Abstract. Source camera model identification (CMI) and image manipulation detection are of paramount importance in image forensics. In this paper, we propose an L2-constrained Remnant Convolutional Neural Network (L2-constrained RemNet) for performing these two crucial tasks. The proposed network architecture consists of a dynamic preprocessor block and a classification block. An L2 loss is applied to the output of the preprocessor block, and categorical crossentropy loss is calculated based on the output of the classification block. The whole network is trained in an end-to-end manner by minimizing the total loss, which is a combination of the L2 loss and the categorical crossentropy loss. Aided by the L2 loss, the data-adaptive preprocessor learns to suppress the unnecessary image contents and assists the classification block in extracting robust image forensics features. We train and test the network on the Dresden database and achieve an overall accuracy of 98.15%, where all the test images are from devices and scenes not used during training to replicate practical applications. The network also outperforms other state-of-the-art CNNs even when the images are manipulated. Furthermore, we attain an overall accuracy of 99.68% in image manipulation detection, which implies that it can be used as a general-purpose network for image forensic tasks.

Keywords: Image Forensics, Camera Model Identification, Image Manipulation Detection, Convolutional Neural Networks

1 Introduction

Camera model identification (CMI) and image manipulation detection are crucial tasks in image forensics with applications in criminal investigations, authenticating evidence, detecting forgery, etc. Digital images go through various camera-internal processing before being saved in the device \cite{27}. Moreover, they are often manipulated after they leave the device that has been used to capture them. Nowadays, professional image editing tools like Adobe Photoshop, ACDsee, and Hornil Stylepix are readily available, consequently making image manipulation a common phenomenon \cite{13}. Also, images undergo different kinds of manipulations when they are shared online. We have observed a proliferation of digitally altered images with the advent of modern technologies. When the
authenticity of such images is questioned, a forensic analyst has to answer two questions first, what is the source of the image under question and whether the image has been manipulated. The image metadata cannot be trusted as a reliable source, as this data can be forged. Therefore, a forensic analyst resorts to different image forensics techniques to answer these questions.

Image forensics is an active research area, and several methods exist in the literature for finding out the source camera model and detecting image-processing operations of a questioned image. But researches are conducted discretely for finding out the source and manipulation history of an image. In [40], [32], we can find a brief overview of the approaches proposed over the last two decades. We see that initial research in CMI has focused on merging image-markers, such as watermarks, device-specific code, etc. [32]. However, using separate external features for each camera model is an unmanageable task [17]. Consequently, researchers have focused on utilizing the intrinsic features, such as the Color Filter Array (CFA) pattern [5], interpolation algorithms [25], and Image Quality Metrics (IQM) [21]. Utilizing Photo Response Non-Uniformity (PRNU) noise patterns have been proposed for device-level identification [15], [20]. Although sensor noise carries device-specific noise artifacts, researchers have developed methods to perform CMI using sensor noise patterns [12], [28]. Most of these approaches attempt to extract camera model-specific features and compare the features with a pre-calculated reference for the corresponding camera model [9].

In the case of image manipulation, traces are found in the image according to the type of processing it has gone through [4]. Following this theory, researchers have used distinct forensic approaches for identifying different kinds of image manipulation, such as resizing [33], [18], contrast enhancement [39], [50], and multiple jpeg compression [6], [30], etc. The drawback of using the above-mentioned statistical feature-based approaches is that the performance degrades sharply, when new cases arise that have not been considered during feature vector selection [13]. For that reason, more recent researches have focused on becoming data-driven, such as utilizing local pixel dependencies used in steganalysis [19], [31] to perform CMI [29], [10] and detect image manipulation [34]. In [16], the authors propose a Gaussian mixture model for image manipulation detection. Though these approaches provide good results, extracting features for different manipulations requires substantial computational resources, and the performance degrades severely depending on the size of the questioned image [13].

