# Deep Learning for Detecting Processing History of Images

*Mehdi Boroumand and Jessica Fridrich, Department of ECE, SUNY Binghamton, NY, USA, {mboroum1,fridrich}@binghamton.edu*

## Abstract

*Establishing the pedigree of a digital image, such as the type of processing applied to it, is important for forensic analysts because processing generally affects the accuracy and applicability of other forensic tools used for, e.g., identifying the camera (brand) and/or inspecting the image integrity (detecting regions that were manipulated). Given the superiority of automatized tools called deep convolutional neural networks to learn complex yet compact image representations for numerous problems in steganalysis as well as in forensic, in this article we explore this approach for the task of detecting the processing history of images. Our goal is to build a scalable detector for practical situations when an image acquired by a camera is processed, downscaled with a wide variety of scaling factors, and again JPEG compressed since such processing pipeline is commonly applied for example when uploading images to social networks, such as Facebook. To allow the network to perform accurately on a wide range of image sizes, we investigate a novel CNN architecture with an IP layer accepting statistical moments of feature maps. The proposed methodology is benchmarked using confusion matrices for three JPEG quality factors.*

## Introduction

Establishing the processing history of an image is important for an image analyst because forensic tools that attempt to determine image integrity and origin generally exhibit varying degree of sensitivity to non-malicious processing, such as tonal adjustment, denoising, and filtering. Knowing the history of processing is also useful for establishing the chain of custody and provenance of digital evidence in legal cases and for intelligence gathering and interpretation to reveal deception attempts, such as "laundering" of manipulated images. Moreover, a processing detector could be applied on smaller image tiles to detect local inconsistencies due to content replacement (to detect "digital forgeries").

A large bulk of prior art focuses on specific processing, such as median filtering [8, 9, 21, 38], tonal adjustment [31, 32, 2, 12, 37], resizing [11, 13, 16, 19, 22, 20, 24, 27, 28, 29, 35, 36], and multiple JPEG compression [5, 6, 26, 30]. Detection of filtering was investigated in [34]. The problem of recovering the order of processing has been studied from the information-theoretical perspective in [10] and in [33].

A detector of processing that seems to work very well on uncompressed images was recently proposed based on a compactified spatial rich model in [23]. The authors themselves acknowledged that their detector is not robust to JPEG compression, which is one of the main goals of this paper. Stamm et al. [4] trained a convolutional neural network (CNN) with constrained front filters [3] to estimate the parameters of four image processing operations: resampling, JPEG quality, Gaussian blur, and median filtering, by converting the problem of estimation to multi-class detection. Their network was designed to work on patches of fixed size with no other post-processing following the operation whose parameters were to be estimated. Furthermore, the detector was not able to detect tonal adjustment, which is one of the most commonly applied global adjustments.

The main focus of this paper is to build a detector that will work in practical application scenarios, which means for images with a wide range of resolutions, sizes, and JPEG quality factors. For example, imagery posted on Facebook may be downscaled to the larger image dimension as short as 960 pixels and then compressed with JPEG quality factors as low as 70. Downscaling followed by low-quality JPEG is also commonly used for "laundering" used to cover up traces left after manipulating the content of the image. It is well recognized that such laundering significantly decreases the accuracy of the vast majority of forensic algorithms based on pixel descriptors. Since we allow the processing to be "laundered," in this initial study we only classify the type of processing rather than estimating its parameters even though we believe that the detector could probably be extended to estimate the parameters in a similar fashion as in [4]. We work with four basic processing classes – low-pass filtering, high-pass filtering (sharpening), denoising, and tonal adjustment, which includes contrast and gamma adjustment.

This work started with our prior research on this topic [7], which was a maximum-likelihood (ML) detector based on modeling the distribution of projections of a rich feature on eigen-processes determined by binary linear classifiers trained to distinguish between unprocessed images and images processed with a fixed class of processing. In this paper, we propose an alternative detector constructed using the tools of deep learning as a CNN with softmax over five output neurons corresponding to four processing classes and the unprocessed class. The architecture of the CNN is explained in the next section. In the third section, we include a preliminary investigation aimed at evaluating how the CNN detector compares to the previous ML detector on a simple sand-boxed setup. In Section "Practical detector," we explain the process consisting of three phases for training this CNN detector to be able to accurately handle images of arbitrary size and resolution by feeding

statistical moments to the classifier part of the network. The results of all our experiments and their discussion appear in Section "Experiments." The paper is concluded in the last section.

## CNN architecture

Our research into a CNN-based processing detector started with an architecture similar to those used in machine vision and evolved through a series of modifications into the final design shown in Figure 1. The network architecture is depicted for a color input image of size $512 \times 512$, which we call in this paper a *tile*. Later, in Section "Practical detector" we introduce additional modifications to this architecture and describe its three-phase training for a more realistic application scenario and to allow the detector to accurately classify images of arbitrary sizes.

Instead of using larger-support kernels in the first layer, we split it into two layers, each with $3 \times 3$ kernels because such designs train more easily. Note that since there is no non-linearity between the first two convolutional layers, we are essentially learning larger $5 \times 5$ kernels. Experimenting with fixed, constrained, and randomly initialized kernels lead us to the realization that, for best results, *no constraints of any kind* should be imposed on the filters from the first layer. Fixed high-pass kernels or kernels constrained to be high-pass (zero mean) [3], remove information about the image luminance, which can be detrimental for example when trying to detect luminance adjustments, such as contrast and brightness changes or gamma correction.

