Universal Image Manipulation Detection using Deep Siamese Convolutional Neural Network

Aniruddha Mazumdar*, Jaya Singh, Yosha Singh Tomar, and Prabin Kumar Bora
Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati, Assam, India - 781039
Email: *m.aniruddha@iitg.ac.in

Abstract—**Detection of different types of image editing operations carried out on an image is an important problem in image forensics. It gives the information about the processing history of an image, and also can expose forgeries present in an image. There have been few methods proposed to detect different types of image editing operations in a single framework. However, all the operations have to be known *a priori* in the training phase. But, in real-forensics scenarios it may not be possible to know about the editing operations carried out on an image. To solve this problem, we propose a novel deep learning-based method which can differentiate between different types of image editing operations. The proposed method classifies image patches in a pair-wise fashion as either similarly or differently processed using a deep siamese neural network. Once the network learns feature that can discriminate between different image editing operations, it can differentiate between different image editing operations not present in the training stage. The experimental results show the efficacy of the proposed method in detecting/discriminating different image editing operations.**

Index Terms—**Image Forensics, Deep Learning, CNN.**

I. INTRODUCTION

With the availability of various image editing software, it has become possible to create visually plausible image forgeries with a minimal effort. Because of this, a large number of forged images are now available on the web. These images when used on different platforms, like the electronic and social media, may create tensions in the society. These concerns necessitate the development of image forensics techniques for checking the authenticity of images before using them as critical information.

While creating a forgery, the forged parts are often processed through different image editing operations to make them look visually plausible. For example, in image *splicing* forgery the spliced objects go through different image editing operations, e.g. resizing, rotating, smoothening, contrast enhanecment, compression. Although imperceptible to human eyes, every image editing operation leaves a unique trace of manipulation. These traces are utilized by researchers to detect different types of editing operations performed on images. Different techniques have been proposed to extract features related to the traces left by different editing operations, and which are utilized to check the authenticity of images. For example, in [1], [2], [3] the authors extracted features for detecting the traces left by resizing and resampling operations, in [4], [5], [6] features related to median filtering traces are

extracted, in [7], [8] features are extracted to detect contrast-enhancement operation, and in [9] JPEG artifact related features are extracted for forgery detection.

Although these methods are good at detecting splicing, methods from each of the categories work only under their respective assumptions about the traces of manipulations left by the forgery process. For example, the median filtering detection methods cannot detect traces left by the resampling operation. To handle this limitation, researchers have focused to develop universal forensics methods, which can detect multiple manipulations in a single framework. The first universal forensics method was proposed by Qiu *et al.* [10], where different steganalysis features were used to detect different types of image processing operations. The method is based on the observation that different image editing operations destroy the natural statistics of the image pixels present in an authentic image in the same way steganography methods do while manipulating the pixels for embedding a message. Fan *et al.* [11] proposed another general-purpose forensics method for detecting different types of image editing operations. The authors proposed to create a Gaussian mixture model (GMM) of image patches corresponding each editing operation. Then, the average log-likelihood of patches under the different GMMs corresponding to different classes are compared to decide the class of the patches.

Inspired by the success in other computer vision areas, the forensics community has recently focused on applying deep learning-based methods for image manipulation detection. Chen *et al.* [12] proposed the first deep learning-based median filtering detection method. This is the first deep learning-based image manipulation detection method, where the first layer computes the median filtering residual and the subsequent layers extract and classify the features useful for median filtering detection. Bayar and Stamm [13] proposed a deep learning-based universal forensics method for detecting different types of image manipulating operations. The image editing features are automatically learned from the training data by employing a convolutional neural network (CNN) [14]. The authors proposed a new convolutional layer, which suppresses the image content and enhances features important for detecting different editing operations.

Although these universal manipulation detection methods perform really well, all the manipulation operations have to be known before training the network. However, there is a large number of image editing operations available in the image

editing software, e.g. Adobe Photoshop and GIMP. Also, new image editing operations are being developed and incorporated in these editing software. In addition to that, there may be multiple editing operations performed subsequently to make the spliced parts look similar to the authentic parts. Therefore, it is not practical to incorporate all the editing operations in the training process as required by the existing univeral manipulation detection methods, i.e. [10], [11], [13]. This necessitates the developement of universal forensics method which can not only detect the different image editing operations present in the training stage, but also is capable of generalizing to editing operations not present in the training stage.

