

# All Works of Ziyue Xiang

Ziyue “Alan” Xiang

October 29, 2019

## Contents

<b>1</b>	<b>Scientific Image Tampering Detection Based On Noise Inconsistencies: A Method And Datasets (STATUS: submitted, under review)</b>	<b>2</b>
<b>2</b>	<b>Estimating Probability of Image Features to Support Figure Element Reuse Investigations (STATUS: ongoing, close to finish)</b>	<b>12</b>
<b>3</b>	<b>Better Performance Evaluation For Image Tampering Localization (STATUS: ongoing)</b>	<b>18</b>
<b>4</b>	<b>Analyzing Robust/Non-robust Features From MNIST (STATUS: ongoing)</b>	<b>24</b>

# Scientific Image Tampering Detection Based On Noise Inconsistencies: A Method And Datasets

Ziyue Xiang

*College of Engineering & Computer Science, Syracuse University*

Daniel E. Acuna\*

*School of Information Studies, Syracuse University*

---

## Abstract

Scientific image tampering is a problem that affects not only authors but also the general perception of the research community. Although previous researchers have developed methods to identify tampering in natural images, these methods may not thrive under the scientific setting as scientific images have different statistics, format, quality, and intentions. Therefore, we propose a scientific-image specific tampering detection method based on noise inconsistencies, which is capable of learning and generalizing to different fields of science. We train and test our method on a new dataset of manipulated western blot and microscopy imagery, which aims at emulating problematic images in science. The test results show that our method can detect various types of image manipulation in different scenarios robustly, and it outperforms existing general-purpose image tampering detection schemes. We discuss applications beyond these two types of images and suggest next steps for making detection of problematic images a systematic step in peer review and science in general.

*Keywords:* Scientific images, Digital image forensics, Noise inconsistency, Scientific image manipulation dataset

---

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Previous Work</b>	<b>2</b>
<b>3</b>	<b>Our Proposed Method</b>	<b>3</b>
3.1	Residual Image Generators	3
3.2	Feature Extraction	4
3.2.1	Patch Reinterpretation	4
3.2.2	Feature Design	5
<b>4</b>	<b>Experiments</b>	<b>5</b>
4.1	Datasets	5
4.2	Test Configurations	6
4.3	Test Results	7
<b>5</b>	<b>Discussion And Conclusion</b>	<b>7</b>

## 1. Introduction

The use of digital images has become increasingly ubiquitous in all types of publications. What comes with the growing importance of digital images is the development

of image tampering techniques. In the past, modifying or concealing the content of an image would require dedicated personnel and tools. Today, however, image tampering is much easier with state-of-the-art image processing software. This trend has affected many aspects of our society, as we see prominent forgery cases occur in journalism and academia [1]. Consequently, many detection techniques have been developed for these scenarios (see [2]). Only

---

\*Corresponding author

*Email addresses:* xziyue@syr.edu (Ziyue Xiang),  
deacuna@syr.edu (Daniel E. Acuna)

recently, however, attention has been paid to image manipulation in scientific publications [3]. Although it is possible to use existing methods on scientific images directly, we hypothesize that significant adaptations must be made due to the fact that they usually possess distinctive statistical patterns, formats and resolutions. In this work, we aim at developing a scientific-specific image manipulation detection technique, which we test on a novel scientific image manipulation dataset of western blots and microscopy imagery—there are no datasets openly available about scientific image manipulation yet (but see [4]). Thus, as most scientific images increasingly come in digital form, the detection of possible manipulations should also get at the same level of quality as other fields that use digital images only.

It is undeniable that an increasing amount of tampered images are finding their ways into scientific publications. Bik, Casadevall and Fang [5] examined 20,621 biomedical research papers from 1995 to 2014, where they find that at least 1.9 percent are subject to deliberate image manipulation. The fact that these suspicious papers went through the careful reviewing process suggests how difficult it is to examine image tampering in scientific research manually. Because the large quantity of digital images present in submitted manuscript, it is crucial for publishers to be able to identify image manipulation in an automated fashion.

The scientific research context sets a different tolerance for image manipulation. Many operations, including resizing, contrast adjusting, sharpening, and white balancing are generally acceptable as part of the figure preparation process. However, some other types of tampering, especially the ones that alter the image content semantically, are strictly prohibited. These manipulations include copy-move (without proper attribution), splicing, removal, and retouching<sup>1</sup>. Acuna, Brookes and Kording [6] developed a method to detect figure element reuse across a paper database. Intra-image copy-move can be detected rather robustly with SIFT features and pattern matching [7]. However, detection of image manipulation that does not involve reuse is significantly more challenging. A comprehensive scientific image manipulation detection pipeline should include manipulation detection.

As scientific papers are reviewed by experts, we reckon that articles containing manipulations that incur in contextual inconsistencies (e.g., brain activation patterns from fMRI in the middle of a microscopy image) will be easily picked out. What humans *cannot* see properly is the noise pattern within an image—and scientists seeking to falsify images exploit this weakness. Therefore, we propose a novel image tampering detection method for scientific images, which is based on uncovering noise inconsistencies. Specifically, our proposed method contains the following features:

1. It is based on supervised learning, which is capable of learning from existing databases and new instances.
2. It works for images of different resolutions and from different devices.
3. It is not restricted to any specific image format.
4. It is capable of generating good predictions with a small training set.
5. It is flexible and can be fine-tuned for different fields of science.

In section 2, we briefly summarize previous work on digital image forensics. In section 3, we discuss the design of our proposed method. In section 4, we introduce our scientific image manipulation datasets and present the test results of our method on them. In section 5, we conclude by discussing limitations and future extension of our method.

## 2. Previous Work

There have been a large amount of previous research on image tampering detection, but very few of them focus on scientific images. The first class of tampering detection methods aims at detecting a specific type of manipulation, the most common being resizing and resampling [8, 9, 10, 11, 12], median filtering [13, 14, 15, 16], contrast enhancement [17, 18, 19, 20], blurring [21], and multiple JPEG compression [22, 23, 24, 25]. Many of these manipulations are valid in the scientific research context, and it can be non-trivial to merge results from single detectors in order to build a comprehensive one.

The second class of tampering detection methods aims at general-purpose image tampering detection. Dirik and Memon [26] try to catch the inconsistency of Color Filtering Array (CFA) patterns within images taken by digital cameras—a signal generated by digital cameras. However, scientific images are not necessarily taken by digital cameras. Wang, Dong and Tan [27] leverage the characteristics of the DCT coefficients in JPEG images to achieve tampering localization, but the method is confined to a specific format. Mahdian and Saic [28] propose a method that predicts tampered regions based on wavelet transform and noise level estimation. All these methods are unable to learn from data, which limits their abilities to generalize to different fields. Another group of methods combines steganalysis tools [29, 30] with Gaussian Mixture Models (GMM) to identify potentially manipulated regions [31, 32]. These unsupervised-learning-based methods are also unable to learn from existing database effectively and therefore tend to underperform in practice.

Because of the occurrence of large image datasets, neural-network-based tampering detection methods are likely to yield good performance [33], especially those based on Convolutional Neural Networks (CNN) [34, 35, 36]. They usually target high resolution natural images. It is unclear, however, whether they can be transitioned for the scientific scenario. For example, it is challenging to train such

<sup>1</sup><https://ori.hhs.gov/education/products/RIandImages/guidelines/list.html>

a network for scientific images exclusively as they usually require tens of thousands of images as training data, which to the best of our knowledge is not yet available.

### 3. Our Proposed Method

Our method is based on a combination of several heterogeneous feature extractors that are later combined to produce single predictions for patches (Figure 1). At first, an input image will go through a variable amount of residual image generators. The type and amount of these generators can be chosen based on the application. Each type of residual image will have its own feature extractor, which is based on our proposed feature extraction scheme with (possibly) different configurations. The features are then fed into a classifier after post-processing.

The proposed method works on residual images, which are essentially image after filtering or the difference between an image and its interpolated version. It is a way to discard content and emphasize noise pattern within an image, which is widely used in image manipulation detection practice. However, in many previous works, only one type of residuals is used [26, 32, 36]. Because each residual may have different sensitivity levels to different types of manipulation, using only one not only limits the method’s ability to detect a wide variety of manipulation, but also renders the method more vulnerable against adversaries. Therefore, we decide to combine a number of residuals in our method to increase the robustness.

Because our feature extraction method drastically reduces the dimensionality of image data, which relieves the need of a huge amount of training data, it is possible to use a light-weight classifier as the back end, such as logistic regression or support vector machine (SVM). As there are many ways to generate residual images, and that the feature extraction method comes with a number of parameters to decide, our image manipulation detection method possesses high degree of flexibility. Unlike the parameters in neural networks, for example, which are rather obscure for human beings, the underlying meanings of the parameters in our feature extraction method are straightforward. Therefore, it is easier for one to manually adapt our method for different fields.

#### 3.1. Residual Image Generators

There are numerous ways of generating residual images, we list the following ones because they are functional for a wide range of applications. Note that the capability of our method is significantly influenced by the choice of residuals. However, it is possible to design new residual image generators for specific scenarios.

##### 1. Steganalytic Filters

Steganalysis (techniques used for detecting hidden messages in communications) has been used in image tampering detection practice extensively. This type of analysis aims to expose hidden information

planted in images by steganography techniques. Although it is not directly linked to image tampering detection, it is suggested that the tasks of image forensics and steganalysis are very much alike when the action of data embedding in steganography is treated as image manipulating [37]. Similar to the rich model strategy proposed in [29], we can apply many different filters and see which one can spot inconsistencies. In our work, we use several filters that provide a relatively comprehensive view of potential inconsistencies (Figure 2).

The filters selected are high-pass because we want to throw away information about the image content and emphasize noise patterns as much as possible. The residual image in this case is the image after convolution. An example of steganalytic filtering residual is shown in Figure 3.

##### 2. Error Level Analysis (ELA)

ELA is an analysis technique that targets JPEG compression. The idea behind it is that the amount of error introduced by JPEG compression is nonlinear: a 90-quality JPEG image resaved at quality 90 is equivalent to a one-time save of quality 81; a 90-quality JPEG image resaved at quality 75 is equivalent to a one-time save of quality 67.5 [38]; and so on. If some part of a JPEG-compressed image is altered with a different JPEG quality factor, when it is compressed again, the loss of information of that part will differ from other regions. To uncover the inconsistency, ELA residual is computed by intentionally resaving the image in JPEG format with a particular quality (e.g. 90) and then computing the difference of the two images. An example of ELA residual is shown in Figure 4.

##### 3. Median Filtering Residual

Median filtering can suppress the noise of an image. When applying median filtering to a tampered image, the tampered part may possess a different noise pattern and therefore respond differently. The median filtering residual is the difference between the original image and median filtered image. An example is shown in Figure 5.

##### 4. Wavelet Denoising Residual

Wavelet denoising is a type of denoising method that represents an image in wavelet domain and cancels the noise based on that representation. Similar to the median filtering residual’s case, the tampered region may react differently compared to the rest of the image and therefore give away its own identity. It is also suggested by Dirik and Memon [26] that using wavelet denoising can uncover the sensor noise inconsistency of digital cameras. The wavelet denoising residual is given by the difference between the original image and the denoised image. An example is shown in Figure 6.

It is worth noticing that the tampered images in the

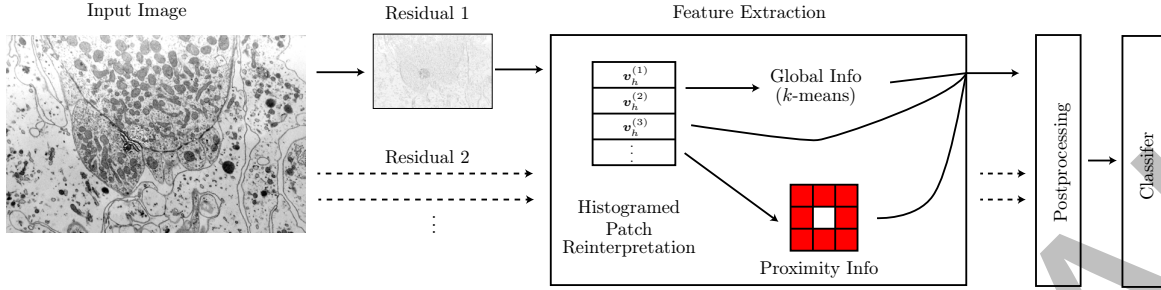


Figure 1: Overall design of our proposed method. The input image goes through several residual generators and feature extractors in parallel. All extracted features will be merged in a postprocessing step and then fed to a classifier.