We tested the performance of the network with various activation functions and with and without the batch normalization layer. The investigation clearly showed the superiority of the ReLU activation function as well as the benefit of the batch normalization (BN) layer that helped with speeding up the training and also improved the overall performance.

Additionally, we found out that the best performance was obtained when disabling pooling between the first two layers, after which the standard $2 \times 2$ average pooling with stride is applied with the exception of the last layer, where average $8 \times 8$ pooling is applied with stride 8 and 0.5 dropout before the fully connected, classification part of the network. We believe that the front part of the network is responsible for "extracting noise residuals" and as such it should be unrestricted as much as possible. Steganalysis as well as forensic analysis makes heavy use of the fine-grain structure in images, such as the acquisition noise properties and short-range dependencies among pixels and the noise residuals. Average pooling suppresses the valuable noise structure, which is our signal of interest, while enhancing the image content, worsening thus the signal to noise ratio.

All filters in the first two layers were initialized randomly using the Xavier initialization with uniform distribution, which was also used for all kernels from all convolutional layers and the weights in the fully-connected IP (inner product) layer.

For a network designed to identify $k$ different processing classes, there should be $k+1$ output neurons corresponding to the $k$ processing classes and the class of unprocessed images. The fully-connected classifier part of the network thus consists of $k+1$ neurons connected to all 1024 features outputted by the last convolutional layer. A softmax over the $k+1$ neurons is taken as the detector's decision. The $k+1$ classes were labeled with integers 0, ..., $k$ for training with the cross entropy loss function.

## Comparison to ML detector

The purpose of the initial experiment in this section is to assess the capability of the proposed CNN in comparison with our previous work [7] that employed a maximum likelihood detector operating in projections of rich feature representations. To this end, we use a rather simple (sandboxed) experimental setup, which is exactly the same as in [7], to limit the distortion applied to the image after processing and to limit the input source diversity. The processing is applied to never-compressed images and then the images are directly saved as JPEGs.

Four types of processing were applied to 10,000 color $512 \times 512$ BOSSbase[1] 1.01 images [1], which were subsequently JPEG compressed with QF 85. The four processing classes included L (low-pass filtering), D (denoising), H (high-pass filtering), and T (tonal adjustment). Each type of processing was represented with eight different operations of varying strength executed in Matlab and Adobe Lightroom. Table 1 shows all processing operations in each of the four classes. The abbreviations in the table stand for: 'Avg' = average filtering, 'Gau' = Gaussian filter, 'KB' = Ker–Böhme $3 \times 3$ filter [18], 'LR' = Lightroom, 'Wie' = Wiener filter, 'Wav' = wavelet based denoising [25], 'Imsh' = imsharpen in Matlab with a pair of parameters `'Radius',`$\sigma$`,'Amount',`$\alpha$, 'Unsh' = unsharp masking, 'Con' = contrast enhancement by $x$%, potentially followed by gamma correction ($\gamma$), with 'HistEq' standing for histogram equalization.

The network was trained in Tensor Flow and run on a Titan X GPU. The stochastic gradient descend (SGD) optimizer was Adam, and the batch size 40 of randomly selected images (with replacement) from the union of all unprocessed and all processed images. Augmentation (rotation by 90 degrees and mirroring) was applied randomly to each image during training. Including the unprocessed image and all its processed versions in the same minibatch and applying same augmentation to them decreased the performance. No weight regularization was applied to the network. A random portion of 5,000 images was used for training, 1,000 of which were used for validation, and 5,000 for testing. In particular, for each unprocessed image in the training set only one processing out of the eight was applied to create one L, H, D, and T image. In other words, the total number of images in the training set was 5,000×5 = 25,000 with the same number in the testing set. The

[1]The images were prepared using the same `convert` script [1] by first converting the RAW format to a 24-bit color PPM format in '`dcraw`', downsampled so that the smaller dimension was 512 using the Lanczos resampling algorithm with antialiasing turned OFF, and centrally cropped to the final size of 512×512 pixels.