Recently, researchers have started applying Convolutional Neural Networks (CNNs) for image forensic tasks [48]. It is expected as CNNs have performed extremely well in different image classification tasks [37]. Usually, CNNs tend to learn features related to the content of an image, whereas, for image forensics, we need to refrain CNNs from learning image contents [3]. As a result, a common practice while using CNNs in digital image forensics is adding a preprocessing layer at the beginning of the CNN architecture. Chen et al. [11] have proposed using a median filter, whereas Tuama et al. [44] have used a high-pass filter before feeding images in their respective CNNs. However, such crude filtering is not supported by the literature as the artifacts introduced by different camera-
internal processing and manipulations can lie in both low and high frequency
domain [28]. Therefore, fixed filters as preprocessor may lose forensics-related
features. Bayar and Stamm [3] have proposed a data-driven constrained con-
volutional layer which has performed better than the above-mentioned fixed filters.
Rafi et al. [36] have used a completely data-driven preprocessor block followed
by a classification block to perform CMI. Bayar and Stamm [3] have also used
their constrained CNN for image manipulation detection. However, some CNN
based approaches do not use any preprocessing scheme. Yang et al. use the
idea of multi-scale receptive fields on an input image to perform CMI [49]. In
[7], the authors use CNN and support vector machine (SVM) for CMI, where
they use the CNN part as a feature extractor. In [35], explores the performance
of DenseNet [24] in both CMI and image manipulation detection. In [13], the
authors investigate the performance of densely connected CNNs in image ma-
nipulation detection. Owing to the performance of the data-driven preprocessing
schemes, it can be inferred that further researches need to be conducted to make
the preprocessing operations more robust for image forensic tasks. Several re-
searches exist in the literature that use auxiliary loss function to enhance the
discrimination between learned features [46], [45], [41]. There is a scope of uti-
lizing such auxiliary loss functions in the modular CNN architectures for image
forensics.

Despite the numerous researches conducted in this field, most researchers
have explored CMI and image manipulation detection problems discretely. Bayar
and Stamm [3] show that it is possible to use the same approach for both tasks.
Therefore, research for coming up with a general-purpose neural network suitable
for both CMI and image manipulation detection requires more attention. Also,
strict measures should be followed while conducting experiments so that the
proposed methods can be applied in real-life scenarios. Kirchner and Gloe suggest
that the test set should always consist of images captured by devices that have
not been used during training or validation [27]. Also, the scenes in the test set
should be different from those used during training and validation. Here, scene
refers to a combination of a location and a specific viewpoint. Keeping separate
devices and scenes in the test set is compulsory for replicating real-life conditions
and making the result reliable for practical applications. These evaluation criteria
will ensure that the neural network is free from data leakage [1] during testing
and can not overperform by learning features specific to the device or scene. Beside,
the performances of CMI and image manipulation detection should be
measured using images manipulated at different intensities. We strictly follow
the above-mentioned points in our experiments.

In this paper, we propose a general-purpose novel CNN architecture, called
L2-constrained Remnant Convolutional Neural Network (L2-constrained Rem-
Net) for performing two crucial tasks in image forensics, CMI and image manip-
ulation detection. Our proposed CNN has two parts, a preprocessor block and a
classification block. The preprocessor architecture consists of several data-driven
remnant blocks, and an L2 loss is applied to the output of the preprocessor block.
A CNN based classification block follows the preprocessor block, and categorical
crossentropy loss is calculated based on its output. The total loss function is a combination of the L2 loss and the categorical crossentropy loss. The whole network is trained end-to-end while minimizing the total loss. The L2-constrained preprocessor learns to suppress image contents making it easier for the classification block to extract image forensics features. Our experiments show that the proposed method can outperform other state-of-the-art networks in both image forensic tasks.

We organize the rest of the paper as follows. Section 2 contains a description of our proposed CNN and loss function. We discuss our training and evaluation criteria, along with the experimental results in Section 3. Finally, we conclude in Section 4.

2 Proposed Method

In this paper, we propose a CNN-based patch-level method for CMI and image manipulation detection. A schematic representation of our proposed method is shown in Fig. 1.

As shown, we first extract high quality clusters of size $256 \times 256$ from an input image. From each cluster, patches of size $64 \times 64$ are taken and fed to the L2-constrained RemNet. It then generates a class probability map for each patch. We assign a camera model or image manipulation type label to each cluster by averaging the class probability maps of its patches. The final prediction is made based on the majority voting on the labels of the clusters of an image.

As well known, CNNs in their standard form tend to learn content-specific features from the training images. In designing CNNs for image forensic tasks, it has been, therefore, a common practice to use a preprocessing scheme to suppress the image contents and intensify the minute signatures induced by the image acquisition pipeline or image manipulation operation. Unlike the conventional

Fig. 1. Schematic representation of the proposed method for CMI and image manipulation detection.
approaches, the benefit of designing a dynamic preprocessing block is that it can adapt itself optimally to perform different image forensic tasks. To this end, we propose a general-purpose novel CNN architecture, called L2-constrained RemNet. A data-driven preprocessing block coupled with L2 loss is used at the beginning of the network, which is followed by a classification block. A line diagram of the proposed network model is shown in Fig. 2. The details of our proposed model are presented in the following.