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

(a) (b)

Figure 2: High-pass filters selected in our experiment

demonstrations are selected so that the manipulation pattern is visible in the specific residual. However, in practice, this may not always be the case. Usually it is necessary to examine multiple residual images before drawing a conclusion.

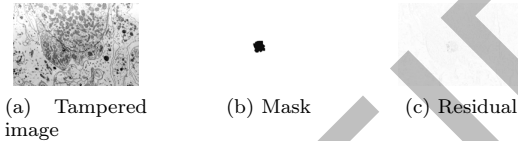


Figure 3: Demonstration of steganalytic residual

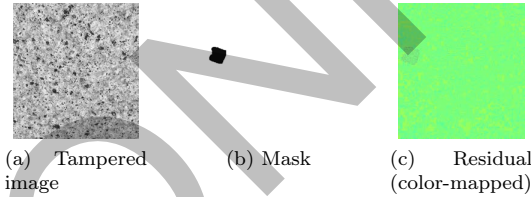


Figure 4: Demonstration of ELA residual

### 3.2. Feature Extraction

Our method is patch-based, which means it will generate a prediction for each patch in the image. Using patches instead of single pixels to represent an image not only shrinks the scale of computation, but also enriches the amount of statistical information within each smallest unit. At the limit, the patch size can be chosen so that pixel-based and patch-based become almost the same. After deciding on the patch size, the feature extraction step

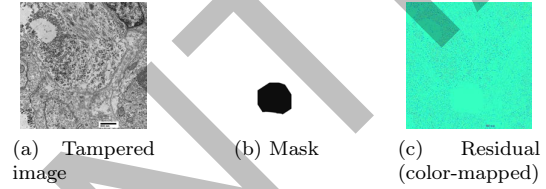


Figure 5: Demonstration of median filtering residual

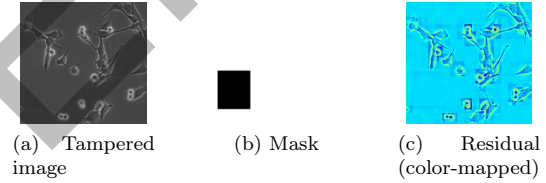


Figure 6: Demonstration of wavelet denoising residual

will generate a corresponding feature vector for each patch in the image. In this section, we discuss how these feature vectors are computed.

#### 3.2.1. Patch Reinterpretation

Residuals reduce the complexity of image data, but they still have the same dimensionality as the original image. To further compress data for classification, we propose a new feature extraction method for image tampering detection. Intuitively, an image region is considered to be *tampered* not because it is unique itself, but mainly due to the fact that it is different from *the rest of the image*. Therefore, an ideal feature design should contain sufficient amount of global information. We add global information by reinterpreting an image region using the rest of the image.

First, an input image of size  $(h, w)$  will be divided into patches of size  $(m, n)$ . If the shapes are not divisible, the image will be cropped to the nearest multipliers of each dimension. Therefore, an image of size  $(h, w)$  will be divided into a patch matrix of size  $(\lfloor h/m \rfloor, \lfloor w/n \rfloor)$ .

Then, the patch matrix will be split into a rectangular patch grid of size  $(s, t)$ , where each cell contains a certain

Symbol	Description
$(h, w)$	size of the image
$(m, n)$	dimension of each patch
$(s, t)$	dimension of the patch grid
$l_{ij}$	the likelihood function of the grid cell on $i$ th row and $j$ th column

Table 1: List of symbols used in feature extraction

number of patches. The number of patches in most cells is

$$\left\lfloor \frac{h/m}{s} \right\rfloor \times \left\lfloor \frac{w/n}{t} \right\rfloor,$$

except for those cells on the edges, which may have fewer patches.

For each cell in the grid, we fit an outlier detector that is capable of telling the likelihood of a new sample being an outlier. Given a patch  $\mathbf{p}$ , it can be reinterpreted by a vector  $\mathbf{v}$ , which is given by

$$\mathbf{v} = (l_{11}(\mathbf{p}), l_{12}(\mathbf{p}), l_{13}(\mathbf{p}), \dots, l_{1t}(\mathbf{p}), \\ l_{21}(\mathbf{p}), l_{22}(\mathbf{p}), l_{23}(\mathbf{p}), \dots, l_{2t}(\mathbf{p}), \\ l_{31}(\mathbf{p}), l_{32}(\mathbf{p}), l_{33}(\mathbf{p}), \dots, l_{3t}(\mathbf{p}), \\ \dots \\ l_{s1}(\mathbf{p}), l_{s2}(\mathbf{p}), l_{s3}(\mathbf{p}), \dots, l_{st}(\mathbf{p})).$$

An illustration of this reinterpretation method is shown in Figure 7, where black blocks represent patches, red blocks represent grid cells and the yellow region represents the tampered region. In this case,  $(s, t) = (3, 4)$ . Because the tampered region has a different residual pattern, and its contaminated patches concentrate in one of the cells, the outlier detector of that cell will learn a distinct decision boundary compared to other ones. As a result, an authentic patch  $\mathbf{p}_a$  will have lower outlier likelihood in all components except for  $l_{23}(\mathbf{p}_a)$ ; a tampered patch  $\mathbf{p}_t$  will have higher outlier likelihood in all components except for  $l_{23}(\mathbf{p}_t)$ . This difference in structure allows us to distinguish between authentic and tampered patches. In practice, we use the histogram of  $\mathbf{v}$  (denoted by  $\mathbf{v}_h$ ), which not only encodes the structure in summary-statistics space, but also becomes position invariant.

### 3.2.2. Feature Design

Besides  $\mathbf{v}_h$ , we include some other information in order to concentrate more global information within the feature. The final feature of a patch contains the following components:

1.  $\mathbf{v}_h$ : the histogrammed patch reinterpretation. After generating all histogrammed reinterpretations of an image, we normalize them to  $[0, 1]$ .
2. Proximity information: how much the patch differs from its neighborhood. We choose the Euclidean distance between the histogrammed reinterpretation of the patch and those of its surrounding neighbors'.

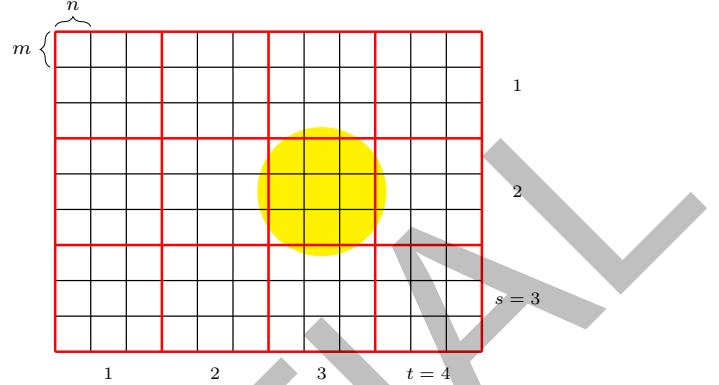


Figure 7: Patch reinterpretation illustration. The parameters are described in Table 1.

3. Global information: how much the patch differs from the entire image. After computing the histogrammed reinterpretations for all patches within an image, we apply  $k$ -means clustering on them, which generates a set of weights and cluster centroids. The additional global information of a patch is given by the Euclidean distance between the reinterpretation and the cluster centroids, as well as the corresponding weights of the centroids.

## 4. Experiments

Due to the lack of science-specific image manipulation detection databases, we synthesize our own database for the experiments.

### 4.1. Datasets

Our novel scientific image manipulation datasets mainly consist of the following three types of manipulations:

1. Removal: covering an image region with a single color or with noise. We manually select a rectangular region to be removed from the image. Then we select another rectangular region to sample the color or noise to fill the removal region, where we can compute the mean  $\mu$  and standard deviation  $\sigma$  of the pixels. We generate four images for each pair of selection according to the configuration given in Table 2.

Removal Region Mean	Removal Region Standard Deviation
$\mu$	0
$\mu$	$0.5\sigma$
$\mu$	$\sigma$
$\mu$	$2\sigma$

Table 2: Image generation configuration of removals. There mean of the removal region is equal to that of the sample region's, but we vary the standard deviation from zero (pure color) to two standard deviations to create different visual effects.

2. Splicing: copying content from another image. We randomly choose a small region from the foreground image and paste it at an arbitrary location on the background image. To create noise inconsistency, the region will either be recompressed with JPEG or processed with sharpening filters.
3. Retouching: modifying the content of the image. We will randomly choose a small region within an image and apply Gaussian blurring to it.

These manipulations are selected because we believe that they are more prevalent in problematic scientific papers.

We build two datasets that contain western blot images and microscopy images, respectively. We choose images around these two topics because of their frequency in the literature huge, and they are more susceptible to manipulation. We also create a natural image dataset to compensate for the lack of microscopy images for training. It is only used in the training phase. The details of datasets are shown in Table 4. The meanings of tampering type abbreviations are shown in Table 3.

Abbreviation	Meaning
R	removal images
J	splicing images recompressed by JPEG
F	splicing images processed with sharpening filters
B	Gaussian blurred images
G	genuine images

Table 3: Tampering type abbreviations

Collection	Image Source	Contents <sup>2</sup>	Average Resolution
western blot	western blot images from the Internet	R(436), G(51)	137, 244
microscopy	microscopy images from the Internet	R(180), J(20), F(19), B(20), G(21)	591, 906
natural image	natural images from the "pristine" collection of IEEE dataset [39]	J(40), F(40), B(40), G(40)	775, 328

Table 4: The specification of the proposed scientific image forensics datasets.

#### 4.2. Test Configurations

The sizes of images in the western blot collection are significantly smaller. Therefore, we need to train a special model for them. For the microscopy model, we added natural images into the training set to compensate for the lack of data. The patches from residual images are transformed into frequency domain by Discrete Cosine Transform (DCT) because it yields slightly better performance.

<sup>2</sup>format: type(number of images)

Within each model, the parameters of each feature extractor are the same. Detailed configurations of the two models that we trained are shown in Table 5.

We use a one-class SVM outlier detector [40], provided by scikit-learn [41], which is based on LIBSVM[42]. The kernel we use is radial basis function, whose kernel coefficient ( $\gamma$ ) is given by the scale, which is

$$\frac{1}{\text{number of features} \times \text{variance of all inputs}}$$

The tolerance of optimization is set to 0.01; and  $\nu$  (the upper bound on the fraction of training errors and the lower bound on the fraction of support vectors) is set to 0.1.

Note that the choice of parameters can significantly influence the speed of feature extraction. One of the most expensive operations is fitting SVM, which has a computational complexity of  $O(N^3)$ , where  $N$  is the number of patches in each grid cell. Therefore, it is important to choose an appropriate  $(m, n)$  and  $(s, t)$  pair. With our Python implementation and the configuration given in Table 5, the extraction speed for western blots is approximately 212.36 sec/megapixel (12.13 sec/image), while the extraction speed for microscopy images is approximately 86.15 sec/megapixel (49.32 sec/image). We tried to use ThunderSVM[43], which is a GPU-accelerated SVM implementation. Although it has a much higher speed, its precision is not ideal compared to LIBSVM. Therefore, our experiments are conducted with LIBSVM only.

The number of centroids of  $k$ -means clustering is set to  $k = 6$ , and the clustering algorithm is run 150 times with different initializations in order to get a best result. We select this particular value of  $k$  because when we apply  $k$ -means clustering to  $\mathbf{v}_h$ , the tampered region would usually blend with other clusters unless there are more than 6 centroids. Therefore, we consider it reasonable to represent the major content of an image by its first 6 cluster centroids.

	Patch Dimension	Patch Grid Dimension	# Training Images	# Testing Images
Western Blot	(6, 6)	(5, 5)	352	135
Microscopy	(10, 10)	(7, 7)	251	106

Table 5: Test configuration parameters

Because the dimensionality of the extracted feature is not very high, the outputs of each feature extractor are simply concatenated into a single feature vector and then fed to the classifier. The classifier we use is a simple Multilayer Perceptron neural network. For the western blot model, we use a four-layer network with 200 units per layer; for the microscopy model, we use a similar network with 300 units per layer. Softmax regression is applied to the last layer to get the classification results.

### 4.3. Test Results

The performance evaluation metric that we use are patch-level accuracy, AUC scores, and F1 scores. We compare the performance of our model with two baseline models, which are widely compared against in related papers:

1. CFA [26]: a method that uses nearby pixels to evaluate the Camera Filter Array patterns and then produces the tampering probability based on the prediction error.
2. NOI [28]: a method that finds noise inconsistencies by using high pass wavelet coefficients to model local noise.