This paper proposes a novel deep learning-based forensics method for detecting image manipulations. The proposed method takes two image patches as input, and check whether they come from the same or different manipulation operations. The proposed method is built upon the work of Bayar and Stamm [13], where they showed that CNNs are capable of learning accurate image editing features automatically from the training data. However, instead of learning features to classify image patches to different manipulation classes, the proposed method learns the features which can discriminate different image editing operations. The reason for this is that from the forensics point of view it is more informative to check whether two image patches have undergone the same type of manipulations or not than classifying individual image patch. For this, the proposed method employs a deep siamese CNN, which has twin CNNs accepting two image patches as the input and classifies the patch pair as either identically processed (IP) or differently processed (DP).

II. BACKGROUND

In this section, we explain the deep learning-based universal forensics method [13] proposed by Basar and Stamm as the background. The method is based on the assumption that each image editing operation leaves behind the trace of the particular manipulation. These traces can be detected by examining the relationship between the neighbouring pixels, as any image manipulation destroys the natural statistics of pixels and modifies them in a unique way [10]. To automatically learn the features useful for the detection different manipulation operations, the authors of [13] proposed to employ a CNN.

A CNN is a special type of neural networks originally proposed for handwritten digit recognition [14]. Since then, they have been successfully used in many other computer vision problems with some variations in its architecture [15], [16]. A CNN typically contains a stack of multiple layers with nonlinearities which enable it to learn different features from the training data itself. The typical layers present in a CNN are the convolutional layer, the pooling layer, and the fully connected layer. The convolutional layer contains several filters which convolve with the input image in parallel and the element-wise rectified linear unit (ReLU) for non-linear mapping. The output of the convolutional layer is called the feature map, and given by

$$\mathbf{f}_k^l = \max(0, \sum_j \mathbf{f}_j^{l-1} * \mathbf{w}_{kj}^l + \mathbf{b}_k^l) \tag{1}$$

where, $*$ is the convolution operation, $\mathbf{f}_k^l$ is the kth feature map in layer l, $\mathbf{w}_{kj}^l$ is the filter connecting the jth feature map in layer $l-1$ to the kth feature map in layer l and $\mathbf{b}_k^l$ is the bias for the kth feature map in layer l. The convolutional layer is followed by a max-pooling layer, which reduces the size of each feature map by taking the maximum value over a region. The stacking of the convolutional and pooling layers one after another enables the CNNs to learn different levels of features at different layers. The initial layers learn low-level features and the final layers learn more problem-specific features. To classify the final features, one or more fully-connected layers are stacked on the top of the final convolutional and pooling layers. The sigmoid non-linearity is used in the fully-connected layer, producing the output in the range of $[0, 1]$. The parameters $\mathbf{w}$ and $\mathbf{b}$ are learnt in the training process using the standard gradient descent-based backpropagation technique [14].

CNNs have proved to be very good in different computer vision tasks, e.g. object detection and recognition [15]. However, they did not perform well when applied directly to image manipulation detection [17]. This is because the conventional CNNs capture the image content rather than important forensics features. To suppress the image content and enhance the relationship between neighbouring pixels, Bayar and Stamm proposed a new convolutional layer as the first layer of the CNN. The filters in this new convolutional layer are constrained to learn a set of prediction error filters. The concept of using the prediction error filters in the first layer is motivated from different image forensics and steganalysis methods. Steganalysis methods like the rich models [18] and the subtractive pixel adjacency matrix (SPAM) [19] utilise this concept of using different prediction filters for computing the prediction errors, which are later used as features to detect hidden messages present in the stago images. In forensics, Chen *et al.* [17] proposed a similar strategy to first extract the median filtering residuals and then using a CNN for the detection of the median filtering operation.

In [13], the filters in the first convolutional layer of the CNN are forced to learn a set of prediction error filters by constraining the weights in each of the K filters as

$$\begin{aligned} w_k^1(0,0) &= -1 \\ \text{and } \sum_{l,m \neq 0} w_k^1(l,m) &= 1 \end{aligned} \tag{2}$$

where, $w_k^1(l,m)$ denotes the weight at position (l,m) of the kth filter and $w_k^1(0,0)$ denotes the weight at the center of the corresponding filter kernel. This procedure is repeated for all the pixels in the image patch by moving the kernels throughout the patch. This prediction error layer extracts the local dependency of pixels with its neighbours, which is the important information from the forensics point of view [10].