2.1 Preprocessing block

We use the remnant blocks proposed in [36] as our preprocessing block. The architecture is influenced by the highway networks [38]. The inherent camera
model-specific features are very subtle and minute features of the image \([40], [28], [12]\). The problem of diminishing minute model-specific features is alleviated in the remnant block through the use of skip connections. Also, it refains the minute features from being lost in a layer. Moreover, it resolves the vanishing gradient problem \([43]\) during training. The use of activation is avoided in the remnant blocks as per the design requirements in \([3]\). Also, it is motivated by our wish to make them perform as optimal digital filters. The final layer of a remnant block is subtracted from its input in a pixelwise manner. This subtraction helps regulate information flow. While choosing the depth of a remnant block, the number of filters in each convolutional layer, and kernel size– we use the hyperparameters proposed in \([36]\). The architecture of the remnant block is illustrated in Fig. 3.

Each remnant block has three convolutional layers. The kernel size is chosen as \(3 \times 3\). Each layer is followed by BN (batch normalization). The feature space is widened from \(64 \times 64 \times 3\) to \(64 \times 64 \times f_i\) in the first two convolutional layers and then reduced to the original dimension again. The output of the last convolutional layer is subtracted from the input. As the convolutional layers are followed by BN, the input to the block is batch normalized as well. To preserve input information throughout a block, the input is propagated to every convolutional layer inside the block. We use three remnant blocks in total. 64, 128, and 256 are chosen as \(f_i\) for the consecutive remnant blocks.

### 2.2 Classification Block

We use the classifier proposed in \([36]\) as our classification block as well. The output of the preprocessing block is passed a classification block. The architecture of the classification block is provided in Table 1. It extracts higher-level image forensics features by gradually reducing the dimensions of the feature space, and finally provide a class probability of the source camera model or the manipulation type of the input image.

The classification block starts with four convolutional layers. Each of the first four convolutional layers is followed by a BN layer and a PReLU activation. The output of the fourth convolutional layer is followed by an average-pooling operation. Lastly, a final convolutional layer with softmax activation is used to generate a probability for the final prediction. The design choices for the classification block are the same as the hyperparameters proposed in \([36]\). There are no fully connected layers in the classification block, which keeps the number of parameters less. Consequently, the network is less prone to overfitting and trains within a shorter time.

### 2.3 Loss Function

The preprocessing block contains \(M\) remnant blocks. The \(i\)-th remnant block applies a transformation \(H_i\) on its input \(x_i\) (which is also the output of the \((i - 1)\)-th remnant block) and subtracts it from its input to produce the output
Table 1. Architecture of Our Proposed L2-constrained RemNet

| Layers           | Output Size | Kernels* |
|------------------|-------------|----------|
| Remnant Block 1  | 64×64×3     | $f_1 = 64$ |
| Remnant Block 2  | 64×64×3     | $f_2 = 128$ |
| Remnant Block 3  | 64×64×3     | $f_3 = 256$ |

Classification Block

|                      |            |          |
|----------------------|------------|----------|
| Conv 2D, BN, & PReLU | 32×32×64  | $F = 64$, $K = 7×7, S = 2$ |
| Conv 2D, BN, & PReLU | 16×16×128 | $F = 128$, $K = 5×5, S = 2$ |
| Conv 2D, BN, & PReLU | 8×8×256   | $F = 256$, $K = 3×3, S = 2$ |
| Conv 2D, BN, & PReLU | 4×4×512   | $F = 512$, $K = 2×2, S = 2$ |
| Average Pool         | 1×1×512   | $K = 4×4$ |
| Conv 2D              | 1×1×N     | $F = N$, $K = 1×1$, $S = 1$ |

Softmax $N$

* Here, $F$, $K$, and $S$ represent the number of filters, kernel size, and strides, respectively. $N$ represents the number of class.

$y_{p_i}$:

$$y_{p_i} = x_i - H(x_i, W_{p_i}),$$  \(1\)

The output of the last remnant block is $y_{p_M}$. A loss is calculated based on a flattened version of this output:

$$L_2 = \sum_{i=1}^{N_{\text{param}}} y^2_{p_{Mi}}.$$  \(2\)

Here, $y_{p_{Mi}}$ is the $l$-th element of $y_{p_M}$ and $N_{\text{param}}$ is the total number of elements in $y_{p_M}$. Afterwards, $y_{p_M}$ is fed the classifier block that applies a transformation $G$ to generate the final output $y_c$:

$$y_c = G(y_{p_M}, W_c).$$  \(3\)