For our method, the threshold for F1 score is 0.5. For the baseline methods, their output map is normalized to  $[0, 1]$ , and the F1 score is acquired by setting the threshold to 0.5.

Table 8 shows the accuracies of the three methods on genuine images, where AUC and F1 scores does not apply. Table 6 and 7 shows the AUC scores and F1 scores of our methods compared to the baseline. The meanings of the abbreviations can be seen in Table 3. The “overall” scores are computed across the entire dataset, including genuine images. A visual comparison of the results of each method is shown in Figure 8.

It can be seen that CFA cannot handle western blot images very well, as it has low accuracy on genuine images. Its performance on J, F and B tampering types are also mediocre. NOI has better behavior at locating noisy regions in the image, but it fails drastically when encountering manipulations that contain less noise. It constantly treats R[0] and B manipulations as negatives, which yields a false negative region that is not always separable. Its performance on J images is not very satisfactory as well. Generally speaking, the performance of our method is more consistent across different types of manipulations, which makes it more reliable in practice.

## 5. Discussion And Conclusion

We have proposed a novel image tampering detection method for scientific images, which is based on uncovering noise inconsistencies. We use residual images to exploit the noise pattern of the image, and we develop a new feature extraction technique to lower the dimensionality of the problem so that it can be handled by a light-weight classifier. The method is tested on a new scientific image dataset of western blots and microscopy imagery. Compare to two base line methods popular in the literature, results suggest that our method is capable of detecting various types of image manipulations better and more consistently. Thus, our solution promises to solve an important part of image tampering in science effectively.

There are also some weaknesses in our study. First, our proposed method is tested on a custom database, which only contains a small amount of samples. We only include several types of manipulations in our datasets, which is rather monotonous compared to the space of all possible image tampering techniques. Nonetheless, the choice of these specific image sources and manipulation types is inspired by existing problematic papers. If our method is capable of detecting these manipulations to some extent, we believe that it can make valuable discoveries once put into practice.

Second, we think that noise-inconsistency-based methods do possess certain limitations. For example, not all manipulation will necessarily trigger noise inconsistency; it is also easier for one to hide the noise inconsistency, had he/she known the underlying mechanism of the automatic detector. This kind of adversarial attack, however, is significantly challenging and unlikely to be done by the average scientist. In the future, we want to develop more advanced methods that take both image content and noise pattern into account.

However, our proposed method is one of the first methods that tackles scientific image manipulation directly. Put together in screening pipelines for scientific publications (similar to [6]), our method would significantly expand the range of manipulations that could be captured at scale. It also makes predictions based on many types of residuals, which possesses improved robustness. The method a set of easily adjustable parameters, which allows it to be adapted for different fields with less effort and a smaller amount of training data.

We would like to continue extending the database with more images from various disciplines to make it standard and comprehensive, and report test results on the updated version. It is our hope that the datasets that we propose can also be useful for the nascent Computational Research Integrity research area. But we are also facing a major difficulty: there are no openly available datasets on images that actually come from *science* (although see the efforts in [4]). The images that we currently have are collected from the Internet, and form a small but significant portion of images with manipulation issues. Unfortunately, access to problematic scientific images are tend to be removed from the public soon after retraction. So far, neither publishers nor authors are yet willing to share those images for understandable reasons. Hopefully, once scientific image tampering detection methods prove their efficacy, publishers and funders can start to share and create datasets with proper safeguards to check for potential problems during peer review – similar to how they do it with full-text through the Crossref organization<sup>4</sup>.

<sup>4</sup><https://crossref.org>

<sup>3</sup>format: R[noise standard deviation]



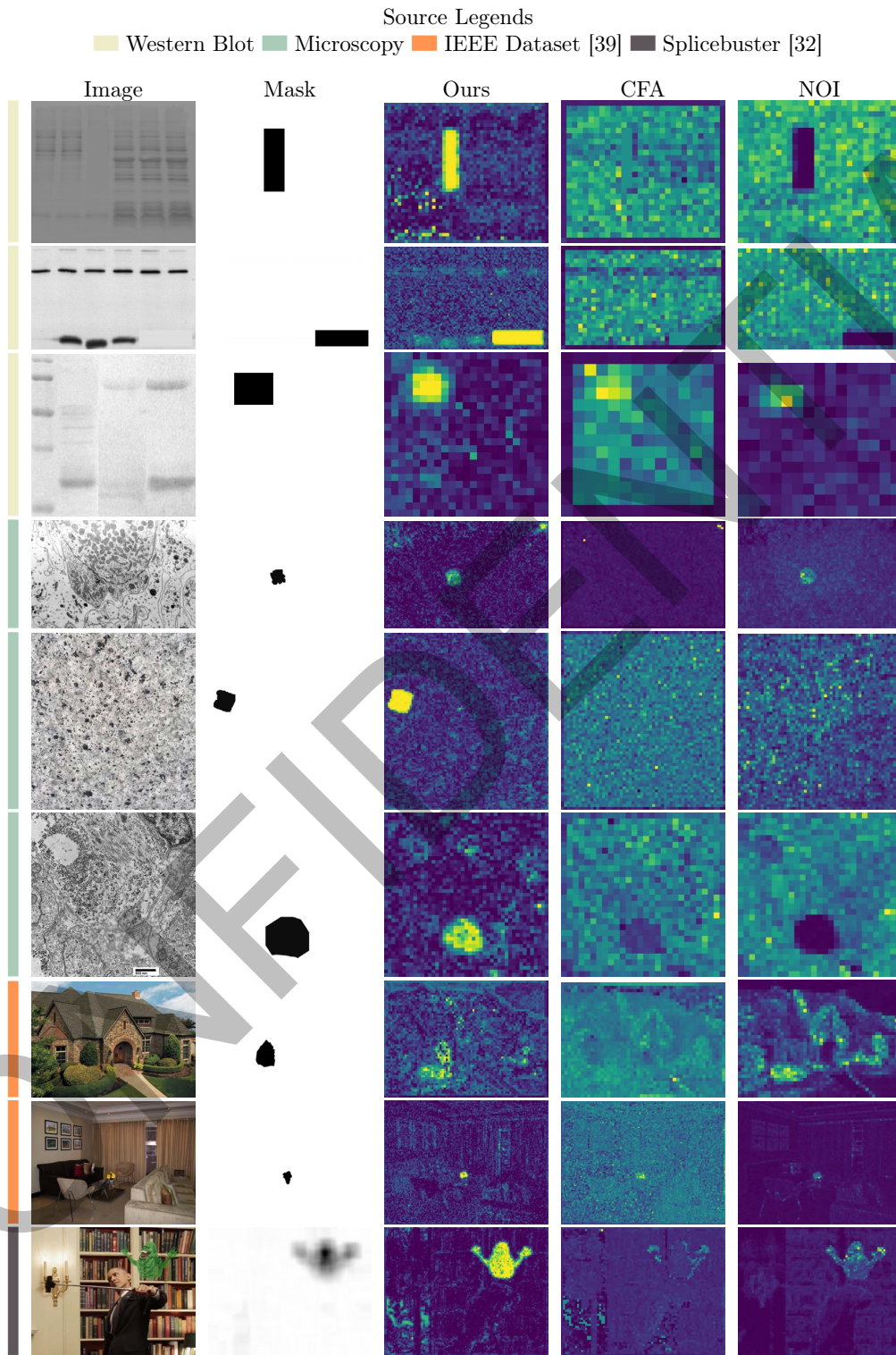


Figure 8: Visual comparison of the results

	Tampering Type	Ours	CFA	NOI		Tampering Type	Ours	CFA	NOI
	Western Blot	R[0] <sup>3</sup>	<b>0.939</b>	0.606		0.026	Microscopy	R[0]	<b>0.924</b>
R[0.5 $\sigma$ ]		0.861	0.866	<b>0.879</b>	R[0.5 $\sigma$ ]	<b>0.903</b>		0.925	0.887
R[ $\sigma$ ]		0.923	0.877	<b>0.968</b>	R[ $\sigma$ ]	<b>0.968</b>		0.940	0.959
R[2 $\sigma$ ]		0.990	0.885	<b>0.992</b>	R[2 $\sigma$ ]	0.966		0.937	<b>0.978</b>
					J	<b>0.994</b>		0.639	0.618
				F	0.868	0.629	<b>0.913</b>		
				B	<b>0.805</b>	0.334	0.104		
overall		<b>0.927</b>	0.813	0.696	overall	<b>0.925</b>	0.864	0.695	

Table 6: The AUC scores on datasets

	Tampering Type	Ours	CFA	NOI		Tampering Type	Ours	CFA	NOI
	Western Blot	R[0]	<b>0.834</b>	0.039		0.003	Microscopy	R[0]	<b>0.834</b>
R[0.5 $\sigma$ ]		<b>0.744</b>	0.399	0.543	R[0.5 $\sigma$ ]	<b>0.745</b>		0.398	0.560
R[ $\sigma$ ]		<b>0.867</b>	0.553	0.712	R[ $\sigma$ ]	<b>0.867</b>		0.414	0.773
R[2 $\sigma$ ]		0.762	0.522	<b>0.880</b>	R[2 $\sigma$ ]	0.762		0.378	<b>0.896</b>
					J	<b>0.966</b>		0.038	0.045
				F	<b>0.623</b>	0.139	0.360		
				R	<b>0.476</b>	0.016	0.001		
overall		<b>0.770</b>	0.300	0.424	overall	<b>0.738</b>	0.329	0.455	

Table 7: The F1 scores on datasets

Western Blot			Microscopy		
Ours	CFA	NOI	Ours	CFA	NOI
<b>0.988</b>	0.513	0.838	<b>0.988</b>	0.774	0.920

Table 8: The accuracy scores on genuine images

## Acknowledgments

Daniel E. Acuna is funded by the Office of Research Integrity grants #ORIIR180041 and #ORIIR19001.

## References

## References

- [1] H. Farid, Photo forensics, MIT Press, 2016.
- [2] H. Farid, Image Forgery Detection A survey, IEEE SIGNAL PROCESSING MAGAZINE 26 (2) (2009) 16–25. doi:{10.1109/MSP.2008.931079}.
- [3] N. Gilbert, Science journals crack down on image manipulation (2009).
- [4] T. S. Beck, Shaping Images: Scholarly Perspectives on Image Manipulation, Walter de Gruyter GmbH & Co KG, 2016.
- [5] E. M. Bik, A. Casadevall, F. C. Fang, The prevalence of inappropriate image duplication in biomedical research publications, MBio 7 (3) (2016) e00809–16.
- [6] D. E. Acuna, P. S. Brookes, K. P. Kording, Bioscience-scale automated detection of figure element reuse, bioRxiv (2018) 269415.
- [7] H. Huang, W. Guo, Y. Zhang, Detection of copy-move forgery in digital images using sift algorithm, in: 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Vol. 2, IEEE, 2008, pp. 272–276.
- [8] A. C. Popescu, H. Farid, Exposing digital forgeries by detecting traces of resampling, IEEE Transactions on signal processing 53 (2) (2005) 758–767.
- [9] M. Kirchner, Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue, in: Proceedings of the 10th ACM workshop on Multimedia and security, ACM, 2008, pp. 11–20.