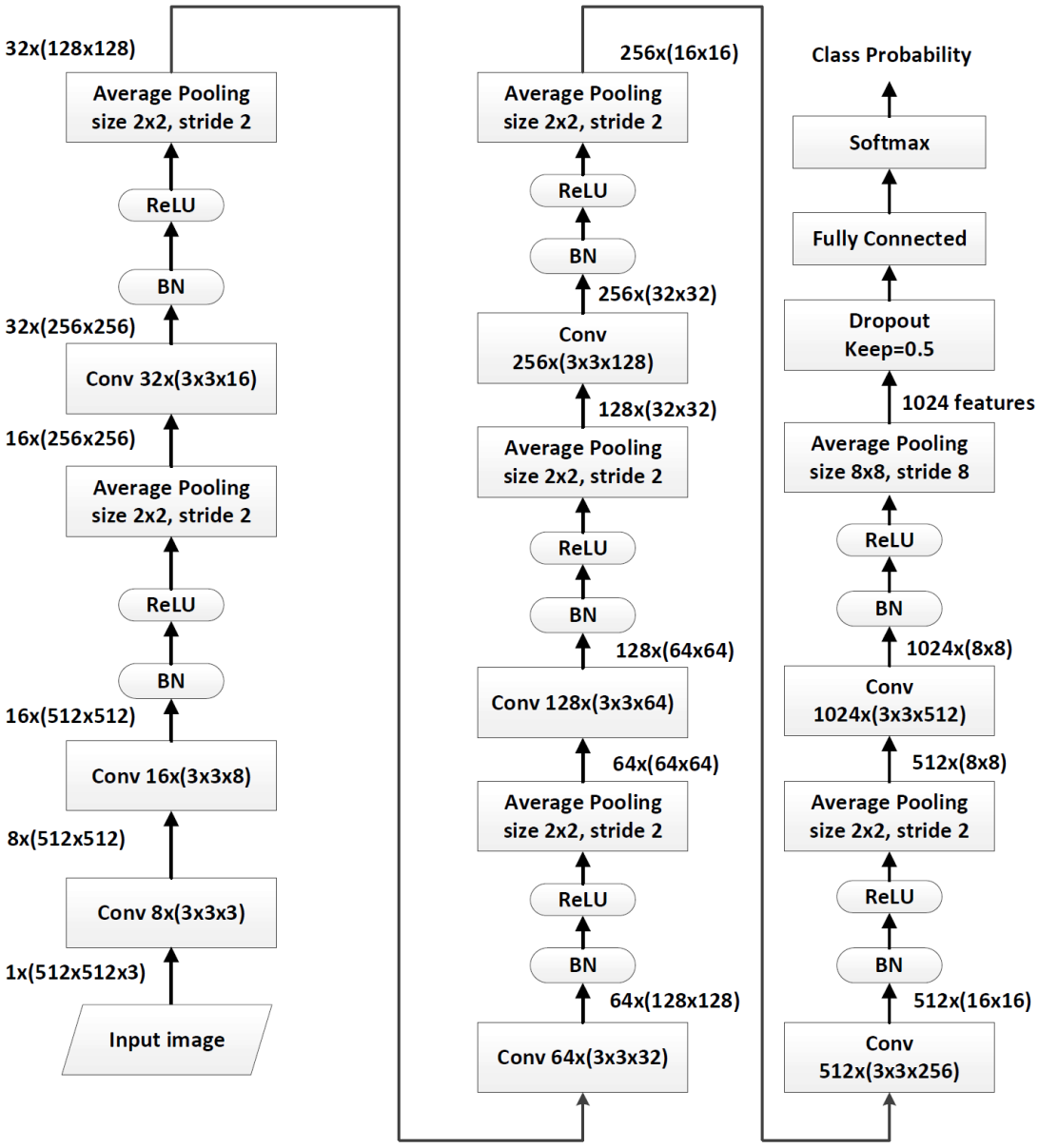


***Figure 1.*** *Architecture of the proposed processing detector.*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| L | Avg3 | Avg5 | Avg7 | Gau3,0.5 | Gau3,1 | Gau5,0.5 | Gau5,1 | \|KB\| |
| D | LR20 | LR30 | LR50 | Wie3 | Wie5 | Wav8 | Wav10 | Wav12 |
| H | Imsh | 1, 1.5 | 1.5, 1.5 | 2, 1.5 | 1, 2 | 1.5, 2 | 2, 2 | Unsh |
| T | Con1% | Con2% | Con4% | Con6% | 2%+0.8 | 4%+0.6 | 6%+1.2 | HistEq. |

**Table 1. All considered processing operations in each of the four processing classes.**

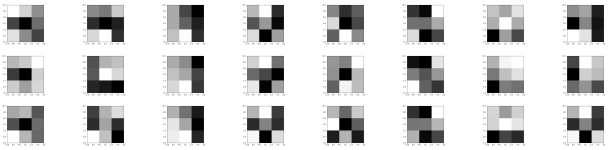

***Figure 2.*** *Kernels learned in the first layer. Sand-boxed setup.*

network was trained for 250K iterations with learning rate 0.001 and another 100K with learning rate 0.0001. The snapshot with the maximum accuracy on the validation set in last 100K iterations was selected as the final detector.

The detection results are displayed via confusion matrices in Table 2 that also contrasts the performance with the ML detector. The average classification accuracy, the average over the diagonal elements in the confusion matrix, is 0.893 for the ML detector and 0.952 for the CNN detector, which indicates a rather significant improvement.

The filters learned in the first layer (Figure 2) can be clearly interpreted as edge and corner detectors, which is natural as we expect a processing detector to mainly use image statistics in textured and generally content-rich areas.

## Practical detector

As already hinted in the introduction and the previous section, the main focus of this paper is on building a processing history detector for realistic situations. This means that we need to consider the fact that the images before applying the processing were most likely JPEG compressed as well as the fact that after the processing was applied, the image could be downsampled by a great range of scaling factors, and then finally saved as JPEG again. As mentioned above, this chain of processing is commonly applied to images uploaded to social networks, such as Facebook. In this specific case, the image after downscaling may only be 960 wide and may be compressed with quality as low as 70. This is a rather *severe distortion* that will likely erase traces of any processing that is too subtle. This is why in this section, we selected only the stronger settings from the classes shown in Table 1.

Also, we need to consider a range of final JPEG quality factors rather than a fixed quality factor. There is no need to diversify the training set over the final JPEG quality factor, however, because the quality factor is available to the analyst. Thus, the diversification can be approached by training separate detectors for several quality factors and then analyze images in practice with the detector trained for the luminance quantization matrix that is closest to the matrices of those detectors. In this section, we trained three separate detectors for the final JPEG quality 75, 85, and 95.