We calculate categorical crossentropy loss between this output and the ground truth using:

$$L_{\text{xent}} = \sum_{k=1}^{N_{\text{class}}} y^*_c(k) \log \left( \frac{y_c(k)}{y_{c_i}} \right).$$  \(4\)

where $y^*_c(k)$ and $y_c(k)$ are the true label and the network output of the $i$-th image at the $k$-th class among the $N_{\text{class}}$ classes, respectively. The total loss $L$ is defined using the following equation:

$$L = \alpha \ast L_2 + L_{\text{xent}}.$$  \(5\)
Here, $\alpha$ indicates how much weight we want to put in the suppression of the residue from the preprocessor block. A larger choice for $\alpha$ may cause the vanishing gradient problem for the classifier \[43\]. We empirically set the value of $\alpha$ as 0.5. During backpropagation, the gradient of $L_2$ is used to update the weights of the preprocessor block. The gradient of $L_{xent}$ is used to update the weights of both the preprocessing block and the classifier block. The whole network is trained in an end-to-end manner. The preprocessing block outputs a residue of the input, and $L_2$ attempts to minimize this output, which results in suppression of image contents. Simultaneously, the classifier tries to extract useful features from this residue for accurate predictions to minimize $L_{xent}$. Minimization of $L$ results in rich image forensics features in the residue for the classifier block.

3 Experimental Results

We perform a number of experiments to prove the efficacy of our proposed method. We discuss the experiments and the results in this section.

3.1 Camera Model Identification

We evaluate our L2-constrained RemNet on Dresden Dataset \[22\]. The dataset contains images captured with 73 devices of 27 different camera models. Multiple snaps have been captured from different scenes for each device. We discard eight camera models to choose the specific camera models, which have images captured using more than one device. Our goal is to keep one device excluded during training and use it only for testing. Also, we consider Nikon D70 and Nikon D70s, as a single camera model, according to \[27\]. We end up with 18 camera models. We split the dataset into train, validation, and test sets following strict criteria that the camera device and scenes used during testing are not used for training or validation. This results in 7938, 1353, and 540 images in the train, validation, and test set, respectively. These criteria, proposed in \[27\] and used in \[7\], is quite necessary to make sure that the evaluation is not biased owing to device-specific features and scene-specific features.

Data augmentation is a commonly used method in deep learning to reduce overfitting. Recently, researchers have started using it for CMI as well \[48\], \[35\], \[36\]. Also, our goal is to perform CMI from both unaltered and manipulated images. Therefore, we choose different image manipulations as our data augmentations, as proposed in \[36\]. The types of augmentation that we use in this work are:

- JPEG-Compression with a quality factor of 70\%, 80\%, and 90\%
- Resizing by a factor of 0.5, 0.8, 1.5, and 2.0
- Gamma-Correction with a factor of $\gamma = 0.8$ and 1.2

This increases our data by nine folds. Afterward, we extract clusters of $256 \times 256$ size from the images. However, saturated and flat regions inside an image
are not less likely to contain features related to CMI. Therefore, we follow the selection strategy proposed in [7], [36] to extract high quality image clusters. For every cluster $P$ in an image, its quality $Q(P)$ is computed as

$$Q(P) = \frac{1}{3} \sum_{c \in \{R,G,B\}} \left[ \alpha \cdot \beta \cdot (\mu_c - \mu^2_c) + (1 - \alpha) \cdot (1 - e^{\gamma \sigma_c}) \right]$$

where $\alpha$, $\beta$, and $\gamma$ are empirically set constants (set to 0.7, 4 and ln(0.01), respectively), $\mu_c$ and $\sigma_c$, $c \in \{R,G,B\}$ are the mean and standard deviation of the red, green, and blue components of cluster $P$, respectively.

Although we extract $256 \times 256$ sized high quality clusters, we use $64 \times 64$ input size for our network according to [49], [7], [51], [36]. Patches of $64 \times 64$ are randomly selected from a cluster of $256 \times 256$ during training. This strategy introduces statistical variations during training which is discussed in detail in [36]. We extract 20 clusters of size $256 \times 256$ from each image and this results in $1587600$ and $270600$ train and validation clusters. We use our custom loss function (see subsection 3.3) and Adam [26] optimizer with exponential decay rate factors $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The choice for our batch size is 64. The learning rate starts with $10^{-3}$ and we decrease it with a factor of 0.5 if the softmax classification loss ($L_{xent}$) does not decrease in three successive epochs. We train our network for a maximum of 70 epochs and save the weight with the least validation softmax classification loss for evaluation.