- [10] N. Dalggaard, C. Mosquera, F. Pérez-González, On the role of differentiation for resampling detection, in: 2010 IEEE International Conference on Image Processing, IEEE, 2010, pp. 1753–1756.
- [11] X. Feng, I. J. Cox, G. Doerr, Normalized energy density-based forensic detection of resampled images, IEEE Transactions on Multimedia 14 (3) (2012) 536–545.
- [12] B. Mahdian, S. Saic, Blind authentication using periodic properties of interpolation, IEEE Transactions on Information Forensics and Security 3 (3) (2008) 529–538.
- [13] M. Kirchner, J. Fridrich, On detection of median filtering in digital images, in: Media forensics and security II, Vol. 7541, International Society for Optics and Photonics, 2010, p. 754110.
- [14] X. Kang, M. C. Stamm, A. Peng, K. R. Liu, Robust median filtering forensics using an autoregressive model, IEEE Transactions on Information Forensics and Security 8 (9) (2013) 1456–1468.
- [15] G. Cao, Y. Zhao, R. Ni, L. Yu, H. Tian, Forensic detection of median filtering in digital images, in: 2010 IEEE International Conference on Multimedia and Expo, IEEE, 2010, pp. 89–94.
- [16] C. Chen, J. Ni, Median filtering detection using edge based prediction matrix, in: International Workshop on Digital Watermarking, Springer, 2011, pp. 361–375.
- [17] M. C. Stamm, K. R. Liu, Forensic detection of image manipulation using statistical intrinsic fingerprints, IEEE Transactions on Information Forensics and Security 5 (3) (2010) 492–506.
- [18] H. Yao, S. Wang, X. Zhang, Detect piecewise linear contrast enhancement and estimate parameters using spectral analysis of image histogram, in: IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009), 2009, pp. 94–97.
- [19] M. Stamm, K. R. Liu, Blind forensics of contrast enhancement in digital images, in: 2008 15th IEEE International Conference on Image Processing, IEEE, 2008, pp. 3112–3115.
- [20] M. C. Stamm, K. R. Liu, Forensic estimation and reconstruction of a contrast enhancement mapping, in: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2010, pp. 1698–1701.
- [21] R. Liu, Z. Li, J. Jia, Image partial blur detection and classification, in: 2008 IEEE conference on computer vision and pattern recognition, IEEE, 2008, pp. 1–8.
- [22] T. Bianchi, A. Piva, Detection of non-aligned double JPEG compression with estimation of primary compression parameters, in: 2011 18th IEEE International Conference on Image

- Processing, IEEE, 2011, pp. 1929–1932.
- [23] T. Bianchi, A. Piva, Image forgery localization via block-grained analysis of JPEG artifacts, *IEEE Transactions on Information Forensics and Security* 7 (3) (2012) 1003–1017.
- [24] R. Neelamani, R. De Queiroz, Z. Fan, S. Dash, R. G. Baraniuk, JPEG compression history estimation for color images, *IEEE Transactions on Image Processing* 15 (6) (2006) 1365–1378.
- [25] Z. Qu, W. Luo, J. Huang, A convolutive mixing model for shifted double JPEG compression with application to passive image authentication, in: 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2008, pp. 1661–1664.
- [26] A. E. Dirik, N. Memon, Image tamper detection based on demosaicing artifacts, in: 2009 16th IEEE International Conference on Image Processing (ICIP), IEEE, 2009, pp. 1497–1500.
- [27] W. Wang, J. Dong, T. Tan, Exploring DCT coefficient quantization effects for local tampering detection, *IEEE Transactions on Information Forensics and Security* 9 (10) (2014) 1653–1666.
- [28] B. Mahdian, S. Saic, Using noise inconsistencies for blind image forensics, *Image and Vision Computing* 27 (10) (2009) 1497–1503.
- [29] J. Fridrich, J. Kodovsky, Rich models for steganalysis of digital images, *IEEE Transactions on Information Forensics and Security* 7 (3) (2012) 868–882.
- [30] T. Pevny, P. Bas, J. Fridrich, Steganalysis by subtractive pixel adjacency matrix, *IEEE Transactions on Information Forensics and Security* 5 (2) (2010) 215–224.
- [31] W. Fan, K. Wang, F. Cayre, General-purpose image forensics using patch likelihood under image statistical models, in: 2015 IEEE International Workshop on Information Forensics and Security (WIFS), IEEE, 2015, pp. 1–6.
- [32] D. Cozzolino, G. Poggi, L. Verdoliva, Splicebuster: A new blind image splicing detector, in: 2015 IEEE International Workshop on Information Forensics and Security (WIFS), IEEE, 2015, pp. 1–6, (program and test images are downloaded from URL).
- [33] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, B. Manjunath, Exploiting spatial structure for localizing manipulated image regions, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4970–4979.
- [34] B. Bayar, 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, ACM, 2016, pp. 5–10.
- [35] B. Bayar, M. C. Stamm, Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection, *IEEE Transactions on Information Forensics and Security* 13 (11) (2018) 2691–2706.
- [36] P. Zhou, X. Han, V. I. Morariu, L. S. Davis, Learning rich features for image manipulation detection, arXiv preprint arXiv:1805.04953.
- [37] X. Qiu, H. Li, W. Luo, J. Huang, A universal image forensic strategy based on steganalytic model, in: Proceedings of the 2nd ACM workshop on Information hiding and multimedia security, ACM, 2014, pp. 165–170.
- [38] N. Krawetz, A picture’s worth..., *Hacker Factor Solutions* 6.
- [39] IEEE Information Forensics and Security Technical Committee, Ieee ifs-tc image forensics challenge dataset, "<http://ifc.recod.ic.unicamp.br/fc.website/index.py?sec=5>" (2013).
- [40] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C. Williamson, Estimating the support of a high-dimensional distribution, *Neural computation* 13 (7) (2001) 1443–1471.
- [41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* 12 (2011) 2825–2830.
- [42] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, *ACM Transactions on Intelligent Systems and Technology* 2 (2011) 27:1–27:27, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [43] Z. Wen, J. Shi, Q. Li, B. He, J. Chen, ThunderSVM: A fast SVM library on GPUs and CPUs, *Journal of Machine Learning Research* 19 (2018) 1–5.

# Estimating Probability of Image Features to Support Figure Element Reuse Investigations

Daniel E. Acuna , Ziyue Xiang 

**Abstract**—When there is a suspicious figure element reuse, research integrity investigators often find it difficult to rebut authors saying that “it happens by chance”. In this article, we would like to provide a supporting tool for research integrity investigators, which associates image features with their probabilities to occur in scientific image space. If the probability of the reused image feature is low, then the probability of the reuse happening by chance is also low. To achieve this, we compute the ORB features of all figures in the PMC Open Access Subset and then applying  $k$ -means clustering with 20k centroids on the transformed features. The distribution of image features is acquired by employing probabilistic interpretation on  $k$ -means clustering results.

**Index Terms**—ORB feature, probabilistic model, figure element reuse investigation

## I. INTRODUCTION

Figure element reuse (also known as copy-move forgery) is a significant problem in science. Bik, Casadevall, and Fang [1] manually inspected a total of 20,621 biomedical research papers, only to find out that 3.8 percent of them may contain problematic figures. They also conclude that the prevalence of papers with problematic images has risen markedly during the past decade. In order to assist editors and research integrity investigators with uncovering suspicious reuses, Acuna, Brookes, and Kording [2] proposed an automated reuse detection tool dedicated to this scenario.

However, the fact that the software finds out potential pairs of reuse does not indicate research integrity investigators can convict authors of misconduct confidently. Moreover, when suspicious authors claim that the reuse happens “at random”, it is usually very difficult for one to prove it wrong. Consider the situation illustrated by Figure 1, in which the three images patches are potentially reused patches found by reuse detection software. Intuitively, the probabilities of such reuses occur at random decreases from right to left. If the reused content is text, then it is likely to be a false alarm. If the reused content contains western blots, because it is possible for blots to look extremely close, maybe it is a coincidence. If the suspicion cannot be eliminated, one can require raw data or ask the author to repeat the experiment. Nevertheless, if the reused patch consists of microscopy images (or other highly complicated contents), then the probability of this happening at random is almost zero. It is shown that the content of somehow defines the magnitude of suspicion.

The goal of this article is to determine the relationship between content and suspicion. We quantify the suspicion by measuring the probabilities of features within an image. The overall probabilities of a image patch determines how unlikely

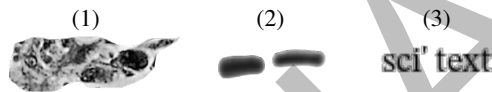


Fig. 1. Assume these three image patches are suspicious reuse regions found by some software. These three patches contain microscopy, western blot and text, respectively. Intuitively, the suspicion decreases from left to right. But it is difficult to justify it. The goal of this article is to provide an estimate of how likely a reuse can happen by chance by studying the probability distribution of image features. The solution to this problem is shown in Figure 5.

a reuse can happen at random, which allows research integrity investigators to reach a verdict with statistical confidence. The flowchart of our work is shown in Figure 2. We acquire all images in the PMC Open Access Subset [3] and compute the top 500 ORB features [4] for each image. We apply dimensionality reduction techniques to reduce the sparsity of ORB features, which also transforms the binary ORB features into numerical ones. In order to get the distribution of all ORB features, we run a large-scale  $k$ -means algorithm on the sampled dataset and employ probabilistic interpretation on the clustering.

In section II, we briefly introduce the mechanisms of ORB features. In section III, we discuss how to apply dimensionality reduction to ORB features and how to build a probability distribution from the  $k$ -means clustering. In section IV, we discuss details of our implementation and analyze the results of our experiments. In section V, we provide a conclusion for this work.

## II. DESCRIBING IMAGE KEYPOINTS WITH ORB FEATURES

ORB is an efficient, public-domain image keypoint detector and descriptor [4]. Compared to other methods such as SIFT [5] and SURF [6], ORB achieves similar performance with much smaller time consumption. The two building blocks of ORB are FAST keypoint detector [7], [8] and BRIEF feature descriptor [9].

As suggested by its name, FAST detectors are widely used because of the rapid computation. However, it does not provide an orientation component. ORB suggests using the intensity centroid [10] as an inexpensive way to estimate the angle of a keypoint. The moments of a patch is given by

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y),$$

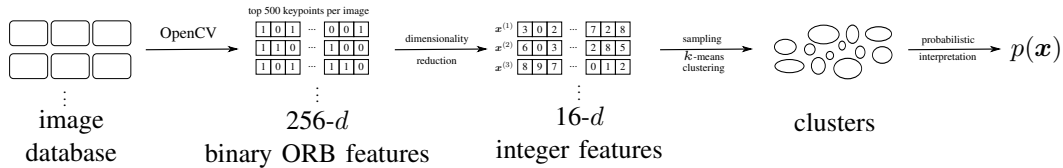


Fig. 2. The flowchart of our work. We acquire all images within the PMC Open Access Subset [3] and compute the top 500 ORB features [4] of each image. We apply dimensionality reduction to decrease the sparsity of ORB data, and the new feature vectors are randomly sampled without replacement. The resulting  $\sim 1.35 \times 10^8$  records are fed into  $k$ -means clustering with 20k clusters. The distribution of ORB features is realized by employing probabilistic interpretation on the clustering results.

where  $I(\cdot)$  denotes the intensity of the pixel at a given point. The intensity centroid of the patch writes

$$P = \begin{pmatrix} m_{10} & m_{01} \\ m_{00} & m_{00} \end{pmatrix}.$$

We can construct a vector from the corner of the patch (denoted by  $O$ ) to  $P$ , and the orientation of the patch is given by the orientation of  $\overrightarrow{OP}$ . It can be seen that the orientation  $\theta$  is

$$\theta = \text{atan2} \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) = \text{atan2}(m_{01}, m_{10}),$$

where  $\text{atan2}$  is the quadrant-aware arctan function. Since FAST does not produce multi-scale features, a scale image pyramid is built where FAST keypoints are detected at each level. Also, due to the fact that FAST does not produce a measure of cornerness, the Harris corner measure [10] is introduced to rank the quality of each detected keypoint.

After acquiring the keypoints, their characteristics are computed by the BRIEF descriptor. The BRIEF descriptor is a bit string representation of an image constructed from a pre-defined set of binary intensity tests. More concretely, assume there are  $T$  point pairs  $(z_1^{(1)}, z_2^{(1)}), \dots, (z_1^{(T)}, z_2^{(T)})$ , where  $z_j^{(i)} = (x_j^{(i)}, y_j^{(i)})$ , the BRIEF descriptor is a  $T$  dimensional binary vector  $\mathbf{B}$ . The  $i$ -th component of  $\mathbf{B}$ , which is denoted by  $B_i$ , is given by

$$B_i = \begin{cases} 1, & \text{if } I(z_1^{(i)}) < I(z_2^{(i)}) \\ 0, & \text{if } I(z_1^{(i)}) \geq I(z_2^{(i)}) \end{cases}.$$

Although BRIEF features are easy and fast to compute, they are not robust against rotations. Calonder, Lepetit, Strecha, *et al.* [9] point out that this problem can be solved by creating a set of rotations and perspective wraps of the image patch, which is computationally expensive. An alternative solution, which is called *steered* BRIEF, transforms the predefined point pairs instead of the patch. In this case, we only need to apply all transformations on the point pairs once to build up a series of precomputed BRIEF templates and then choose the best template during the actual computation step, which eliminates the need of recalculating the image. Eventually, the BRIEF template which has the rotation angle closest to  $\theta$  will be used to compute the BRIEF descriptor.

We compute ORB features with the OpenCV library [11], where the value of  $T$  is set to be 256 by default.

### III. MODELING THE DISTRIBUTION OF ORB FEATURES

Although for our purpose it is intuitive to use a generative clustering model such as Gaussian Mixture Model (GMM), the huge amount of data and clusters make it unfeasible to solve the corresponding optimization problem. Instead, we have to use a more scalable yet non-probabilistic approach: we approximate the distribution of image features using the results of the  $k$ -means algorithm.