Note that the diversification over the downscaling factor will inevitably lead to images with a wide range of sizes. This poses an additional problem for training because images that are too large would force us to train on very small mini-batches, which would lead to noisy gradients and negatively affect the network accuracy as well as the speed of convergence. Resizing or cropping the images to a smaller size prior to training would negatively affect the network performance because it would reduce the amount of data available to the detector since it is intuitively clear that the detection accuracy should monotonically increase with increasing image size. Moreover, downsizing tends to average out the fine-grain structure in the image that is undoubtedly leveraged by the CNN detector. Applying the detector trained on small images on disjoint or overlapping tiles from a larger image, however, would lead to a non-trivial problem of how to fuse these outputs exhibiting complex dependencies. In this paper, we decided to address the problem of greatly varying input image size by first training a tile detector and then use it as a "moment extrator" and retrain just the IP layers on moments extracted from arbitrarily sized images. This achieved in three phases described below.

### *Introducing moments of feature maps*

As already explained above, to address the problem with the input image size, we applied a similar approach [15] proposed for steganalysis of arbitrarily sized images. There, established scaling laws [17, 14] have been used to show that scalability w.r.t. input image size without loss of performance can be achieved by outputting nonlinear moments by the last convolutional layer. The training of our detector thus proceeds in three separate phases:

**Phase I**: First, a "tile detector" (a CNN) is trained on small images (tiles) obtained by cropping the (arbitrarily downscaled) training images to a fixed size that permits training the CNN on a given hardware. In our case, we cropped two $512 \times 512$ tiles from the center of every training image. The CNN architecture was identical to what is shown in Figure 1 with one modification. Instead of computing just the average of each $8 \times 8$ feature map before the IP layer, we added the minimum, maximum, and variance. Thus, the dimensionality of the input to the IP layer was $4 \times 1024$ instead of 1024.

**Phase II**: In the second phase of training, the front part of the tile detector that outputs $4 \times 1024$ moments to the fully-connected layers is used as a "universal feature extractor" to extract the above four statistical moments from all training images. This time, they are presented in their full resolution, i. e., not cropped. Since there is no training during this second phase, the front part of the tile detector trained in Phase I is merely used to process the arbitrarily sized training images, one by one if needed, and convert each image in the training set to $4 \times 1024$ moments.

**Phase III**: In the final, third phase, a two-layer fully-connected multi-layered perceptron (MLP) (Figure 3) is trained to classify the 4096-dimensional vectors of moments extracted from all training images. The network consists of two fully connected layers, each with 4096 neurons, and five output neurons followed by a softmax, each neuron corresponding to one processing class plus the unprocessed class. We used a two-layer perceptron in case the classification problem becomes non-linear when switching from feature maps to their moments. Since the MLP training

ML detector

| $TRU \setminus DET$ | U | L | D | H | T |
|---|---|---|---|---|---|
| U | 0.871 | 0.041 | 0.034 | 0.008 | 0.047 |
| L | 0.059 | 0.876 | 0.050 | 0.005 | 0.010 |
| D | 0.065 | 0.041 | 0.879 | 0.001 | 0.013 |
| H | 0.020 | 0.022 | 0.0004 | 0.959 | 0.018 |
| T | 0.088 | 0.006 | 0.004 | 0.024 | 0.877 |

Accuracy = 0.893

CNN detector

| $TRU \setminus DET$ | U | L | D | H | T |
|---|---|---|---|---|---|
| U | 0.923 | 0.031 | 0.038 | 0.002 | 0.005 |
| L | 0.023 | 0.966 | 0.011 | 0.000 | 0.000 |
| D | 0.055 | 0.019 | 0.926 | 0.000 | 0.000 |
| H | 0.011 | 0.001 | 0.000 | 0.986 | 0.001 |
| T | 0.032 | 0.003 | 0.002 | 0.003 | 0.959 |

Accuracy = 0.9522

**Table 2. Classification accuracy of the maximum likelihood detector (ML) and the proposed CNN-based detector for four processing classes (Low-pass, High-pass, Denoising, and Tonal). Sand-boxed setup.**

is on moments, we can afford rather large mini-batches (mini-batches of 1000 moment vectors were used in all experiments in this section).

Non-linear moments provide information about the resolution and size of the input image, respectively. Note that there is a difference between the image size, which we measure as the number of pixels, and the image resolution, which relates to the native resolution of the image at acquisition. For instance, a small crop from a high resolution image will keep the smooth nature of a high resolution image (smaller variance of feature maps) despite being small. On the other hand, a resized image (with antialiasing turned off) will have a significantly increased level of detail, which will translate into a larger variance. The order moments monotonically decrease with cropping. Consequently, we believe the four moments provide enough information to the IP layer to allow the CNN adjust itself to accurately classify processing applied to images of arbitrary size and resolution.

### *Processing in each class*

As explained above, due to the severity of the modifications applied to the processed image, we only include the stronger processing operations for training the detector. This is to prevent the network to be presented with examples of images that cannot be in principle distinguished. Below, we summarize the processing diversification in each class.

**Low-pass** class $\mathcal{L}$:
`imfilter(`$X$`,fspecial('gaussian',3,1),'symmetric')`
`imfilter(`$X$`,fspecial('gaussian',5,1.5),'symmetric')`
`imfilter(`$X$`,fspecial('average',3),'symmetric')`

**High-pass** class $\mathcal{H}$:
`imfilter(X,fspecial('unsharp',0.5),'symmetric')`
`imsharpen(X,'Radius',1.5,'Amount',2)`
`imsharpen(X,'Radius',2,'Amount',2)`

**Denoising** class $\mathcal{D}$:
`wiener2(X,[3 3])`
`wiener2(X,[5 5])`
Daubechies 12-tap wavelet denoising [25], $\sigma = 8$

**Tonal adjustment** class $\mathcal{T}$:
The processing involved both contrast adjustment gamma correction, and histogram equalization
`imadjust(X,stretchlim(X,2/100),[],0.8)`
`imadjust(X,stretchlim(X,6/100),[],1.2)`
`histeq(X)`

In the next section, we describe the process used to prepare the dataset for training.