The test set contains $10800$ clusters of size $256 \times 256$ from $540$ full images. During testing, we average the predictions on all non-overlapping patches of size $64 \times 64$ to make a prediction for a cluster and assign a camera model label $\hat{L}_n$ to it. We use majority voting to make the final prediction $\hat{L}$ for the full image. The metric we use for evaluating the performance of our models is provided in the following equation:

$$\text{Accuracy} = \frac{N_C}{N_T}.$$  

| Method                          | Dresden Dataset |
|--------------------------------|-----------------|
| Yang et al. [49]                | 95.19           |
| Bayar and Stamn. [3]            | 93.89           |
| Bondi et al. [7]                | 92.59           |
| DenseNet [24]                   | 95.05           |
| ResNet [23]                     | 95.19           |
| ResNeXt [47]                    | 95.55           |
| RemNet without preprocessing block [36] | 95.74 |
| RemNet [36]                     | 97.59           |
| **Proposed Method**             | **98.15**       |
Table 3. Accuracy (in %) of different methods in CMI for manipulated test dataset

| Method          | Gamma Correction | JPEG Compression | Resize Scale |
|-----------------|------------------|------------------|--------------|
|                 | 0.5   | 0.75  | 1.25 | 1.5  | 95    | 90   | 85   | 80   | 0.8   | 0.9   | 1.1   | 1.2   |
| Yang et al. [49] | 94.26  | 93.37 | 95.00 | 92.78 | 94.07 | 94.07 | 92.59 | 92.59 | 94.26 | 92.59 | 90.93 | 90.56 |
| Bayar and Stamm. [3] | 93.52 | 94.44 | 94.44 | 94.63 | 92.59 | 94.81 | 88.15 | 85.74 | 87.04 | 64.44 | 59.07 |
| Bondi et al. [7] | 85.92  | 91.85 | 89.07 | 92.03 | 84.07 | 85.92 | 91.48 | 90.74 | 92.56 | 92.77 | 91.48 | 89.44 |
| DenseNet [24]    | 91.66  | 95.18 | 92.03 | 94.62 | 92.77 | 92.96 | 94.26 | 94.81 | 95.00 | 94.81 | 94.44 | 94.26 |
| ResNet [23]      | 91.85  | 95.18 | 92.77 | 94.81 | 93.88 | 94.82 | 95.55 | 95.00 | 95.18 | 95.18 | 95.00 | 95.18 |
| ResNeXt [47]     | 94.25  | 95.55 | 93.88 | 95.18 | 95.18 | 94.82 | 94.25 | 94.07 | 95.00 | 96.11 | 95.55 |
| RemNet [36]      | 96.11  | 97.22 | 96.11 | 95.56 | 97.59 | 94.82 | 92.59 | 92.78 | 95.00 | 93.33 | 92.04 | 92.41 |
| Proposed Method  | 96.29  | 98.14 | 97.59 | 97.96 | 92.96 | 93.33 | 96.11 | 97.03 | 96.67 | 96.67 | 90.74 | 91.66 |

Here, \( N_C \) is the number of correct prediction and \( N_T \) is the total number of test images. We also compare our results with four other state-of-art CNNs in CMI [3], fusion residual networks [49], [7], and [36]. Moreover, we provide comparison with two other popular deep CNNs, ResNet [23] and DenseNet [24] as they have been used for image forensics as well [14], [2], [8], [35]. We use the same input size for all the networks for fair comparison.

At first, we evaluate the performance of the models on the unaltered test dataset. L2-constrained RemNet achieves an overall accuracy of 98.15%, which is better than all other approaches we compare with (see Table 2). It should be noted that we set the value for \( \alpha \) in our custom loss function (5) empirically. We have achieved accuracy of 97.77%, 98.15%, and 97.77%, when \( \alpha \) is chosen as 0.1, 0.5, and 1, respectively. Therefore, we propose using \( \alpha = 0.5 \).

We perform several experiments to justify the use of the L2-constrained preprocessing block in our network. First, we train the RemNet without any preprocessing block at the beginning of the network, that is, we only train the classification block. Then, we train the RemNet without any auxiliary L2 loss at the output of the preprocessing block. Afterward, we experiment with replacing the L2 loss with the L1 loss. The lower accuracy of the RemNet without the preprocessing block justifies the use of the preprocessing step. Similarly, the lower accuracy of RemNet without any additional loss justifies the use of the auxiliary loss. When we use the L1 loss in our custom loss function, the total loss oscillates throughout the training and does not converge. After a complete run, the L1-constrained RemNet attains an accuracy of 58.88%. The L1 loss enforces sparsity on the output of the preprocessing block, whereas the image forensics features, in this case, are non-sparse and present throughout the image. The
L2 loss forces the output of the preprocessing block to be small and provides a non-sparse solution.