#### A. Dimensionality reduction

We observe that the ORB feature space is sparse, as it is supposed to be so for effective feature matching. In order to study its distribution, we can decrease its sparsity by reducing the dimensionality. We apply dimensionality reduction by grouping every 16 bits of the ORB feature into one integer that counts the number of ones in the bits. In this way, the 256 bit binary ORB feature generated by OpenCV is turned into a 16 dimensional feature with numeric values, which allows the use of  $k$ -means algorithm.

Since feature matching with ORB is done by comparing the hamming distance, which is essentially the squared Euclidean distance between bits, it would be desirable that the hamming distance of two ORB feature vectors before transformation is strongly linearly correlated with the squared Euclidean distance after transformation. To study the correlation, for each hamming distance  $d = 1, 2, \dots, 30$ , we generate 200,000 pairs of random ORB feature vectors that have hamming distance  $d$ , and then apply dimensionality reduction to them. Then, we compute the Pearson correlation coefficient between the hamming distance of original vectors and the squared Euclidean distance of transformed vectors. The value of correlation coefficient acquired is  $\rho = 0.807$ , which indicates a strong linear correlation between the two quantities. This suggests that the dimensionality reduction process preserves the similarity of ORB features to some extent.

In the subsequent discussion, the terms “sample” and “feature vector” both refer to the ORB features after dimensionality reduction.

#### B. Probabilistic interpretation of $k$ -means clustering results

Assume that we set the number of clusters in the  $k$ -means algorithm to be  $K$ . Denote the  $j$ -th cluster by  $C_j$ ,

Symbol	Description
$\mathbf{x}_s^{(t)}$	the $s$ -th component of $t$ -th sample
$K$	number of clusters of $k$ -means clustering
$C_j$	the $j$ -th cluster
$ C_j $	the size of $j$ -th cluster
$N$	total number of samples
$M$	the size of a feature vector

TABLE I  
LIST OF SYMBOLS USED IN THIS SUBSECTION

$j = 1, \dots, K$ .  $k$ -means will assign a certain number of samples to each cluster, that is

$$C_j = \emptyset, \quad \text{or}$$

$$C_j = \{\mathbf{x}^{(j_1)}, \dots, \mathbf{x}^{(j_{|C_j|})}\},$$

where  $|C_j|$  denotes the size of  $C_j$ . Let  $N$  be the total number of samples, it is easy to see that

$$N = \sum_{j=1}^K |C_j|.$$

For each cluster  $j$ , we would like to model  $p(\mathbf{x} | C_j)$ . It is common to apply the conditional independence assumption, which indicates

$$p(\mathbf{x} | C_j) = \prod_{i=1}^M p(\mathbf{x}_i | C_j),$$

where  $\mathbf{x}_i$  denotes the  $i$ -th component of the feature vector and  $M$  denotes size of the feature vector. Since each  $\mathbf{x}_i$  takes on discrete values, we can model  $p(\mathbf{x}_i | C_j)$  with categorical distribution. To cope with missing values, we apply add-one smoothing to the parameter estimations. Therefore, we have

$$p(\mathbf{x}_i = a | C_j) = \frac{\left( \sum_{s=1}^{|C_j|} \mathbf{1}\{\mathbf{x}_i^{(j_s)} = a\} \right) + 1}{|C_j| + l},$$

where  $\mathbf{1}\{\cdot\}$  is the indicator function. Once we can compute  $p(\mathbf{x} | C_j)$ , by the law of total probability, we can write

$$p(\mathbf{x}) = \sum_{j=1}^K p(\mathbf{x} | C_j) p(C_j),$$

where

$$p(C_j) = \frac{|C_j|}{N}.$$

In order to estimate the scale of  $p(\mathbf{x})$  more precisely with floating point arithmetic, we use the value of  $\ln p(\mathbf{x})$  instead of  $p(\mathbf{x})$ . Meanwhile, the log-sum-exp trick is applied as follows:

$$\begin{aligned} p(\mathbf{x}) &= \sum_{j=1}^K \exp \{ \ln [p(\mathbf{x} | C_j) p(C_j)] \} \\ &= \sum_{j=1}^K \exp \{ \ln p(\mathbf{x} | C_j) + \ln p(C_j) \} \\ &= \sum_{j=1}^K \exp \left\{ \sum_{i=1}^M \ln p(\mathbf{x}_i | C_j) + \ln p(C_j) \right\}. \end{aligned}$$

Therefore, if we let  $h_j(\mathbf{x})$  be

$$h_j(\mathbf{x}) = \sum_{i=1}^M \ln p(\mathbf{x}_i | C_j) + \ln p(C_j),$$

a more numerically stable value of  $\ln p(\mathbf{x})$  is given by

$$\begin{aligned} \ln p(\mathbf{x}) &= \ln \left[ \sum_{j=1}^K e^{h_j(\mathbf{x})} \right] \\ &= \text{LogSumExp}(h_1(\mathbf{x}), \dots, h_K(\mathbf{x})). \end{aligned}$$

## IV. COMPUTING & ANALYZING THE RESULTS

### A. Getting the distribution

We acquire all 7,636,156 figures inside the PMC Open Access Subset [3], which are considered the sample space of all images. Then, we compute ORB features for the top 500 keypoints within the image and apply the dimensionality reduction technique introduced in section III. For the  $k$ -means algorithm, we use the `kmcuda` software [12], which supports multiple GPU acceleration and a large number of clusters. In our experiment,  $K$  is set to 20,000. Due to memory constraints, the clustering algorithm runs on a randomly sampled subset of all data points, which contains 1/27 of all records. The sampled subset contains approximately  $1.35 \times 10^8$  records.

After clustering, the weight of each cluster ( $p(C_j)$ ) is shown in Figure 3. It can be seen that the weights of most clusters are between 0.00004 and 0.00006, while there are number of clusters whose weights are greater than this range.

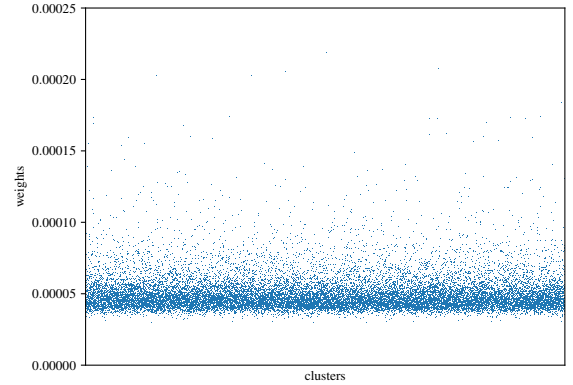


Fig. 3. The weights of each cluster after  $k$ -means clustering.

### B. Computing the ORB feature of any point in an image

So far, we have tried to acquire the distribution of top 500 keypoints within a known image in the data base. Given a new image, we are interested in evaluating the similarity between every single point in the image and existing keypoints in the database, which is described in terms of probabilities.

To compute the ORB feature of an arbitrary point in the image, we can apply modifications to OpenCV's ORB feature computation process. More specifically, for a given point in an image, we compute its Harris response in each layer of the image pyramid and select the layer with the highest

Harris response. The orientation and BRIEF descriptor are then computed within the best layer.

Since it is computationally expensive to estimate the probability of every single point in the image, we sample points from the image uniformly and then apply linear interpolation to the results. In our implementation, the sample distance is 3 pixels.

### C. Analyzing the results

To verify the validity of our results, we select images that only contain three prevalent figure elements in scientific publications and compute the probability of image features inside them. These three types of elements are microscopy (which is close to natural images), western blot and text. Examples of microscopy, western blot and text are shown in Figure 6, 7 and 8 respectively. In Figure 6 and 7, irrelevant regions are masked by black color.

From the results, we can see that microscopy photos are rich in low probability features, for dark blue points can be seen in the probability map frequently. For western blots, their image features are of higher probability, as light blue and white are the dominant colors in the probability map. Features from text images have the highest probability, because red points appear regularly. This is reasonable considering the nature and the frequency of these three types of features in scientific images. As we plot the histogram of image feature probabilities (Figure 4), it can be seen that these three groups of features possess three distinctive distributions. To further study the significance of the result, we compute the mean log probability of each image and use them to form observations for each group. Then, a series of Welch’s  $t$ -tests (Table II) are conducted to measure the difference among the distributions. The  $p$ -values imply with strong confidence that the mean probabilities of image features inside distinctive image groups are different. This conform with our intuitions, as the contents of microscopy images are highly complicated and unlikely to be duplicated within the dataset, which accounts for their lower probability. For blots and text, however, they should have greater probabilities due to higher intra-group feature similarity.

Test objects	$p$ -value
microscopy & western blots	$3.045 \times 10^{-4}$
microscopy & text	$7.825 \times 10^{-26}$
text & western blots	$3.422 \times 10^{-13}$

TABLE II  
P-VALUES OF WELCH’S T-TEST AMONG THE AVERAGE LOG PROBABILITY OF EACH IMAGE GROUP.

Back to the problem raised by Figure 1. Our proposed solution to it is shown in Figure 5. After computing the probability map of image features within a given patch, a research integrity investigator can reach a fact-supported verdict by visually inspecting the output map or by analyzing the mean probabilities.

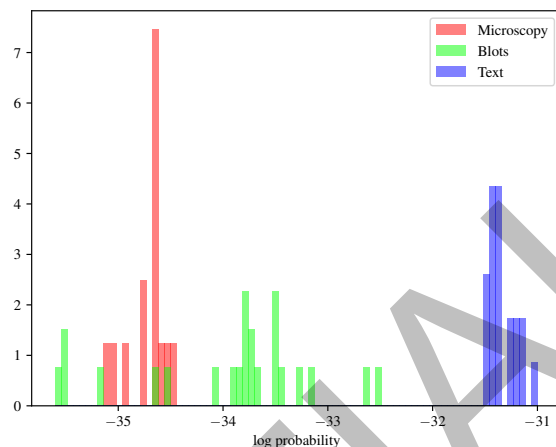


Fig. 4. The histogram of image feature probabilities across three types of images.

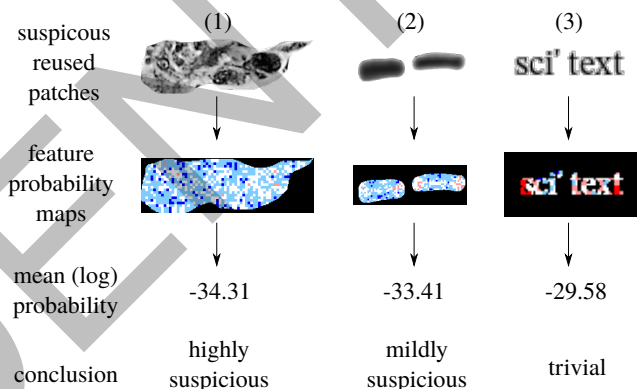


Fig. 5. Examples of using the distribution of ORB features to support figure element reuse investigation.

## V. CONCLUSION

In this paper, we proposed a tool that gives probabilities of image features to support figure element reuse investigations. The probability distribution is acquired by employing probabilistic interpretation on the  $k$ -means clustering of ORB features acquired from the PMC Open Access Subset. To reduce the sparsity of ORB features for better clusterings, we apply similarity-preserving dimensionality reduction to them. The statistical tests on the results of experiments show that various groups of image contents tend to have distinctive probability distributions, and that estimated probabilities of these groups conform with our intuitions.

## REFERENCES

- [1] E. M. Bik, A. Casadevall, and F. C. Fang, “The prevalence of inappropriate image duplication in biomedical research publications,” *MBio*, vol. 7, no. 3, e00809–16, 2016.