### *Dataset preparation*

We started with 10,000 RAW BOSSbase full-resolution color images and develop them using '`dcraw`' to true-color TIFF images of the same native resolution. Next, all 10,000 images were randomly divided into three disjoint subsets $\mathcal{B}_{Tr}, \mathcal{B}_V$, and $\mathcal{B}_{Ts}$ with 7,000, 1,000, and 2,000 images, respectively, to generate the training, validation, and testing sets. Given the final JPEG quality factor $Q_2$, for each image $\mathbf{I} \in \mathcal{B}_{Tr}$, the following chain of steps was executed while initializing $\mathcal{M}_{Tr} = \emptyset$ and $\mathcal{C}_{Tr} = \emptyset$:

1. Uniformly at random, select the first quality factor $Q_1 \in \{85, \ldots, 98\}$ and compress $\mathbf{I}$ with $Q_1$. Decompress to the spatial domain and denote $\mathbf{X}$.
2. Process $\mathbf{X}$ with all three processing L$\in \mathcal{L}$
3. Process $\mathbf{X}$ with all three processing H$\in \mathcal{H}$
4. Process $\mathbf{X}$ with all three processing D$\in \mathcal{D}$
5. Process $\mathbf{X}$ with all three processing L$\in \mathcal{T}$
6. Consequently, for each $\mathbf{X}$ there will now be 13 images: the original image U, and its three L, H, D, and T versions ($1 + 4 \times 3 = 13$ images).
7. Apply random downscaling to all 13 images so that the largest size is at least 960 pixels.
8. Centrally crop all 13 images to two $512 \times 512$ images.
9. Compress all 13 randomly resized images created in Step 7 with quality factor $Q_2$ and add to database $\mathcal{M}_{Tr}$.
10. Compress all 26 cropped images from Step 8 with quality factor $Q_2$ and add to database $\mathcal{C}_{Tr}$.

After executing the above ten steps for all images from $\mathcal{B}_{Tr}$, we end up with $13 \times 7{,}000 = 91{,}000$ randomly resized color images in $\mathcal{M}_{Tr}$ and twice as many $26 \times 7{,}000 = 182{,}000$ cropped $512 \times 512$ color images in $\mathcal{C}_{Tr}$. The same ten steps were executed with images from $\mathcal{B}_V$ and $\mathcal{B}_{Ts}$ with one modification: only one randomly chosen processing out of three in each class was applied, producing thus 5 images for each image from $\mathcal{B}_V$ and $\mathcal{B}_{Ts}$ instead of 13 or 26. The sets $\mathcal{M}_V$ and $\mathcal{C}_V$ contained 5,000 images while 10,000 images were in $\mathcal{M}_{Ts}$ and $\mathcal{C}_{Ts}$.

**Remark 1**: The random downscaling in Step 7 is carried out so that the number of pixels after resizing, $N_p$, is an integer uniformly distributed in the range $M_{\min} \times N_{\min} \le N_p \le M \times N$, where $M_{\min} = M \times 960/\max(M,N)$ and $N_{\min} = N \times 960/\max(M,N)$ to make sure that the larger size of the downscaled image is at least 960.

**Remark 2**: In Step 8, it may happen that the two cropped images have a small overlap if the larger size of the downscaled image is less than 1024.

### *Phase I: Training moment extractors*

As explained above, the first phase of building the processing detector is to train the moment extractor (tile detector) for the secondary quality factor $Q_2$. In particular, the CNN depicted in Figure 1 with the modification to extract three additional moments, the minimum, maximum, and variance, in addition to the average in the last convolutional layer is trained on $\mathcal{C}_{Tr}$ with the validation and test sets $\mathcal{C}_V$ and $\mathcal{C}_{Ts}$. The variance was treated as a constant during back-propagation for better stability of the training. Note that since all images are $512 \times 512$, there is no issue with image size for training the CNN. We followed the same procedure for training the network as described in the previous section.

### *Phase II: Extracting moments*

In the second phase, the front part of the tile detector trained in Phase I is used to extract moments of all images from $\mathcal{M}_{Tr}$, $\mathcal{M}_V$, and $\mathcal{M}_{Ts}$. The moment extraction was achieved by feeding the images one by one to make sure we do not run out of memory for the potentially large images in these three sets. We repeat that 4096 moments were extracted from each image in this phase. The moments were saved for training the IP layers of the final processing detector in Phase III.

### *Phase III: Training the IP layers*

In the third phase, we train only the IP layers – a MLP with 4096 inputs and two fully-connect layers, each with 4096 neurons, and an output layer with five neurons (see Figure 3) on $\mathcal{M}_{Tr}$, validate on $\mathcal{M}_V$, and test on $\mathcal{M}_{Ts}$, obtaining thus the detection performance of the entire detector. The weights were initialized with the normal distribution with zero mean and standard deviation 0.01. We note that the final detector of processing for JPEG images at quality $Q_2$ consists of the front part of the tile detector trained in Phase I and the IP layers trained in Phase III.

In the next section, we report the classification performance of this detector trained for three JPEG qualities.