Using this approach, Bayar and Stamm detected four different types image editing operations, namely the Gaussian filtering, the median filtering, the corruption of the image by the AWGN and resampling, with good accuracies.

III. PROPOSED METHOD

The universal forensics method proposed by Bayar and Stamm [13] shows that CNNs can automatically learn features important for detecting different image editing operations. However, there are some limitations of the method. For training the CNN, all the image editing operations should be known *a priori*. If an image is edited with an operation other than the ones used for training, the method will not be able to detect it properly. Moreover, in a real forgery, there may be multiple editing operations performed on the image subsequently. In this case also, the method [13] will fail as the CNN is trained to classify the images only to one of the many editing operations, not combinations of different manipulations.

To overcome these limitations, we propose a method which checks whether two image patches have undergone the same operation or two different operations. For this, we employ a deep siamese CNN which takes a pair of image patches as input and classify these as either IP or DP. The reasons for the pair-wise classification image patches are as follows:

1) The spliced regions in an image may go through different image editing operations than the authentic regions. Therefore, from the forensics point of view, it is more informative to know whether all the patches of an image have been manipulated in the same way or not.

2) The pair-wise classification of patches removes the necessity of classifying the patches into different manipulation classes. This is an important advantage of the proposed method, as it allows the proposed method to classify image patches coming from a different type of manipulation not considered in the training stage.

3) Since, the methods [10], [11] and [13] classify the image patches into one of the different but fixed types of manipulations, they are more vulnerable to anti-forensics. This is because the anti-forensics methods can be developed to hide the traces left by each of the operations considered in these methods [20]. On the other hand, the proposed method does not learn any class-specific feature as it is designed to check whether two patches have undergone the same type of manipulation or not. Hence, developing anti-forensics techniques to counter the proposed method will be more difficult.

Figure 1 shows the block diagram of the proposed framework. It has twin neural networks CNN1 and CNN2 sharing the same set of weights. It accepts two input images, which are independently processed by CNN1 and CNN2 and then a distance layer [21] computes a distance metric between the outputs of the twin networks. Because of the sharing of weights, CNN1 and CNN2 map two similar input images to very close points in the feature space. The proposed siamese CNN automatically learns the features that can check whether a pair of images has been similarly or differently manipulated.

A. Network Architecture

1) CNN: As in [13], the first convolutional layer in each of CNN1 and CNN2 is a constrained convolutional layer. The
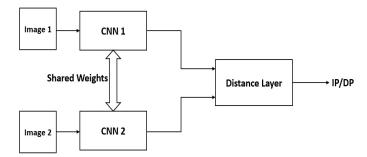

Fig. 1: Framework of the siamese network to classify patch pairs as IP or DP.

filters in the constrained convolutional layer are forced to learn a set of prediction error filters, which suppress image contents and produce prediction error. Each of CNN1 and CNN2 has the architecture shown in Figure 2. It contains 3 convolutional layers, 2 max-pooling layers and 3 fully-connected layers. The block diagram of the CNN is shown in Figure 2. The first convolutional layer is the constrained convolutional layer [13] with 16 prediction error filters of size 5×5 and stride 1. Its weights follow the constraints given by Equation (2). This layer is followed by an unconstrained convolutional layer with 64 filters of size 7×7 with stride 2. The ReLU nonlinearity is applied element-wise to the output of this layer followed by the max-pooling layer with a kernel size 3×3 and stride 2. The output of this layer is fed to another unconstrained convolutional layer with 128 filters of size 3×3 and stride 1. The ReLU nonlinearity is applied element-wise to the output of this layer. It is followed by a max-pooling layer with a kernel size 3×3 and stride 2. This layer is followed by three fully-connected layers with 4096, 4096 and 2048 neurons respectively. The sigmoid non-linearity is used in each of these layers. The neurons in the fully-connected layers are dropped out [22] with a probability of 0.5 at each iteration of the training process. The output of the final fully-connected layer represents the features learned by the CNN.

2) Distance Layer: Given a pair of image patches $\mathbf{x}_1$ and $\mathbf{x}_2$ as input, CNN1 and CNN2 compute the feature vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ respectively. A distance layer computes a distance metric between them, which is then fed to a single sigmoidal output neuron. This neuron computes the prediction of the input image patch pair as $p = \sigma(\sum_j \alpha_j |f_1(j) - f_2(j)|)$, where σ is the sigmoid non-linearity function and α_j is a learnable parameter representing the importance of each component of the feature vectors in the classification of the patch-pair.