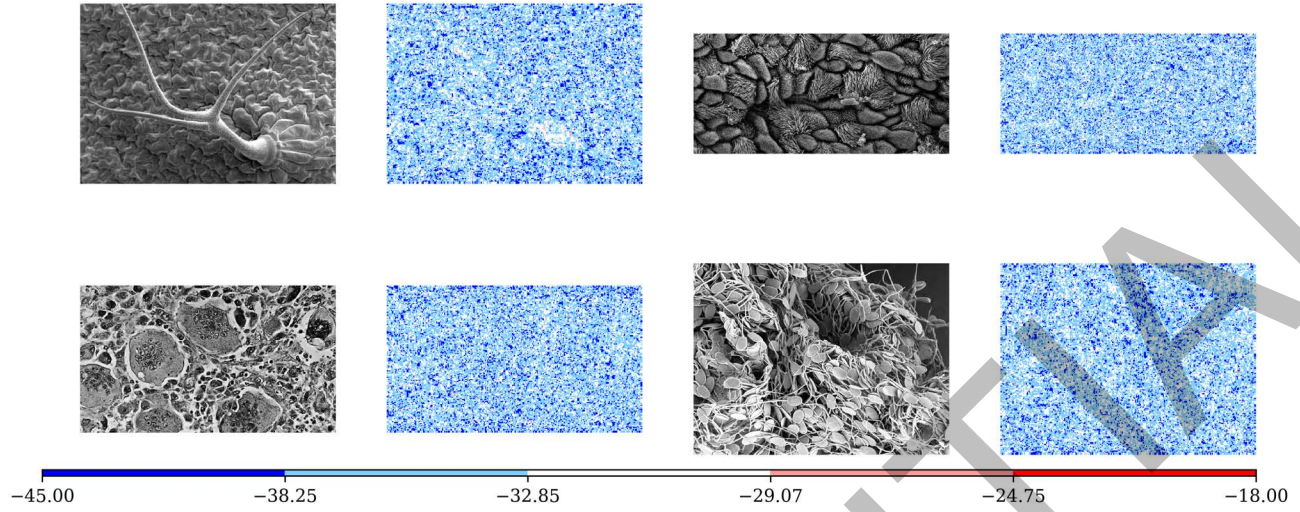


Fig. 6. Examples showing the log probability of microscopy images.

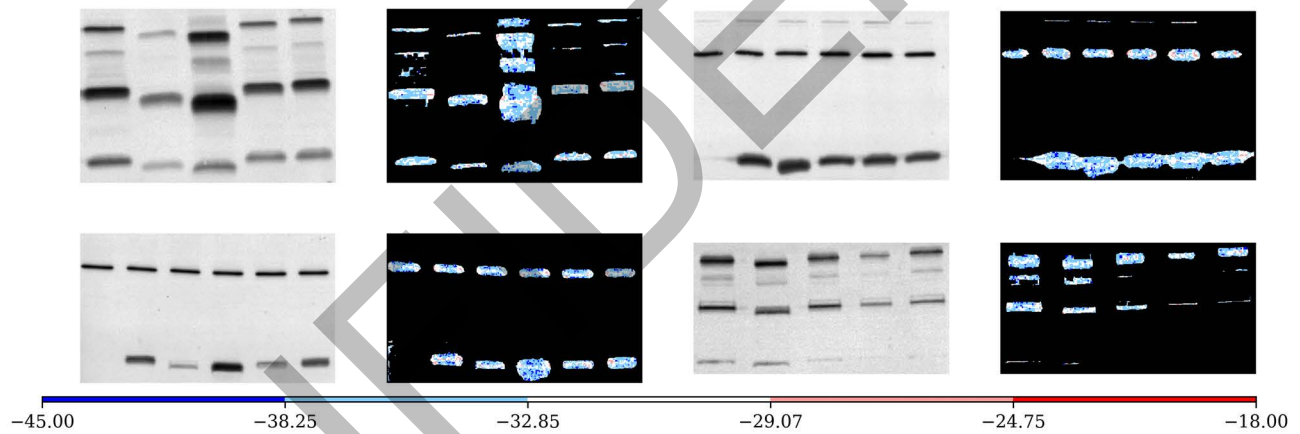


Fig. 7. Examples showing the log probability of western blot images. Irrelevant regions are masked by black color.

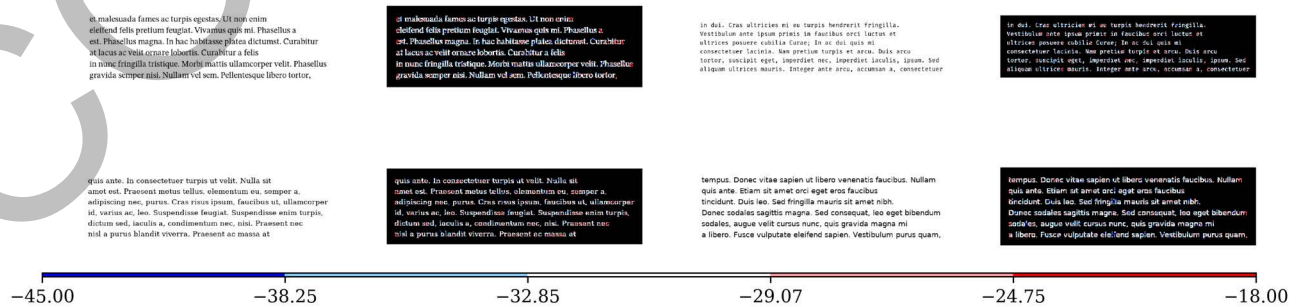



Fig. 8. Examples showing the log probability of text images. Irrelevant regions are masked by black color.



- [2] D. E. Acuna, P. S. Brookes, and K. P. Kording, "Bioscience-scale automated detection of figure element reuse," *bioRxiv*, 2018. DOI: 10.1101/269415. eprint: <https://www.biorxiv.org/content/early/2018/02/23/269415.full.pdf>. [Online]. Available: <https://www.biorxiv.org/content/early/2018/02/23/269415>.
- [3] PubMed Central, *PMC Open Access Subset*. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/>.
- [4] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF.," in *ICCV*, Citeseer, vol. 11, 2011, p. 2.
- [5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [6] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *European conference on computer vision*, Springer, 2006, pp. 404–417.
- [7] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*, Springer, 2006, pp. 430–443.
- [8] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 105–119, 2008.
- [9] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *European conference on computer vision*, Springer, 2010, pp. 778–792.
- [10] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999.
- [11] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [12] V. Markovtsev and M. Cuadros, *src-d/kmcuda: 6.0.0-1*, version v6.0.0, Feb. 2017. DOI: 10.5281/zenodo.286944. [Online]. Available: <https://doi.org/10.5281/zenodo.286944>.

# Better Performance Evaluation For Image Tampering Localization

Ziyue Xiang 

**Abstract**—Image tampering localization is one of the key tasks of image forensics, which is receiving more attention recently as the need of ensuring the authenticity of images increases. However, the performance of related works are usually measured by generic machine learning performance evaluation metrics (e.g. accuracy, AUC, etc.). These metrics assume that samples are independent from each other, which is not the case for images because each image contains millions of pixels that are correlated intrinsically. As a result, there is a discrepancy between metric scores and the perceptual effectiveness. In this paper, we demonstrate such inconsistencies by synthesizing output maps of localization methods that possess a given metric score, allowing readers to experience them visually. We also discuss how to design perceptually consistent metrics, which serve as a better indicator of performance for human users.

This is an ongoing project. The expectations of this project is listed in the last section.

**Index Terms**—image tampering localization, performance evaluation, perceptually consistent metric

## I. INTRODUCTION

Nowadays, the use of digital images has become increasingly ubiquitous. Modern digital formats such as JPEG and PNG greatly compress the sizes of pixel data, therefore reducing the cost of storage and transmission. The advancement of sensor and image processing technologies enable portable devices to produce photos whose qualities are comparable to those taken by high-end digital cameras. As a result, the cover image of magazines or websites can come from a normal person with cell phone instead of a professional photographer with DSLR camera; the proof of a paper document has generally shifted from a scanned copy to a photo copy. What comes with the growing importance of digital images is the development of image tampering techniques. In the old days, modifying or concealing the content of an image would require dedicated personnels and tools. But at present time, after several hours of training, many people can do as well as those image tampering experts with the help of state-of-the-art image processing softwares. Because the quantity of digital images is too large to be examined manually, it is crucial for publishers, government agencies and many other organizations to be able to detect tampered images in an automatic fashion.

Due to the potential values of this problem, numerous attempts have been made to resolve it. However, there has not yet been a reliable industrial-grade solution to this date. Many previous works focuses on tackling specific image manipulation schemes, such as resizing and resampling [1]–[5], median filtering [6]–[9], contrast enhancement [10]–[13], multiple JPEG compression [14]–[17], etc. However, it is non-trivial, if not challenging to merge the results of single

detectors in order to build a comprehensive one. Therefore researchers also try to explore general-purpose image tampering detection techniques. Wang, Dong, and Tan [18] make use of the characteristics of the DCT coefficients to localize image manipulation in JPEG images. Steganalytic tools start to find their ways in image manipulation detection practice [19]–[21]. These tools are also combined with Gaussian Mixture Models (GMM) to identify potentially manipulated clusters in the image [22], [23]. Because of the high dimensionality of image data, neural network based tampering detection method tend to yield good performance [24], especially those based on Convolutional Neural Networks (CNN) [25]–[27], albeit the need of enormous amount of training data.

Because image tapering detection is usually done in pixel or patch level, one should be able to tell the tampered region based on the result as well. We observed the fact that, in recent literature, the performance of tampering localization techniques are evaluated by rather universal metrics, such as accuracy, ROC curve, AUC, F-score and so on. These metrics are designed for binary classification problems where individual samples are independent. However, it is easy to see that in the case of image tampering localization, the image patches (pixels) that are fed into the classifier do not follow the metrics' design. As it will be shown in this article, these metrics not very descriptive when applied to image tampering detection practice.

In section II, we show how image tampering localization techniques are often used and explain why general performance measures become less ideal. In section III, we describe our methodology of recreating the problems of existing performance evaluation schemes. In section III, we demonstrate the problem of existing performance metrics with illustrative examples. In section V, we devise new and better metrics for image tampering detection scenario.

## II. TAMPERING LOCALIZATION, PERFORMANCE EVALUATION AND HUMAN PERCEPTION

Due to the complexity of image tampering localization problem, it is very unlikely for the user of an automatic image tampering localization system to consider its results as a final decision. That is to say, such systems usually serve as an *assistance* to allow human inspectors to make better judgments.

Usually the localization is done at patch level, where a patch is a small region (say,  $10 \times 10$ ) extracted from a rectangular grid in the image (as shown in Figure 1). It is also possible to achieve localization at pixel level, which has a smaller

granularity and therefore higher precision, but there is little difference between these two approaches essentially. Figure 1a shows a pixel level mask, where the black region denotes the tampered region; Figure 1b shows the patch level mask based the pixel level mask and the grid. It can be seen that the patch level approach is less accurate, but it is more commonly used because a patch of image contains more statistical features and reduces the dimensionality of image data (because there are less patches than pixels). For simplicity, in the subsequent discussion, we assume that the localization is done at patch level.

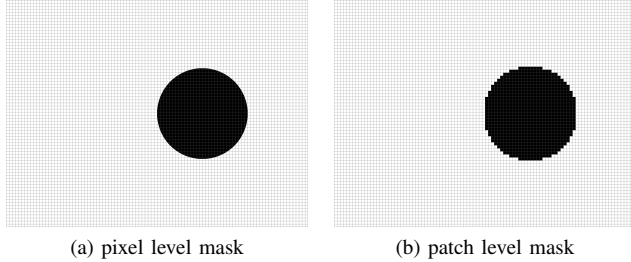


Fig. 1. Illustration of masks of different granularities

In recent literature, authors usually use general purpose machine learning performance evaluation metrics to assess the effectiveness of image tampering localization methods. The commonly used metrics include accuracy, AUC and F-score. However, the image tampering localization scenario is different from normal binary classification problems for the following reasons:

- 1) A single image contains an enormous amount of inputs, whose outputs need to be inspected as a whole
- 2) the inputs to the tampering localization classifier are not independent
- 3) the goal of tampering localization is to generate human perceivable results

These factors render these metrics less reflective in tampering localization scenario. Or at least their values should be interpreted in a different manner.

For reference, we briefly introduce the concepts in performance evaluation below:

- Terminology from a confusion matrix:
  - True Positive ( $TP$ ): positive sample classified as positive
  - True Negative ( $TN$ ): negative sample classified as negative
  - False Positive ( $FP$ ): negative sample classified as positive
  - False Negative ( $FN$ ): positive sample classified as negative
- Numeric derivations from a confusion matrix (each symbol represents the count of that event, and  $P$  and  $N$  denotes the count of all positives and negatives respectively):
  - recall =  $\frac{TP}{P} = \frac{TP}{TP+FN}$   
(a.k.a. true positive rate,  $tpr$ )
  - fall-out =  $\frac{FP}{N} = \frac{FP}{FP+TN}$

(a.k.a. false positive rate,  $fpr$ )

$$\text{precision} = \frac{TP}{TP+FP}$$

$$\text{accuracy} = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{F-score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2TP}{2TP+FP+FN}$$

- Definition of ROC curve and AUC:
  - The ROC curve is created by plotting the recall against the fall-out at various threshold settings.
  - The Area Under the Curve (AUC) is the area under the ROC curve.