## Experiments

This section contains the results of all experiments and their discussion. The detection accuracy is reported using confusion matrices in Table 3 and the overall correct classification accuracy (the average of the diagonal elements of the confusion matrix) corresponding to final JPEG quality factors 75–95. To find out how the detection accuracy depends on the image size, we split the testing set $\mathcal{M}_{Ts}$ into three disjoint sets of equal size containing small (less than 4.5 MP), medium (4.5–9 MP), and large (more than 9 MP) images, and report the average correct detection accuracy (the sum of all diagonal elements from the confusion matrix) in Figure 4. Overall, the accuracy degrades rather gracefully with the JPEG quality factor. It depends more

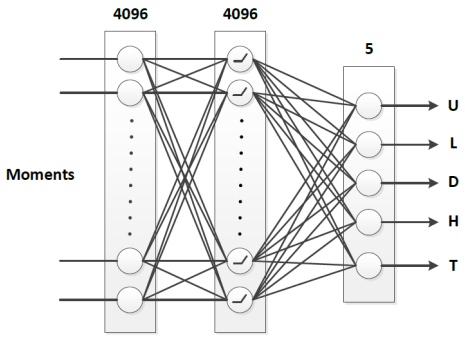


***Figure 3.*** *Two-layer IP trained in Phase III.*

sensitively on the image size as Figure 4 shows. For images with more than 9 megapixels, the detection accuracy is 99.5% or better.

For a processing detector with a larger scope of applicability, i.e., one than can be applied to an arbitrary JPEG image as well as uncompressed images, the plan is to train a family of detectors for selected JPEG qualities $Q_2 \in \mathcal{Q}$. To analyze a given image, first the luminance quantization table $T$ is extracted from the header and then the detector trained for such $Q \in \mathcal{Q}$ whose quantization matrix is closest to $T$ is applied to the image. This will also cover the case of non-standard quantization matrices.

To obtain some insight into how fine the granularization over the QF needs to be (how large the set $\mathcal{Q}$ should be), we applied the three detectors trained for three final JPEG qualities to all three test sets and displayed the results in Table 5. Note that training on a smaller quality factor $Q_1$ and testing on larger $Q_2 > Q_1$ is always better than vice versa. This suggests that the final detector will require a fine granularization (large set of quality factors $\mathcal{Q}$). Moreover when a quantization table is not in $\mathcal{Q}$, it will be better to apply the closest smaller quality factor from $\mathcal{Q}$.

As our final test, we applied the three trained detectors to images processed by operations not included in the training set to assess the ability of the detector to generalize to previously unseen operations: median $3 \times 3$ filtering and color saturation adjustment by 50% (`H = rgb2hsv(X); H(:,:,2) = 1.5*H(:,:,2); H(H > 1) = 1;X' = hsv2rgb(H)`). The results are summarized in Table 4. The detector was correctly able to generalize to median filtering, detecting it as either Denoised or Low-pass filtered with accuracy 94%, 95.3%, and 95.5% for quality factors 75, 85, and 95. On the other hand, since the detector was not trained on any manipulation involving separate processing of color channels, a saturation boost by 50% has been detected as unprocessed. We assume that this behavior is desirable as it is better to not to detect processing than to return a "false alarm."

| $TRU \setminus DET$ | U | L | D | H | T |
|---|---|---|---|---|---|
| U | 0.9040 | 0.0365 | 0.0415 | 0.0090 | 0.0090 |
| L | 0.0185 | 0.9530 | 0.0265 | 0.0015 | 0.0005 |
| D | 0.0140 | 0.0365 | 0.9480 | 0.0010 | 0.0005 |
| H | 0.0040 | 0.0005 | 0.0020 | 0.9935 | 0.0000 |
| T | 0.0045 | 0.0010 | 0.0010 | 0.0000 | 0.9935 |
| QF 75, Accuracy = 0.9584 | | | | | |
| $TRU \setminus DET$ | U | L | D | H | T |
| U | 0.9315 | 0.0270 | 0.0285 | 0.0050 | 0.0080 |
| L | 0.0175 | 0.9595 | 0.0210 | 0.0005 | 0.0015 |
| D | 0.0055 | 0.0195 | 0.9735 | 0.0000 | 0.0015 |
| H | 0.0045 | 0.0005 | 0.0010 | 0.9935 | 0.0005 |
| T | 0.0025 | 0.0000 | 0.0005 | 0.0000 | 0.9970 |
| QF 85, Accuracy = 0.9710 | | | | | |
| $TRU \setminus DET$ | U | L | D | H | T |
| U | 0.9475 | 0.0205 | 0.0175 | 0.0085 | 0.0060 |
| L | 0.0095 | 0.9750 | 0.0150 | 0.0005 | 0.0000 |
| D | 0.0080 | 0.0115 | 0.9795 | 0.0010 | 0.0000 |
| H | 0.0030 | 0.0000 | 0.0000 | 0.9960 | 0.0010 |
| T | 0.0020 | 0.0005 | 0.0000 | 0.0000 | 0.9975 |
| QF 95, Accuracy = 0.9791 | | | | | |

**Table 3. Classification accuracy of the proposed detector on the dataset from Section "Practical detector" for secondary JPEG quality 75, 85, and 95.**

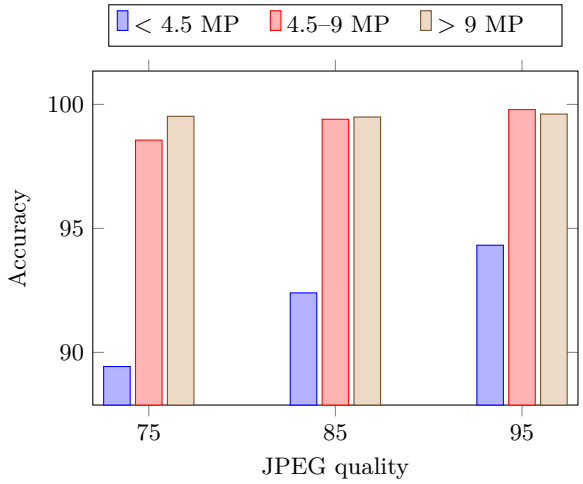


***Figure 4.*** *Correct classification accuracy of the proposed detector for three JPEG quality factors on the testing set $\mathcal{M}_{Ts}$ split into small, medium, and large images generated in Section "Practical detector". See text for more details.*