B. Learning

The proposed siamese network is a binary classifier with label $y(\mathbf{x}_1, \mathbf{x}_2) = 1$ when both input image patches $\mathbf{x}_1$ and $\mathbf{x}_2$ come from the same manipulation class, and $y(\mathbf{x}_1, \mathbf{x}_2) = 0$ when $\mathbf{x}_1$ and $\mathbf{x}_2$ come from two different manipulation classes. The network is trained by minimising the average cross-entropy loss function C over a batch of pairs given by [23]
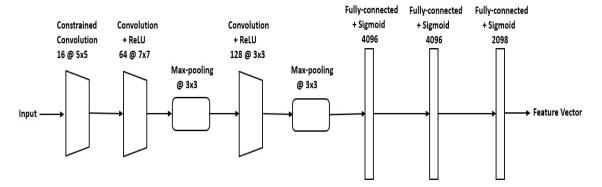
Fig. 2: The CNN architecture used in the proposed siamese network.

$$C = \frac{1}{M} \sum_{i=1}^{M} y(\mathbf{x}_1^i, \mathbf{x}_2^i) \log p(\mathbf{x}_1^i, \mathbf{x}_2^i)$$
$$+ (1 - y(\mathbf{x}_1^i, \mathbf{x}_2^i)) \log(1 - p(\mathbf{x}_1^i, \mathbf{x}_2^i)) \tag{3}$$

where M is the number of images in each batch. The parameters of the network are learnt in the training phase by minimising C using the stochastic gradient descent (SGD)-based backpropagation technique. In SGD method, the weights are updated in each iteration using the following equations:

$$\mathbf{w}_{kj}^l = \mathbf{w}_{kj}^l + \Delta \mathbf{w}_{kj}^l$$
$$\Delta \mathbf{w}_{kj}^l = \mu \Delta \mathbf{w}_{kj}^l - \eta \nabla_{\mathbf{w}_{kj}^l} C - \lambda \mathbf{w}_{kj}^l \tag{4}$$

where, $\nabla_{\mathbf{w}_{kj}^l}$ is the gradient of C with respect to the weight matrix $\mathbf{w}_{kj}^l$, η is the learning rate, μ is the momentum and λ is the L_2 regularization term.

Once, the network is trained, it is used to detect/discriminate the different image processing operations.

IV. EXPERIMENTS AND RESULTS

To train and test the proposed method, a dataset was created using the unprocessed raw images taken from the Dresden Image Database [24]. The database contains more than $14,000$ images with resolutions of about 2000×3000 captured by 73 different digital cameras. A set of 1566 raw images were compressed in the JPEG format with 100% quality factor (QF) and converted to grayscale images by considering only the green channel of the images. We cropped image patches of size 150×150 from these images, resulting in $114,000$ unaltered image patches.

The proposed system was implemented using the Python-based Keras [25] deep learning library on a Tesla K20c GPU with 5 GB of RAM. The Nadam optimiser [26] was used with the parameters set as: *learning rate*(η) = 0.002, *momentum*(μ) = 0.002 and *decay* = 0.005 and *regularization term*(λ) = 0.0001. We have used the learning rate decay technique to converge to the minimum of C by reducing the fluctuations [27]. The training batch size was set to 16 images. We have used the batch normalisation technique [28] as it helps in achieving faster convergence and higher generalisation accuracy.

A. Manipulation Detection Results

To test the performance of the proposed siamese network in detecting/discriminating different image editing operations, we have carried out a series of experiments. For this, five different versions of the $114,000$ unaltered patches are created by editing them with the following operations: Gaussian blurring, median filtering, resampling, corrupting with the additive white Gaussian noise (AWGN), gamma correction. The details of the manipulations are listed in Table I. This way, we obtain $114,000$ patches from the altered as well as from each of the editing operations.

In the first experiment, we have trained the network using the image patches coming from four different classes: original, Gaussian blurring, median filtering, and resampling. We randomly selected $40,000$ patches from each class to create the training set. We sample $500,000$ IP pairs of patches randomly where both image patches of a pair come from the same class (i.e. both patches of a pair come either from unaltered class or from the same manipulation class). Similarly, we sample $500,000$ DP pairs randomly, where the two patches of a pair come from two different classes (e.g. one patch may come from Gaussian blurring operation and the other may come from Median filtering operation). To monitor the classification performance of the network during training, we apply it on a validation set which contains $10,000$ IP pairs and $10,000$ DP pairs that are not in the training set. Once, the model is trained we check the performance of the method on a test set which contains $50,000$ IP pairs and $50,000$ DP pairs.