### III. GENERATION OF HYPOTHETICAL PREDICTED MASK

We would like to show that the decline of metric effectiveness is not bond to specific classifier. In fact, it is due to the the limitation of performance metrics themselves as they are not designed for this scenario. In order to demonstrate how ubiquitous the problem is, we devise a way to randomly generate predicted masks based on a given performance metric, which can yield the *bad cases* we want at high probability. In all the following examples, it is assumed that the ground truth mask is already known. To show the different visual perception effects of distinctive shapes in masks, we apply the evaluation on several different predefined pixel level masks, which are the illustrated in Figure 2. There corresponding patch level masks can be acquired easily and therefore are not attached.

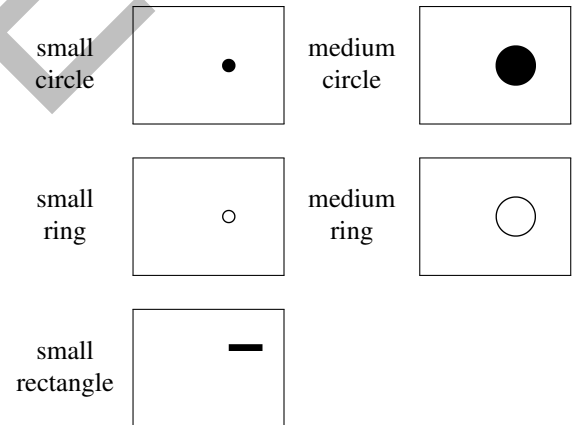


Fig. 2. Pixel level masks used in demonstrations

For the subsequent discussion, the ground truth mask will be denoted by  $g$ , where the value on  $i$ th row,  $j$ th column is given by

$$g(i, j) = \begin{cases} 0, & \text{if the patch is genuine} \\ 1, & \text{if the patch is tampered.} \end{cases} \quad (1)$$

In our illustrations,  $g(i, j) = 0$  is shown by white patches, while  $g(i, j) = 1$  is shown by black patches.

The hypothetical predicted mask will be denoted by  $h$ , which can take on both binary values or real values on  $[0, 1]$  depending on the context. It can be clearly seen that  $h$  has the same dimensionality as  $g$ .

#### A. Generate $h$ given accuracy

It is relatively simple to generate a hypothetical predicted mask  $h$  given an accuracy value  $a$ . In the scenario,  $h$  only needs to take on binary values.

To generate  $h$ , we can sample from a Bernoulli distribution  $\mathcal{B}(p, q)$ , where  $p = a$  and  $q = 1 - a$ . The algorithm can be seen as Algorithm 1.

```

Algorithm 1: Generate  $h$  given accuracy  $a$ 
def get_h_given_acc(g, a):
    p = a, q = 1 - a
    foreach possible (i, j) pair do
        sample a value v from  $\mathcal{B}(p, q)$ 
        if v = 1 then
            h(i, j) = g(i, j)
        else
            // equivalent to flipping the binary bit
            h(i, j) = 1 - g(i, j)
        end
    end
end
return h
    
```

It is easy to see why  $h$  will have accuracy  $a$  approximately, because the event of  $h(i, j)$  being assigned with a correct value has a probability of  $p = a$ .

### B. Generate $h$ given AUC

Because the AUC implies a huge degree of freedom, given an AUC value  $u$ , it is difficult to enumerate all possible ROC curves, which may lead to different visual effects. As our purpose is to create an illustration that loosely represents the given AUC value, we would like to make the following assumptions to make the problem more tractable:

- 1) When the AUC is high, the shapes of ROC curves tend to look alike and therefore can be approximated by a family of curves.
- 2) Suppose an ROC curve is uniformly discretized into a set of points  $P = \{p_1, p_2, \dots, p_n\}$ , where  $P$  is well-sorted by their spatial occurrence from the top right corner to the bottom left corner. We assume that the threshold values are also uniformed distributed (by the length of the ROC curve) from 0 to 1 on these  $n$  points.

The family of curves that is chosen for ROC curve approximation is a simple 3-line-segment scheme. The equation of the first line segment is given by  $l_1 : y = kx$ , where  $k \geq 1$  is the gradient. Because the ROC curve is constrained in a square box, it can be seen that  $l_1$  intersects with the  $(0, 1) \rightarrow (1, 0)$  diagonal at  $p(\frac{1}{k+1}, \frac{k}{k+1})$ . We would like to scale the line segment between the origin and  $p$  by a factor of  $1 - \frac{1}{2k}$ . The second line segment is the symmetry of the first line segment about the same diagonal, and the third line segment connects the previous two line segments. It can be expressed as the piecewise function below:

$$f(x) = \begin{cases} kx, & 0 \leq x < \frac{k-\frac{1}{2}}{k(k+1)} \\ \frac{kx(k+1) + \frac{k(2k-1)}{2} - k + \frac{1}{2}}{k(k+1)}, & \frac{k-\frac{1}{2}}{k(k+1)} \leq x < \frac{3}{2(k+1)} \\ \frac{k+x-1}{k}, & \frac{3}{2(k+1)} \leq x \leq 1. \end{cases} \quad (2)$$

Because the shape of the curve are completely determined by the choice of  $k$ , for simplicity, we shall call it a  $k$ -ROC curve. Different  $k$ -ROC curves and their corresponding AUC values are shown in Figure 3.

When  $k \rightarrow +\infty$ , because  $\lim_{k \rightarrow +\infty} 1 - \frac{1}{2k} = 1$ ,  $p$  and its symmetry will both be close to  $(0, 1)$ . Therefore, it is easy to see that when  $k \rightarrow +\infty$ ,  $u \rightarrow 1$ .

The relationship between  $k$  and  $u$  is given by

$$u = \frac{8k^3 - k + 1}{8k^2(k+1)}. \quad (3)$$

Because it is nontrivial to solve for  $k$  given  $u$ , a  $k-u$  table is attached to help one select a nearest  $k$  value given  $u$ , as shown in Table I.

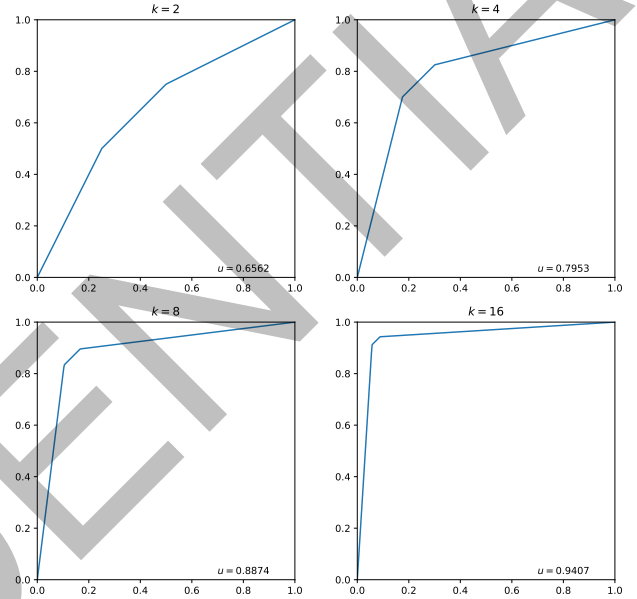


Fig. 3. The ROC curves and AUCs given different  $k$  values

$k$	$u$	$k$	$u$	$k$	$u$
1.0	0.5000	7.0	0.8731	16	0.9407
1.5	0.5889	7.5	0.8807	17	0.9441
2.0	0.6562	8.0	0.8874	18	0.9470
2.5	0.7057	8.5	0.8934	19	0.9497
3.0	0.7431	9.0	0.8988	20	0.9521
3.5	0.7721	9.5	0.9036	30	0.9676
4.0	0.7953	10.0	0.9081	40	0.9755
4.5	0.8143	11	0.9158	50	0.9803
5.0	0.8300	12	0.9223	80	0.9876
5.5	0.8433	13	0.9279	120	0.9917
6.0	0.8547	14	0.9328	160	0.9938
6.5	0.8645	15	0.9370	200	0.9950

TABLE I  
THE  $k-u$  TABLE

We also need to parameterize the  $k$ -ROC curve into  $t \in [0, 1]$  by its length, where  $t = 0$  indicates the bottom left corner and  $t = 1$  indicates the top right corner. More specifically, we need to determine a function  $rpos(k, t)$  that returns the Cartesian coordinate of the parameter  $t$  on the curve. Since  $k \geq 1$ , it is easy to compute that the length of the first line segment, denoted by  $rlen_1(k)$ , is

$$rlen_1(k) = \frac{(2k-1)\sqrt{k^2+1}}{2k(k+1)}. \quad (4)$$

The length of the second line segment equals to the first, and the length of the third line segment, denoted by  $\text{rlen}_3(k)$ , is

$$\text{rlen}_3(k) = \frac{\sqrt{2}}{2k}. \quad (5)$$

Denote the total length of a  $k$ -ROC curve by  $\text{rlen}(k)$ , it can be seen that

$$\text{rlen}(k) = 2 \cdot \text{rlen}_1(k) + \text{rlen}_3(k). \quad (6)$$

By interpolating the segments linearly, we can see that the relationship between the  $x$  component of Cartesian coordinate and  $t$  is as follows:

If  $0 \leq t < \frac{\text{rlen}_1(k)}{\text{rlen}(k)}$ , then

$$x = \left[ t / \frac{\text{rlen}_1(k)}{\text{rlen}(k)} \right] \cdot \frac{k - \frac{1}{2}}{k(k+1)}. \quad (7)$$

If  $\frac{\text{rlen}_1(k)}{\text{rlen}(k)} \leq t < \frac{\text{rlen}_1(k) + \text{rlen}_3(k)}{\text{rlen}(k)}$ , then

$$x = \frac{k - \frac{1}{2}}{k(k+1)} + \left[ \left( t - \frac{\text{rlen}_1(k)}{\text{rlen}(k)} \right) / \frac{\text{rlen}_3(k)}{\text{rlen}(k)} \right] \cdot \left[ \frac{3}{2(k+1)} - \frac{k - \frac{1}{2}}{k(k+1)} \right]. \quad (8)$$

If  $\frac{\text{rlen}_1(k) + \text{rlen}_3(k)}{\text{rlen}(k)} \leq t \leq 1$ , then

$$x = \frac{3}{2(k+1)} + \left[ \left( t - \frac{\text{rlen}_1(k) + \text{rlen}_3(k)}{\text{rlen}(k)} \right) / \frac{\text{rlen}_1(k)}{\text{rlen}(k)} \right] \cdot \left[ 1 - \frac{3}{2(k+1)} \right]. \quad (9)$$

It can be seen that  $\text{rpos}(k, t) = (x, f(x))$ .

Now we can generate  $h$  given  $k$  and  $n$  with Algorithm 2.

Algorithm 2 can indeed generate  $h$  that has a  $k$ -ROC curve, because as the  $fpr$  and  $tpr$  changes between two points of the  $k$ -ROC curve, the algorithm assigns patches with certain values so that the  $fpr$  and  $tpr$  of  $h$  will change in the same way. Once the ROC curve of  $h$  follows the  $k$ -ROC curve,  $h$  will also have the AUC value defined by the  $k$ -ROC curve.

#### IV. VISUAL OBSERVATIONS

*A. Observation 1: big gap in metrics may lead to similar performance in practice*

Figure 4 shows the hypothetical output maps given particular accuracy scores. Figure 5 shows the hypothetical output maps given particular  $k$  values, where each value of  $k$  corresponds to an AUC score. It can be seen that objects are observable in some output maps with lower metric scores.