## Conclusions

This work describes a deep CNN for classifying the type of global processing applied to an image prior to laundering consisting of a potentially agressive downscaling and low quality JPEG compression. Four types of processing is detected: low-pass filtering (blurring), high-pass filtering (sharpening), denoising (content adaptive low-pass filtering), and tonal adjustment, such as histogram equalization, gamma correction, and contrast enhancement.

When designing the detector, attention was paid to make sure that the detector classifies images of arbitrary size with the best possible accuracy (accuracy similar to a detector that could be trained on large images). To this end, we trained the detector in three phases. The first phase involved training a "moment extractor" module on small images ($512 \times 512$ tiles) which was then in Phase II used to extract moments from all (arbitrarily sized) training images. In the final third phase, just the classification part, the IP layers, were trained to map the extracted moments to processing classes.

The detector was trained for three final JPEG quality factors separately. It was able to generalize to previous unseen median filtering and correctly classify it as either low-pass filtering or denoising. It was not, however, able to recognize color saturation boost as a tonal adjustment because this type of processing was not included in training.

As part of our future effort, we intend to build the final detector that can return reliable processing classification for all JPEG quality factors as well as non-standard JPEG quantization tables. The number of processing operations could also be extended to include color saturation and vibrance boost to give the detector the ability to identify this type of adjustment.

All code used to produce the results in this paper, including the network configuration files are available from `http://dde.binghamton.edu/download/`.

| QF | U | L | D | H | T |
|---|---|---|---|---|---|
| Median $3 \times 3$ | | | | | |
| 75 | .051 | .214 | .726 | .004 | .005 |
| 85 | .042 | .193 | .760 | .003 | .004 |
| 95 | .041 | .138 | .817 | .001 | .003 |
| Saturation increase by 50% | | | | | |
| 75 | .871 | .051 | .050 | .014 | .015 |
| 85 | .903 | .034 | .031 | .007 | .027 |
| 95 | .936 | .024 | .017 | .006 | .019 |

**Table 4. Classification of previously unseen processing.**

| $Q_{TRN} \setminus Q_{TST}$ | 75 | 85 | 95 |
|---|---|---|---|
| 75 | .9584 | .9295 | .8015 |
| 85 | .7719 | .9710 | .8427 |
| 95 | .6035 | .7480 | .9791 |

**Table 5. Correct classification accuracy on 2000 test JPEGs with quality $Q_{TST}$ with the detector trained for the final JPEG quality $Q_{TRN}$. The diagonal corresponds to a match between the training QF and the QF of the tested image.**

## Acknowledgments

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

## References


[1] P. Bas, T. Filler, and T. Pevný. Break our steganographic system – the ins and outs of organizing BOSS. In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, *Information Hiding, 13th International Conference*, volume 6958 of Lecture Notes in Computer Science, pages 59–70, Prague, Czech Republic, May 18–20, 2011.

[2] S. Battiato, G. Messina, and D. Strano. Chain of evidence generation for contrast enhancement in digital image forensics. In R. Creutzburg and D. Akopian, editors, *Proceedings of SPIE, Multimedia on Mobile Devices*, volume 7542, page 75420E, San Jose, CA, January 18, 2010.

[3] B. Bayar and M. C. Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In F. Perez-Gonzales, F. Cayre, and P. Bas, editors, *4th ACM IH&MMSec. Workshop*, Vigo, Spain, June 20–22, 2016.

[4] B. Bayar and M. C. Stamm. A generic approach towards image manipulation parameter estimation using convolutional neural networks. In M. Stamm and M. Kirchner, editors, *5th ACM IH&MMSec. Workshop*, Philadelphia, PA, June 20–22, 2017.

[5] T. Bianchi and A. Piva. Analysis of non-aligned double JPEG artifacts for the localization of image forgeries. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, November 29–December 2, 2011.

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

[7] M. Boroumand and J. Fridrich. Scalable processing history detector for JPEG images. In A. Alattar and N. D. Memon, editors, *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2017*, San Francisco, CA, January 29–February 1, 2017.

[8] G. Cao, Y. Zhao, R. Ni, L. Yu, and H. Tian. Forensic detection of median filtering in digital images. In *International Conference on Multimedia and Expo (ICME)*, pages 89–94, Singapore, July 19–23, 2010.

[9] C. Chen and J. Ni. Median filtering detection using edge based prediction matrix. In Y. Q. Shi, H.-J. Kim, and F. Pérez-González, editors, *Digital Forensics and Watermarking, 10th International Workshop (IWDW)*, volume 7128 of *Lecture Notes in Computer Science*, pages 361–375, Berlin, Heidelberg, October 23–26, 2011. Springer-Verlag.

[10] V. Conotter, P. Comesana, and F. Perez-Gonzalez. Forensic detection of processing operator chains: recovering the history of filtered JPEG images. *Information Forensics and Security, IEEE Transactions on*, 10(11):2257–2269, November 2015.

[11] N. Dalgaard, C. Mosquera, and F. Pérez-González. On the role of differentiation for resampling detection. In *IEEE International Conference on Image Processing (ICIP)*, pages 1753–1756, September 26–29, 2010.