The network was trained for $70,000$ iterations and stopped when it started converging. Figure 3a shows the training and validation losses with respect to iterations, and Figure 3b shows the training and validation accuracies with respect to iterations. It can be seen that after $60,000$ iterations the training loss and training accuracy start saturating indicating the convergence the network. The validation accuracy also reaches more than 99% after $60,000$ iterations and saturates. We stop the training process at $70,000$ iterations and save the final parameters of the model for the future use. To test the performance of the model, the trained model was tested on the test set which consists of $50,000$ IP pairs and $50,000$ DP pairs. It is to be noted that, the test image patches also come from the same four classes only. On this test set, the model

TABLE I: Different manipulations considered in this paper

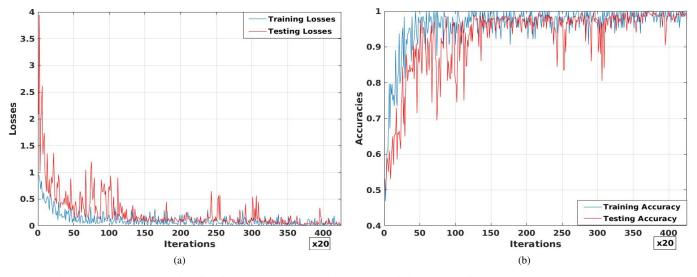| Editing Operation	Detail
Gaussian blurring	Kernel size = 5×5 and standard deviation (σ) = 1.1
Median filtering	Kernel size = 5×5
Resampling	Scaling factor = 1.5 and bilinear interpolation
Noise addition	AWGN with standard deviation (σ) = 2
Gamma correction	Parameter (γ) = 1.5

(a)

(b)

Fig. 3: (a) Training and validation loss versus iteration. (b) Training and validation accuracy versus iteration.

achieves an accuracy of 99.38%. This experiment shows that the proposed siamese network can discriminate the different types of image editing operations with a very high accuracy.

Another experiment is carried out to compare our method with Bayar and Stamm's method. In this experiment, we check the ability of the proposed method in classifying each of the four manipulation types individually. For this, we have created four different test sets as follows: for the Gaussian blurring test set, $50,000$ IP pairs are created by taking both images of the pairs from the Gaussian blurring only, and $50,000$ DP pairs are created by taking one image from the Gaussian manipulation and the other from any of the rest three classes. The test sets for the original, the median filtering, and the resampling cases are also created following the similar procedure. We have checked the classification accuracies of the already trained siamese network on these four test sets. For comparison, we have implemented Bayar and Stamm's method and tested its performance on these test images. It should be noted that the size of image patches used in this experiment is 150×150 as opposed to 227×227 used in the paper [13]. The classification results are shown in Table II. The proposed method classifies the original, the Gaussian blurred, the median filtered and the resampled patches with accuracies of 99.35%, 99.51%, 99.64% and 99.26% respectively, whereas Bayar and Stamm's method classified with accuracies of 98.70%, 99.80%, 98.85% and 99.13% respectively. These results show that except the Gaussian manipulation, the proposed siamese network outperforms the CNN method [13] for all other manipulations.

The next experiment is carried out to see the ability of the proposed method in discriminating manipulations not present

TABLE II: Classification accuracies on different manipulation classes

Manipulation	Proposed Method	Bayar and Stamm [13]
Original	**99.35**	98.70
Gaussian blurring	99.51	**99.80**
Median filtering	**99.64**	98.75
Resampling	**99.26**	99.13

TABLE III: Classification accuracies on manipulations not present in the training stage