**Algorithm 2:** Generate  $h$  given  $k$  and  $n$

```

def get_h_given_AUC(g, k, n):
    // assume that n > 1
    thrs = array of n evenly spaced numbers over [0, 1]
    pos = indices of all positive samples in g
    neg = indices of all negative samples in g
    randomly shuffle p and n
    for i = 0 to n - 2 do
        fpr_i, tpr_i = rpos(k, 1 - thrs[i])
        fpr_i, tpr_{i+1} = rpos(k, 1 - thrs[i + 1])
        // the following values are computed
        // according to the fpr/tpr values above
        tpi, fpi = number of true positive/false positive samples
        at thr[i]
        tpi+1, fpi+1 = number of true positive/false positive
        samples at thr[i + 1]
        thr = (thrs[i] + thrs[i + 1]) / 2 // threshold now
        for j = 1 to tpi+1 - tpi do
            ind = the first item popped from pos
            h(ind) = thr
        end
        for j = 1 to fpi+1 - fpi do
            ind = the first item popped from neg
            h(ind) = thr
        end
    end
    return h
    
```

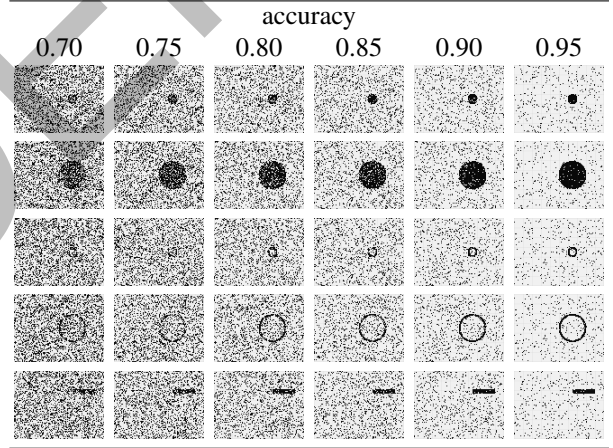


Fig. 4. Predicted mask given different accuracies

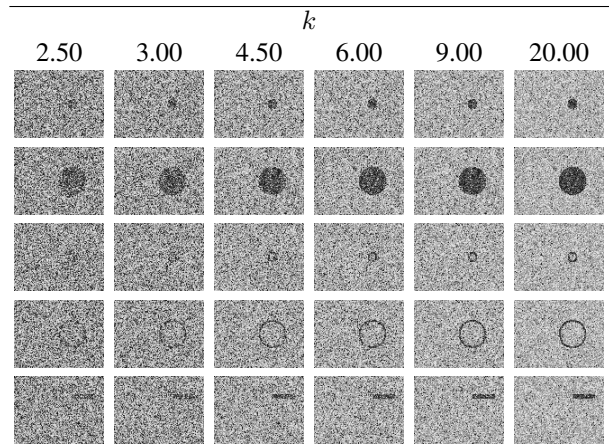


Fig. 5. Predicted mask given different  $k$  values. The conversion between  $k$  and AUC score is given in Table I.

*B. Observation 2: methods with lower metrics may work better*

It is pointed out by Zhou, Han, Morariu, *et al.* [27] that the edges of the tampered region are easier to detect because the statistical patterns change greatly there. What if there is a classifier that is very sensitive to the edges of the tampered region but not the inner? For example, assume that the tampered region is given by medium circle in Figure 2. The output of classifier *A* is shown in Figure 6, and the output of classifier *B* is shown in Figure 7. If we compute the AUC of classifier *B*, the value is only around 0.62, which is lower than the output of classifier *A*, which has an AUC of approximately 0.70. However, the shape of tampered region is much clearer in the output of classifier *B*.

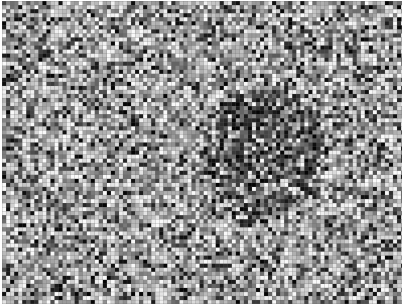


Fig. 6. Output of classifier *A* (second row, first column of Figure 5)

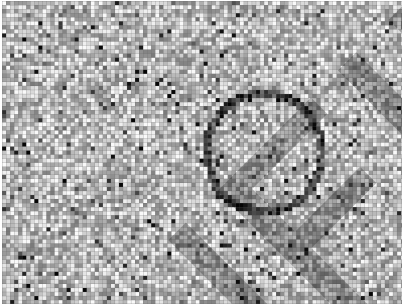


Fig. 7. Output of classifier *B* (fourth row, sixth column of Figure 5)

## V. BETTER PERFORMANCE METRICS

1) *Metric to tackle observation 1:* We make the assumption that it is easier for humans to find the correct tampered region in an output mask because the same pattern is less likely to appear elsewhere. Therefore, we can use a statistical test to evaluate how likely it is for the pattern of the tampered region to appear elsewhere in the output mask.

The statistical test that we choose is the one-sample *t*-test, which is given by

$$T = \frac{\bar{x} - \mu}{s/\sqrt{N}}, \quad (10)$$

where  $\bar{x}$  is the sample mean,  $s$  is the sample standard deviation,  $N$  is sample size and  $\mu$  is the mean that we would like to test against. We select this test because when  $N$  is large,  $T$  will have an approximate standard normal distribution. Hence,

even if  $N$  is different, the value of  $T$  will correspond to the magnitude of tail probability.

The test procedure is given as Algorithm 3. It is worth noticing that in the algorithm, we need to select  $K$  distinct regions. We assume that this can always be satisfied.

### Algorithm 3: Statistical test for observation 1

```

def stat_test (g, h, K):
    Input: K: number of repetitions of the test
           μ = mean of the tampered region in g
           randomly select K distinct regions of the same shape of the
           tampered region on h that are non-overlapping with the
           original tampered region and store them in r
    res = []
    foreach region in r do
        compute the mean of the region
        compute the absolute value of the test statistic and store it
        in res
    end
    return the average value of res

```

With  $K = 100$ , we run Algorithm 3 on outputs in Figure 2. The results are given in Figure 8.

7.14	8.70	11.27	14.50	17.37	23.35
19.36	25.18	35.64	41.77	50.91	65.82
5.36	5.73	9.08	10.54	13.37	16.53
10.34	12.21	16.52	20.17	25.12	33.93
6.89	8.40	12.68	14.80	18.15	23.96

Fig. 8. Metric value computed for Figure 2

## VI. CONCLUSION

So far, we have shown how to generate hypothetical output maps that possess a given accuracy and AUC score. From the output maps, we observe two major inconsistencies, namely:

- 1) big gap in metrics may lead to similar performance in practice
- 2) methods with lower metrics may work better

Then, we discuss how to design a metric that is not affected by the first discrepancy.

This project is still ongoing. We would like to achieve the following:

- Demonstrate the inconsistency for F-measure
- Design a metric that counters Observation 2
- Conduct human experiments to verify the validity of the metrics

## REFERENCES

- [1] A. C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of resampling," *IEEE Transactions on signal processing*, vol. 53, no. 2, pp. 758–767, 2005.
- [2] M. Kirchner, "Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue," in *Proceedings of the 10th ACM workshop on Multimedia and security*, ACM, 2008, pp. 11–20.
- [3] N. Dalgaard, C. Mosquera, and F. Pérez-González, "On the role of differentiation for resampling detection," in *2010 IEEE International Conference on Image Processing*, IEEE, 2010, pp. 1753–1756.

- [4] X. Feng, I. J. Cox, and G. Doerr, "Normalized energy density-based forensic detection of resampled images," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 536–545, 2012.
- [5] B. Mahdian and S. Saic, "Blind authentication using periodic properties of interpolation," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 529–538, 2008.
- [6] M. Kirchner and J. Fridrich, "On detection of median filtering in digital images," in *Media forensics and security II*, International Society for Optics and Photonics, vol. 7541, 2010, p. 754 110.
- [7] X. Kang, M. C. Stamm, A. Peng, and K. 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.
- [8] G. Cao, Y. Zhao, R. Ni, L. Yu, and H. Tian, "Forensic detection of median filtering in digital images," in *2010 IEEE International Conference on Multimedia and Expo*, IEEE, 2010, pp. 89–94.
- [9] C. Chen and J. Ni, "Median filtering detection using edge based prediction matrix," in *International Workshop on Digital Watermarking*, Springer, 2011, pp. 361–375.
- [10] M. C. Stamm and K. 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.
- [11] H. Yao, S. Wang, and X. Zhang, "Detect piecewise linear contrast enhancement and estimate parameters using spectral analysis of image histogram," 2009.
- [12] M. Stamm and K. R. Liu, "Blind forensics of contrast enhancement in digital images," in *2008 15th IEEE International Conference on Image Processing*, IEEE, 2008, pp. 3112–3115.
- [13] M. C. Stamm and K. R. Liu, "Forensic estimation and reconstruction of a contrast enhancement mapping," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2010, pp. 1698–1701.
- [14] T. Bianchi and A. Piva, "Detection of non-aligned double JPEG compression with estimation of primary compression parameters," in *2011 18th IEEE International Conference on Image Processing*, IEEE, 2011, pp. 1929–1932.
- [15] —, "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.
- [16] R. Neelamani, R. De Queiroz, Z. Fan, S. Dash, and R. G. Baraniuk, "JPEG compression history estimation for color images," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1365–1378, 2006.
- [17] Z. Qu, W. Luo, and J. Huang, "A convolutive mixing model for shifted double JPEG compression with application to passive image authentication," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2008, pp. 1661–1664.
- [18] W. Wang, J. Dong, and T. Tan, "Exploring DCT coefficient quantization effects for local tampering detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 10, pp. 1653–1666, 2014.
- [19] 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.
- [20] X. Qiu, H. Li, W. Luo, and J. Huang, "A universal image forensic strategy based on steganalytic model," in *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, ACM, 2014, pp. 165–170.
- [21] T. Pevny, 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.
- [22] W. Fan, K. Wang, and F. Cayre, "General-purpose image forensics using patch likelihood under image statistical models," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2015, pp. 1–6.
- [23] D. Cozzolino, G. Poggi, and L. Verdoliva, "Splicebuster: A new blind image splicing detector," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2015, pp. 1–6.
- [24] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj, and B. Manjunath, "Exploiting spatial structure for localizing manipulated image regions," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4970–4979.
- [25] 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*, ACM, 2016, pp. 5–10.
- [26] —, "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691–2706, 2018.
- [27] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Learning rich features for image manipulation detection," *arXiv preprint arXiv:1805.04953*, 2018.

# Analyzing Robust Features From MNIST

website: [https://github.com/xziyue/robust\\_mnist\\_feature\\_py](https://github.com/xziyue/robust_mnist_feature_py)

## TODOs:

- Implement robust training
- Implement a sufficient amount of perturbation
- Compare performance of std model and robust model
- Implement gradient descent for reconstructing features

## Current Problems

- Convergence: the robust model does not seem to converge well (may need to pretrain the model first)
- Why does horizontal lines hurt accuracy more significantly than vertical lines?

## Goals

- Is it possible to synthesize "robust" features directly?
- Is it possible to differentiate nonrobust and robust features blindly?
- Is it possible to create perturbation that leads to human-readable robust features?

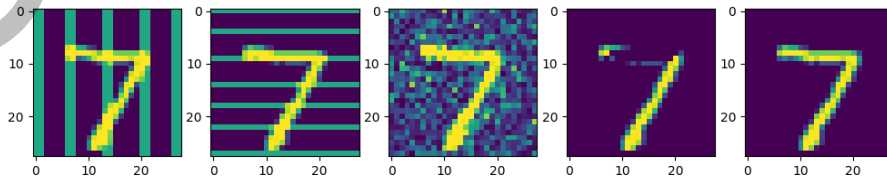
## The dataset

The MNIST dataset is available at <http://yann.lecun.com/exdb/mnist/>.

If you would like to run this script on your computer, go to `/dataset` folder and uncompress all the dataset files to that folder.

## Test results

The perturbed image samples can be seen in figure below. The last column is ground truth. The group IDs correspond to the order of images in the figure.





For evaluation purposes only. Please do not distribute this document.

Group Id	Std Accuracy	Robust Accuracy
1	0.829	0.968
2	0.549	0.967
3	0.808	0.969
4	0.727	0.950
5	0.977	0.972

## Reconstruction

---

CONFIDENTIAL

For evaluation purposes only. Please do not distribute this document.

Original	Reconstruction (Robust)	Reconstruction (Nonrobust)

For evaluation purposes only. Please do not distribute this document.

Original	Reconstruction (Robust)	Reconstruction (Nonrobust)

## File description:

- `perturbation.py`: creates and manages perturbations
- `load_mnist.py`: loading data from MNIST idx format (need to correct endianness if the data format has sizes greater than 1 byte)
- `train_std_model.py`: trains standard model
- `train_pretrained_model`: trains a pretrain model as initial weights for robust model
- `train_robust_model.py`: trains the robust model
- `test_std_model`: tests the performance of std model on adversarial dataset
- `test_robust_model`: tests the performance of robust model on adversarial dataset

## References

- Ilyas, Andrew, et al. "Adversarial examples are not bugs, they are features." *arXiv preprint arXiv:1905.02175* (2019).

CONFIDENTIAL