Furthermore, we apply various manipulations on the test set and evaluate the performance of our method. To make sure that the network has not overfitted on the manipulation factors used during training, we also manipulate the test images with factors that are not used during training. The test images are created using gamma correction with $\gamma = 0.5, 0.75, 1.25, \text{ and } 1.5$; JPEG compression quality factors (QFs) 95%, 90%, 85%, and 80%; and resize scaling factor of 0.8, 0.9, 1.1, 1.2. The highest result for each manipulation factor is made bold (see Table 3). We can see that our proposed method has substantial improvement over other methods for Gamma Correction. In the case of JPEG Compression, our network achieves better performance for two factors, and RemNet \cite{36} achieves better performance in two. For Resize manipulation, we see that ResNeXt \cite{47} gains higher accuracy for two manipulation factors, whereas our proposed method gains higher accuracy in the other two factors. We can conclude that our proposed method proves to be most robust to external manipulation. Also, deep CNNs perform better than shallow networks in the face of manipulated images.

### 3.2 Image Manipulation Detection

Now, we show the use of our network in a completely different image forensic task. We use it to identify the kind of image-manipulation done on an image. The same network is used here except the number of output classes, which is four—unaltered, rescale, JPEG compression, and gamma correction. The input size for all the networks is also maintained at $(64 \times 64)$. We use the same train and validation set from our experiments with CMI and sub-divide it into the four manipulation classes. The L2-constrained RemNet is then trained to detect the type of manipulation applied to an image. It is to be mentioned that, during training, our dataset consisting of 1587600 train and 270600 validation clusters has been reduced in order to make the training data evenly distributed among four classes. Since the number of unaltered train and validation clusters are 158760 and 27060, respectively, we select 158760 train and 27060 validation clusters randomly for each type of manipulation.

In testing, we have used the test images from the Dresden dataset and generated a total of $540 \times 12 = 6480$ test images, which include 540 unaltered images; $540 \times 4 = 2160$ gamma-corrected images with $\gamma = 0.5, 0.75, 1.25, \text{ and } 1.5$; $540 \times 3 = 1620$ JPEG compressed images compressed with factors of 85%, 90%,

| Method          | Dresden Dataset |
|-----------------|-----------------|
| Yang et al. \cite{49} | 91.74           |
| Bayar and Stamm \cite{3}   | 87.28           |
| RemNet \cite{36}     | 98.27           |
| **Proposed Method** | **99.68**       |
Table 5. Accuracy (in %) of image manipulation detection for different manipulation factors

| Method       | Gamma Correction | JPEG Compression | Rescale |
|--------------|------------------|------------------|---------|
|              | 0.5 | 0.75 | 1.25 | 1.5 | 95 | 90 | 85 | 0.8 | 0.9 | 1.1 | 1.2 |
| Yang et al. [49] | 99.07 | 98.52 | 97.04 | 98.70 | 49.44 | 100 | 100 | 100 | 97.40 | 60.74 | 100 |
| Bayar and Stamm [3] | 94.44 | 83.33 | 77.22 | 90.56 | 11.30 | 100 | 100 | 100 | 100 | 90.93 | 99.63 |
| RemNet [36] | 100 | **99.81** | **99.63** | **100** | 81.48 | 98.33 | 100 | 100 | 100 | 100 | 100 |
| Proposed method | **100** | **99.63** | **99.26** | **98.7** | **100** | **98.7** | **100** | **100** | **100** | **100** | **100** |

and 95%; and $540 \times 4 = 2160$ resized images images with scaling factor of 0.8, 0.9, 1.1, and 1.2. Details of the results are given in Table 5. We achieve an overall accuracy of 99.68% in this task whereas RemNet [36], Bayar and Stamm [3], and Yang et al. [49] achieve 98.27%, 87.28% and 91.74%, respectively (see Table 4). We demonstrate the detection accuracy for different factors of manipulation in Table 5. For gamma-corrected images, the performances of [49], RemNet [36] and our proposed method are substantially better than that of [3]. In the case of JPEG compression, all four networks perform almost the same except at the compression factor of 95, where [3] and [49] fail miserably by misclassifying most of the compressed images as unaltered images. There is a significant drop in the detection accuracy for RemNet [36] as well. This is expected since there is very little difference between the original image and JPEG compressed image with factor 95. However, our proposed method achieves 100% accuracy even at this factor, which indicates that the network can detect even minute manipulation artifacts introduced during manipulation operation. When detecting rescaled images, our network and RemNet [36] performs the same by attaining a 100% accuracy. Of the other two networks, [3] performs better than [49].