[12] H. Farid. Blind inverse gamma correction. *IEEE Transactions on Image Processing*, 10(10):1428–1433, October 2001.

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

[14] T. Filler, A. D. Ker, and J. Fridrich. The Square Root Law of steganographic capacity for Markov covers. In N. D. Memon, E. J. Delp, P. W. Wong, and J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Media Forensics and Security*, volume 7254, pages 08 1–11, San Jose, CA, January 18–21, 2009.

[15] C. Fuji-Tsang and J. Fridrich. Steganalyzing images of arbitrary size with CNNs. In A. Alattar and N. D. Memon, editors, *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2018*, San Francisco, CA, January 29–February 1, 2018.

[16] A. C. Gallagher and T.-H. Chen. Image authentication by detecting traces of demosaicing. In *IEEE Workitorial on Vision of the Unseen (in conjunction with CVPR)*, Anchorage, AK, June 23, 2008.

[17] A. D. Ker. A capacity result for batch steganography. *IEEE Signal Processing Letters*, 14(8):525–528, 2007.

[18] A. D. Ker and R. Böhme. Revisiting weighted stego-image steganalysis. In E. J. Delp, P. W. Wong, J. Dittmann, and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, volume 6819, pages 5 1–17, San Jose, CA, January 27–31, 2008.

[19] M. Kirchner. Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue. In *ACM Multimedia and Security Workshop*, pages 11–20, Oxford, UK, September 22–23, 2008.

[20] M. Kirchner. Linear row and column predictors for the analysis of resized images. In *MM&Sec'10, Proceedings of the 2010 ACM SIGMM Multimedia & Security Workshop*, pages 13–18. ACM Press, September 9–10, 2010.

[21] M. Kirchner and J. Fridrich. On detection of median filtering in images. In *Proc. SPIE, Electronic Imaging, Media Forensics and Security XII*, volume 7542, pages 10 1–12, San Jose, CA, January 17–21, 2010.

[22] M. Kirchner and T. Gloe. On resampling detection in re-compressed images. In *Information Forensics and Security (WIFS). First IEEE International Workshop on*, pages 21–25, December 2009.

[23] H. Li, W. Luo, X. Qui, and J. Huang. Identification of various image operations using residual-based features. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(1):31–45, January 2018.

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

[25] M.K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin. Low-complexity image denoising based on

statistical modeling of wavelet coefficients. *IEEE Signal Processing Letters*, 6(12):300–303, December 1999.

[26] 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*, 15(6):1365–1378, June 2006.

[27] H. C. Nguyen and S. Katzenbeisser. Robust resampling detection in digital images. In B. De Decker and D. W. Chadwick, editors, *Communications and Multimedia Security (CMS)*, volume 7394 of *Lecture Notes in Computer Science*, pages 3–15, Berlin, Heidelberg, September 3–5, 2012. Springer-Verlag.

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

[29] S. Prasad and K. R. Ramakrishnan. On resampling detection and its application to detect image tampering. In *International Conference on Multimedia and EXPO (ICME)*, pages 1325–1328, July 9–12, 2006.

[30] Z. Qu, W. Luo, and J. Huang. A convolutive mixing model for shifted double JPEG compression with application to passive image authentication. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1661–1664, March 31–April 4, 2008.

[31] M. Stamm and K. J. R. Liu. Blind forensics of contrast enhancement in digital images. In *IEEE International Conference on Image Processing (ICIP)*, pages 3112–3115, October 12–15, 2008.

[32] M. Stamm and K. J. R. Liu. Forensic estimation and reconstruction of a contrast enhancement mapping. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1698–1701, March 14–19, 2010.

[33] M. C. Stamm, X. Chu, and K. J. R. Liu. Forensically determining the order of signal processing operations. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 162–167, November 18–21, 2013.

[34] A. Swaminathan, M. Wu, and K. J. R. Liu. Digital image forensics via intrinsic fingerprints. *IEEE Transactions on Information Forensics and Security*, 3(1):101–117, March, 2008.

[35] F. Uccheddu, A. de Rosa, A. Piva, and M. Barni. Detection of resampled images: Performance analysis and practical challenges. In *18th European Signal Processing Conference (EUSIPCO)*, pages 1675–1679, August 23–27, 2010.

[36] R. Wang and X. Ping. Detection of resampling based on singular value decomposition. In *Fifth International Conference on Image and Graphics (ICIG)*, pages 879–884, September 2009.

[37] H. Yao, S. Wang, and 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)*, pages 94–97, December 7–9, 2009.

[38] H.-D. Yuan. Blind forensics of median filtering in digital images. *Information Forensics and Security, IEEE Transactions on*, 6(4):1335–1345, December 2011.

## Author Biography

*Mehdi Boroumand received his B.S. degree in electrical engineering from the K. N. Toosi University of Technology, Iran, in 2004 and his M.S. degree in electrical engineering from the Sahand University of Technology, Iran in 2007. He is currently pursuing his Ph.D. degree in Electrical Engineering at Binghamton University. His areas of research include steganography, steganalysis, digital image forensics, and machine learning.*

*Jessica Fridrich is Distinguished Professor of Electrical and Computer Engineering at Binghamton University. She received her PhD in Systems Science from Binghamton University in 1995 and MS in Applied Mathematics from Czech Technical University in Prague in 1987. Her main interests are in steganography, steganalysis, and digital image forensics. Since 1995, she has received 20 research grants totaling over $11 mil that lead to more than 180 papers and 7 US patents.*