Manipulation	Accuracy (%)
AWGN ($\sigma = 2$)	96.61
Gamma correction ($\gamma = 1.5$)	95.24

at the training phase. We have trained the network using the image patch pairs coming from the four classes as already mentioned. We test it on a set which contains images coming from a different type of manipulation. For this, we considered two different manipulation classes obtained by corrupting the images with AWGN and applying gamma correction operations on the image patches. For the AWGN case, we created $50,000$ IP pairs by taking both images of a pair from AWGN class only, and $50,000$ DP pairs are created by taking one image of the pair from AWGN and the other comes from one of the four classes, i.e. the original, the Gaussian blurring, the median filtering, and the resampling. On this set, the network achieved a classification accuracy of 96.61%. Similarly, to see the generalization ability of the network on gamma correction class, we created $50,000$ IP and $50,000$ DP test pairs in the

same manner, and tested the network on it. In this case, the network achieved an accuracy of 95.24%. From these results, it is evident that the network can discriminate images coming from different types of manipulations, even if the images come from manipulation classes not used in the training stage. Table III summarises the generalisation accuracy of the method. It should be noted that the existing universal forensics methods [10], [11], [13] cannot be applied in this case. This is because they can only classify images to one of the manipulations present in the training stage. This is a huge advantage of the proposed method over the state-of-the-art.

V. Conclusions and Future Work

In this paper, we proposed a novel image forensics method which can discriminate different types of image manipulations carried out on images. The proposed method employs a deep siamese CNN which takes a pair of image patches as input and decides whether they are identically or differently processed. That is, instead of classifying image patches to some fixed classes (types) of manipulations, the proposed method checks whether two image patches are processed through the same operation or not. Because of this, the proposed method can even discriminate image manipulations which were not present in the training stage. The experimental results show that the proposed method can differentiate between different types of manipulations with good accuracies.

The future work will involve further exploration of the universal nature of the proposed method. Also, checking the effectiveness of the proposed method in detecting real splicing forgeries is another task included in the future work.

References

[1] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of re-sampling," *IEEE Transactions on Information Forensics and Security*, vol. 53, no. 2, pp. 758–767, 2005.

[2] X. Feng, I. J. Cox, and G. Doerr, "Normalized energy density-based forensics detection of resampled images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 536–545, 2012.

[3] B. Mahdian and S. Saic, "Blind authentication using periodic properties of interpolation," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 529–536, 2008.

[4] X. Kang, M. C. Stamm, A. Peng, and K. J. R. Liu, "Robust median filtering forensics using an autoregressive model," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 9, pp. 1456–1468, 2013.

[5] M. C. Stamm and K. J. R. Liu, "Forensic detection of image manipulation using statistical intrinsic fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 3, pp. 492–506, 2010.

[6] C. Chen and J. Ni, "Median filtering detection using edge based prediction matrix," *Digital Forensics and Watermarking*, pp. 361–375, 2012.

[7] M. C. Stamm and K. J. R. Liu, "Blind forensics of contrast enhancement in digital images," in *IEEE International Conference on Image Processing*, 2008, pp. 3112–3115.

[8] ——, "Forensic estimation and reconstruction of contrast enhancement mapping," in *IEEE International Conference on Acoustic Speech and Signal Processing*, 2010, pp. 1698–1701.

[9] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of jpeg artifacts." *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, 2012.

[10] X. Qiu, H. Li, W. Luo, and J. Huang, "A universal image forensics strategy based on steganalytic model." in *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, no. 165-170, 2014.

[11] W. Fan, K. Wang, and F. Cayre, "General-purpose image forensics using patch likelihood under image statistical models," in *IEEE International Workshop on Information Forensics and Security (WIFS)*, no. 1-6, 2015.

[12] J. Chen, X. Kang, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, 1849.

[13] 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*, 2016, pp. 5–10.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst.*, ser. 1097-1105, 2012.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[17] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Processing Letters*, no. 1849-1853, 2015.

[18] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.

[19] T. Penvy, P. Bas, and J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 215–224, 2010.

[20] M. Fontani and M. Barni, "Hiding traces of median filtering in digital images," in *Proc. 20th Eur. Signal Process. Conf.*, 2012, pp. 1239–1243.

[21] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning workshop*, 2015.

[22] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, arXiv preprint arXiv:1207.0580.

[23] Y. LeCun and J. Huang, "Loss functions for discriminative training of energy-based models," in *Proc. AISTATS*, 2005.

[24] T. Gloe and R. Bohme, "The 'dresden image database' for benchmarking digital image forensics." in *Proceedings of the 25th Symposium on Applied Computing*, 2010, pp. 1585–1591.

[25] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[26] T. Dozat, "Incorporating nesterov momentum into adam," in *ICLR Workshop*, 2016.

[27] M. Welling and Y. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceeding of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 681–688.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift." in *ICML*, 2015.