4 Conclusion

In this paper, we have proposed an L2 loss constrained RemNet for performing two important image forensics tasks, namely, CMI and image manipulation detection. The proposed modular CNN model comprises of a dynamic preprocessor and a classification block in series. The L2-constrained preprocessor, while trained end-to-end along with the classification block, suppresses unnecessary image contents dynamically and generates a residue of the image from where the classification block can easily extract image forensics features. We have comprehensively conducted multiple experiments on the Dresden dataset to demonstrate the efficacy of such a preprocessing scheme assisted by the L2 loss in CMI. During testing, we use images captured by devices not seen during training to replicate practical applications. The results of the experiments have shown that
our proposed method can be successfully used in real-world scenarios. Additionally, we have used our proposed method for image manipulation detection. The satisfactory performances of our network on both classification tasks prove that it can be used for a general-purpose network for image forensics.

References

1. Alneyadi, S., Sithirasenan, E., Muthukkumarasamy, V.: A survey on data leakage prevention systems. Journal of Network and Computer Applications 62, 137–152 (2016)
2. Barni, M., Costanzo, A., Nowroozi, E., Tondi, B.: Cnn-based detection of generic contrast adjustment with jpeg post-processing. In: Proc. IEEE Int. Conf. on Image Process. (ICIP). pp. 3803–3807 (Oct 2018). https://doi.org/10.1109/ICIP.2018.8451698
3. Bayar, B., Stamm, M.C.: Design principles of convolutional neural networks for multimedia forensics. Electronic Imaging 2017(7), 77–86 (2017)
4. Bayar, B., Stamm, M.C.: Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection. IEEE Trans. Inf. Forensics Secur. 13(11), 2691–2706 (2018)
5. Bayram, S., Sencar, H., Memon, N., Avcibas, I.: Source camera identification based on cfa interpolation. In: Proc. IEEE Int. Conf. on Image Process., (ICIP). vol. 3, pp. III–69. IEEE (2005)
6. Bianchi, T., Piva, A.: Image forgery localization via block-grained analysis of jpeg artifacts. IEEE Transactions on Information Forensics and Security 7(3), 1003–1017 (2012)
7. Bondi, L., Baroffio, L., Güera, D., Bestagini, P., Delp, E.J., Tubaro, S.: First steps toward camera model identification with convolutional neural networks. IEEE Signal Process. Lett. 24(3), 259–263 (2017)
8. Boroumand, Mehdi, F.J.: Deep learning for detecting processing history of images. Electronic Imaging (2018)
9. Cao, H., Kot, A.C.: Accurate detection of demosaicing regularity for digital image forensics. IEEE Trans. Inf. Forensics Secur. 4(4), 899–910 (2009)
10. Chen, C., Stamm, M.C.: Camera model identification framework using an ensemble of demosaicing features. In: Proc. IEEE Int. Works. Infor. (WIFS). pp. 1–6. IEEE (2015)
11. Chen, J., Kang, X., Liu, Y., Wang, Z.J.: Median filtering forensics based on convolutional neural networks. IEEE Signal Process. Lett. 22(11), 1849–1853 (2015)
12. Chen, M., Fridrich, J., Goljan, M., Lukás, J.: Determining image origin and integrity using sensor noise. IEEE Trans. Inf. Forensics Secur. 3(1), 74–90 (2008)
13. Chen, Y., Kang, X., Shi, Y.Q., Wang, Z.J.: A multi-purpose image forensic method using densely connected convolutional neural networks. Journal of Real-Time Image Processing 16(3), 725–740 (2019)
14. Chen, Y., Kang, X., Wang, Z.J., Zhang, Q.: Densely connected convolutional neural network for multi-purpose image forensics under anti-forensic attacks. In: Proc. 6th ACM Workshop Inf. Hiding Multimedia Secur. pp. 91–96. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3206004.3206013
15. Dirik, A.E., Sencar, H.T., Memon, N.: Source camera identification based on sensor dust characteristics. In: Proc. IEEE Workshop Signal Process. Appl. Public Secur. Forensics. pp. 1–6. IEEE (2007)
16. Fan, W., Wang, K., Cayre, F.: General-purpose image forensics using patch likelihood under image statistical models. In: 2015 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1–6. IEEE (2015)

17. Farid, H.: Image forgery detection. IEEE Signal Process. Mag. 26(2), 16–25 (2009)

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

19. Fridrich, J., Kodovsky, J.: Rich models for steganalysis of digital images. IEEE Transactions on Information Forensics and Security 7(3), 868–882 (2012)

20. Fridrich, J., Lukas, J., Goljan, M.: Digital camera identification from sensor noise. IEEE Trans. Inf. Forensics Secur. 1(2), 205–214 (2006)

