise of mean 0 and variance 0.06. (d)-(f) Corresponding histograms

have such a distribution are not used for segmentation.

The feature maps that are used for the segmentation is determined by Otsu method of thresholding image histograms [32] where it can detect bimodal distributions based on the fact that this threshold minimizes the intra-class variance and maximizes the inter-class variances of two predicted classes. As seen in the Fig. 8, the Otsu threshold is either a value close to the mean of the distribution or it separates the two classes of a bimodal distribution.

A random-walker segmentation [40] is then applied to each of these maps to diffuse the corresponding pixels in between the two modes to either of the classes (class-1 being the pixel values less than the first mode and class-2 being the pixel values greater than the second mode). This probabilistic based method is useful as it locates the weak/missing boundaries and is even robust in noisier conditions. Finally, a bitwise-OR operation is used to add all these binary images to obtain a final mask image.

3.2.1 Localization based confidence score metric

Bilateral filtering is implemented using ArrayFire package in Python which uses GPU's for faster computations by making some approximations. Therefore, a gray-scale mask is generated instead of a binary mask (by considering the mean of all the binary masks for many iterations). The confidence score is generated by adding the normalized non-zero pixel values (normalized pixel value of 1 corresponds to a black or manipulated region) and averaging them over the number of non-zero pixels in the gray-scale mask image.

3.3. Patch Classification Framework

Now-a-days authentication of image tampering is a very difficult problem due to the close resemblance between the manipulated and genuine images. However, most of the image manipulations have a common characteristic, which shows discriminative features in the boundary shared between manipulated and non-manipulated regions. In this paper, we present a network which exploits resampling features along with long-short term memory (LSTM) cells in order to localize the manipulated regions.

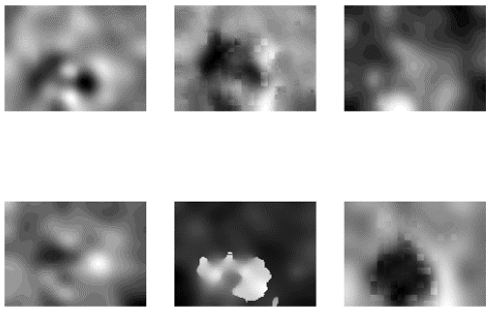
3.3.1 Resampling Features

In [33], the authors proposed the seminal method to detect resampling in digital images. We will provide a brief introduction to their approach. The idea is that resampling introduces periodic correlations among pixels due to interpolation. To detect these correlations, they use a linear model in which each pixel is assumed to belong to two classes: a resampled class and a non resampled class , each with equal probability. The conditional probability for a pixel belonging to the resampled class is assumed to be Gaussian while the conditional probability for a pixel belonging to the other class is assumed to be uniform. To simultaneously estimate a pixel's probability of being a linear combination with its neighboring pixels and the unknown weights of the combination, an Expectation Maximization (EM) algorithm is used. In the Expectation Step, the probability of a pixel belonging to the resampled class is calculated. This is used in the Maximization step to estimate the weights. The stopping condition is enforced when the difference in weights between two consecutive iterations is very small. At this stage, the matrix (same dimensions as image) of probability values obtained in the Expectation step for every pixel of the image is called the "Probability map (p-map)". For a resampled image this p-map is periodic and peaks in the 2D Fourier spectrum of the p-map indicate resampling. In the p-map, a probability value close to 1 indicates that a pixel is resampled. However, the EM algorithm based p-map is computationally expensive. In this paper, we use a simpler method to compute the p-map that first filters the image using a linear filter and then computes the residual image on which the p-map was calculated [19].

3.3.2 Long-Short Term Memory (LSTM) Network

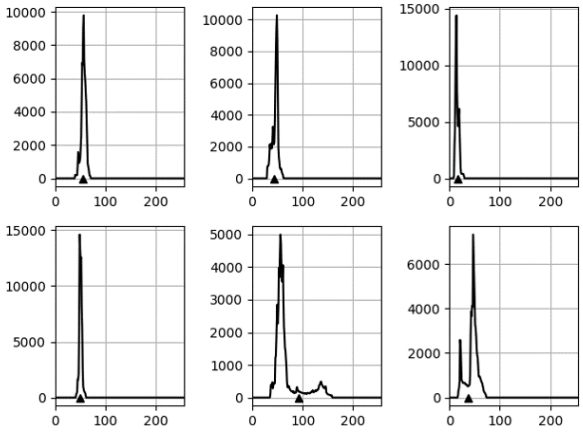
Manipulation distorts the natural statistics of an image, especially in the boundary region. In this paper, we utilize the LSTM network to learn the correlation between blocks of resampling features as shown in Fig. 9. In order to utilize LSTM, we first divide the 2D resampling feature map into blocks. We split into 8×8 blocks, where each block has size 8×8 (total 64 pixels). Now, we learn the long distance block dependency feeding each block to each cell of the LSTM in a sequential manner. The LSTM cells correlate neighboring blocks with current block. In this work, we utilize 3 stacked layers, and at each layer, 64 cells are used. In the last layer, each cell provides 256-d feature vector which is fed into softmax classifier. The key insight of using LSTM is to learn the boundary transformation between different blocks, which provides discriminative features between manipulated and non-manipulated patch. In the following, we will briefly discuss about the parameters of a LSTM cell.