21. Gloe, T.: Feature-based forensic camera model identification. In: LNCS Trans. Data Hiding and Multimed. Secur. VIII, Vol. 7228 of Lect. Notes Comput. Sc., pp. 42–62. Springer (2012)

22. Gloe, T., Böhme, R.: The dresden image database for benchmarking digital image forensics. J. Digital Forensic Practice 3, 150–159 (01 2010)

23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR). pp. 770–778 (2016)

24. Huang, G., Liu, Z., v. d. Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR). pp. 2261–2269 (July 2017). https://doi.org/10.1109/CVPR.2017.243

25. Kharrazi, M., Sencar, H.T., Memon, N.: Blind source camera identification. In: Proc. IEEE Int. Conf. on Image Process., (ICIP). vol. 1, pp. 709–712. IEEE (2004)

26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

27. Kirchner, M., Gloe, T.: Forensic camera model identification. Proc. WOL Handbook of Digital Forensics of Multimedia Data and Devices pp. 329–374 (2015)

28. Lukas, J., Fridrich, J., Goljan, M.: Digital camera identification from sensor pattern noise. IEEE Trans. Inf. Forensics Secur. 1(2), 205–214 (2006)

29. Marra, F., Poggi, G., Sansone, C., Verdoliva, L.: A study of co-occurrence based local features for camera model identification. Multimedia Tools and Applications 76(4), 4765–4781 (2017)

30. Neelamani, R., De Queiroz, R., Fan, Z., Dash, S., Baraniuk, R.G.: Jpeg compression history estimation for color images. IEEE Transactions on Image Processing 15(6), 1365–1378 (2006)

31. Pevny, T., Bas, P., Fridrich, J.: Steganalysis by subtractive pixel adjacency matrix. IEEE Transactions on information Forensics and Security 5(2), 215–224 (2010)

32. Piva, A.: An overview on image forensics. Proc. ISRN Signal Process. 2013 (2013)

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

34. Qiu, X., Li, H., Luo, W., Huang, J.: A universal image forensic strategy based on steganalytic model. In: Proceedings of the 2nd ACM workshop on Information hiding and multimedia security. pp. 165–170 (2014)

35. Rafi, A.M., Kamal, U., Hoque, R., Abrar, A., Das, S., Laganière, R., Hasan, M.K.: Application of densenet in camera model identification and post-processing detection. In: CVPR Workshops. pp. 19–28 (2019)

36. Rafi, A.M., Tonmoy, T.I., Kamal, U., Hoque, R., Hasan, M., et al.: Remnet: remnant convolutional neural network for camera model identification. arXiv preprint arXiv:1902.00694 (2019)

37. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks 61, 85–117 (2015)
38. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
39. Stamm, M.C., Liu, K.R.: Forensic detection of image manipulation using statistical intrinsic fingerprints. IEEE Trans. Inf. Forensics Secur. 5(3), 492–506 (2010)
40. Stamm, M.C., Wu, M., Liu, K.R.: Information forensics: An overview of the first decade. IEEE Access 1, 167–200 (2013)
41. Sun, Y., Wang, X., Tang, X.: Deeply learned face representations are sparse, selective, and robust. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2892–2900 (2015)
42. Thai, T.H., Cogranne, R., Retraint, F.: Camera model identification based on the heteroscedastic noise model. IEEE Trans. Image Process. 23(1), 250–263 (2014)
43. Tong, T., Li, G., Liu, X., Gao, Q.: Image super-resolution using dense skip connections. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4799–4807 (2017)
44. Tuama, A., Comby, F., Chaumont, M.: Camera model identification with the use of deep convolutional neural networks. In: Proc. IEEE Int. Workshop on inf. Forensics and Secur. (WIFS). pp. 1–6. IEEE (2016)
45. Wen, Y., Li, Z., Qiao, Y.: Latent factor guided convolutional neural networks for age-invariant face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4893–4901 (2016)
46. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: European conference on computer vision. pp. 499–515. Springer (2016)
47. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017)
48. Yang, P., Baracchi, D., Ni, R., Zhao, Y., Argenti, F., Piva, A.: A survey of deep learning-based source image forensics. Journal of Imaging 6(3), 9 (2020)
49. Yang, P., Zhao, W., Ni, R., Zhao, Y.: Source camera identification based on content-adaptive fusion network. Pattern Recog. Lett. 119, 195–204 (2019)
50. Yao, H., Wang, S., Zhang, X.: Detect piecewise linear contrast enhancement and estimate parameters using spectral analysis of image histogram (2009)
51. Yao, H., Qiao, T., Xu, M., Zheng, N.: Robust multi-classifier for camera model identification based on convolution neural network. IEEE Access 6, 24973–24982 (2018)