The cell is an important entity to build an LSTM network. The information flow in the cells is controlled by gates. There are mainly three gates that link a cell to neighboring cell, and they are (1) input gate, (2) forget gate, and (3) output gate. Let us denote cell state and output state as C_t and z_t for current cell t. Cell state and output states are controlled by these gates. Each gate has a value ranging from zero to one, activated by a sigmoid function. These gates actually make decisions about how much information should be passed through. Higher value implies the flow of more information. Each cell produces new candidate cell state $\bar{C}_t$. Using the previous cell state $\mathcal{C}_{t-1}$ and $\bar{C}_t$, we can



a b c
d e f

Figure 7: (a)-(f) Bilateral filtered heatmaps



a b c
d e f

Figure 8: (a)-(f) Corresponding histograms of images in Fig. 7 with Otsu thresholds marked with "▲"

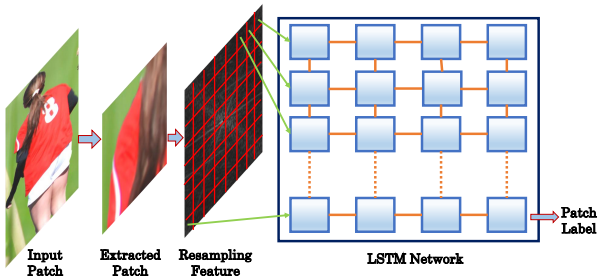


Figure 9: LSTM Framework for Patch Classification.

write the updated cell state $\mathcal{C}_t$ as

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

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

$$h_t = o_t \circ tanh(\mathcal{C}_t) \tag{2}$$

In Eqns. 1 and 2, i, f, o represent input, forget and output gates.

3.3.3 Soft-max Layer

In the proposed network, we have a softmax layer for patch classification task. For this task, the labels are predicted at the final layer of LSTM network. Given a patch of an image, we obtain the features which are used to predict the manipulated class (either patch label or pixel class) using *softmax* function. Let $\mathcal{W}$ be the parameter associated with feature. Then, $\mathcal{F}$, the softmax function can be written as

$$P(\mathcal{Y}_k) = \frac{\exp^{(\mathcal{W})^T \mathcal{F}}}{\sum_{k=1}^{N_c} \exp^{(\mathcal{W}^k)^T \mathcal{F}}} \tag{3}$$

Here, N_c is the number of classes ($\in \mathcal{R}^{2\times 1}$) - manipulated vs non-manipulated. $\mathcal{W}_L^k$ implies the weight vector associated to the class k. Using Eqn. 3, we compute the probability distribution over various classes. Now, we can predict labels by maximizing $P(\mathcal{Y}_k)$ with respect to k. The predicted label can be obtained by $\hat{\mathcal{Y}} = \arg\max_k \ P(\mathcal{Y}_k)$.

3.3.4 Training the Network

In this paper, we perform patch classification whether it is manipulated or not. We compute the cross entropy loss of this task given ground-truth patch labels.

Patch Classification. Patch labels are predicted at the end of the LSTM network as shown in Fig. 9. Let us denote $\theta_p = [\theta_1, \mathcal{W}]$, which is a weight vector associated with patch classifiation. Here, θ_1 contains the parameters of first two convlution layers and LSTM layers. $\mathcal{W}$ is the parameter at the softmax layer of patch classification. Now, we can compute the cross entropy loss for patch classification as follows.

$$\mathcal{L}_p(\theta_p) = -\frac{1}{M_p} \sum_{j=1}^{M_p} \sum_{k=1}^{N_p} \mathbb{1}(\mathcal{Y}^j = k) \log(\mathcal{Y}^j = k | x^j; \theta_p) \tag{4}$$

Here, $\mathbb{1}(.)$ is the *indicator function*, which equals to 1 if $j = k$, otherwise it equals 0. $\mathcal{Y}^j$ and x^j imply the patch label (manipulated or non-manipulated) and the feature of the sample j. M_p is the number of patches.

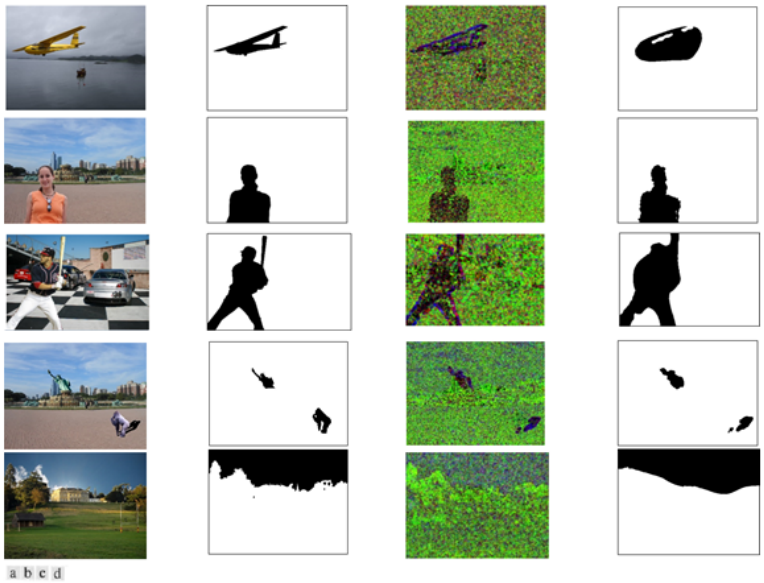


Figure 10: (a) Manipulated Image (b) Ground truth (c) Resampling detector output visualizing 3 out of 6 classifiers. Each color represents a different type of manipulation. (d) Predicted mask

4. Experimental Results

4.1. Results on Radon transformed and Deep Learning based detection and localization

In this section, we demonstrate our experimental results for two tasks-(1) identification of tampered patch, and (2) segmentation of manipulated regions given a patch. We evaluate the proposed model on NIST Nimble 2016 dataset [1] for the Media Forensics challenge. This dataset includes mainly three types of manipulation: (a) copy-clone, (b) removal, and (c) splicing. The images are tampered in a sophisticated way to beat current state-of-the-art detection techniques. The results in Fig. 10 show the efficacy of the proposed model for different images in Nimble 2016 dataset.

4.2. Results on LSTM based patch classifier

Here we again use the Nimble 2016 dataset [1] to test the efficacy of LSTM based approach. We use ROC curves to evaluate our method. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold. Fig. 11 shows ROC curve of the proposed method for patch classification. We also compare our methods on raw pixels of a patch. From the Fig. 11, we can see that utilization of resampling features boosts the performance in patch classification. We also show our classification result in Table 2. The results in Table 2 and Fig. 11 attest that the method is effective in classifying tampered patches.

Method	Classification	AUC
Proposed Method	94.86%	0.9138

Table 2: Results of patch classification for LSTM based approach

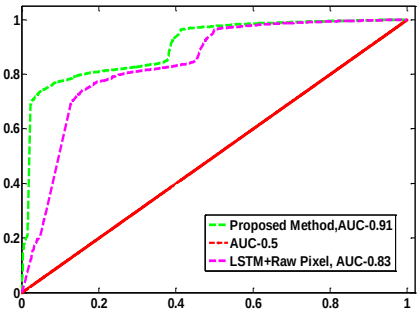


Figure 11: ROC Curve comparing LSTM based method on Pixels and Resampling Detection features

5. Conclusion

In this paper, we presented two methods to detect and localize manipulated regions in images. Our experiments showed that both Convolutional Neural Networks (CNNs) and LSTM based networks are effective in exploiting resampling features to detect tampered regions. In future, we will look into ways of combining these methods and detect image forgeries.

6. Acknowledgements

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.

References

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

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[3] 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.

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

[5] 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*. IEEE.

[6] 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.

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

[8] 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.

[9] T. Bianchi, A. De Rosa, and A. Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2444–2447. IEEE, 2011.

[10] M. Buccoli, P. Bestagini, M. Zanoni, A. Sarti, and S. Tubaro. Unsupervised feature learning for bootleg detection using deep learning architectures. In *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*, pages 131–136. IEEE, 2014.

[11] 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.

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

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

[14] X. Feng, I. J. Cox, and G. Doërr. An energy-based method for the forensic detection of re-sampled images. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[15] 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.

[16] A. Gallagher. Detection of linear and cubic interpolation in JPEG compressed images. In *Computer and Robot Vision, 2005. Proc. The 2nd Canadian Conference on*, pages 65–72, May 2005.

[17] 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.

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

[19] M. Kirchner. On the detectability of local resampling in digital images. volume 6819, page 68190F. SPIE, 2008.

[20] V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst*, 2(3):4, 2011.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[27] 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.

[28] 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.

[29] 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.

[30] L. Nataraj, A. Sarkar, and B. S. Manjunath. Adding gaussian noise to "denoise" JPEG for detecting image resizing. In *ICIP*, 2009.

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

[32] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

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

[34] S. Prasad and K. Ramakrishnan. On resampling detection and its application to detect image tampering. In *2006 IEEE International Conference on Multimedia and Expo*, pages 1325–1328. IEEE, 2006.

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

[36] Y. Rao and J. Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016.

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

[38] 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*, pages 107–116. ACM, 2009.

[39] G. Schaefer and M. Stich. Ucid: An uncompressed color image database. In *Electronic Imaging 2004*, pages 472–480. International Society for Optics and Photonics, 2003.

[40] A. K. Sinop and L. Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. *ICCV 2007*, pages 1–8, 2007.

[41] Q. Wu, S.-J. Sun, W. Zhu, G.-H. Li, and D. Tu. Detection of digital doctoring in exemplar-based inpainted images. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 3, pages 1222–1226. IEEE, 2008.

[42] 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: Cyber-Security by Design*, volume 14, page 1, 2016.

[43] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[44] 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*, pages 487–495, 2014.