Noname manuscript No.
(will be inserted by the editor)

RemNet: Remnant Convolutional Neural Network for Camera Model Identification

Abdul Muntakim Rafi[1] · **Thamidul Islam Tonmoy**[2] · **Uday Kamal**[3] · **Q. M. Jonathan Wu**[1] · **Md. Kamrul Hasan**[3*]

Received: date / Accepted: date

Abstract Camera model identification (CMI) has gained significant importance in image forensics as digitally altered images are becoming increasingly commonplace. In this paper, a novel convolutional neural network (CNN) architecture is proposed for CMI with emphasis given on the preprocessing task considered to be inevitable for removing the scene content that heavily obscures the camera model fingerprints. Unlike the conventional approaches where fixed filters are used for preprocessing, the proposed remnant blocks, when coupled with a classification block and trained end-to-end minimizing the classification loss, learn to suppress the unnecessary image contents dynamically. This helps the classification block extract more robust camera model-specific features for CMI from the remnant of the image. The whole network, called RemNet, consisting of a preprocessing block and a shallow classification block, when trained on 18 models from the Dresden database, shows 100% accuracy for 16 camera models with an overall accuracy of 97.59% on test images from unseen devices, outperforming the state of the art deep CNNs used in CMI. Furthermore, the proposed remnant blocks, when cascaded with the existing deep CNNs, e.g., ResNet, DenseNet, boost their performances by a large margin. The proposed approach proves to be very robust in identifying the source camera models, even if the original images are post-processed. It also achieves an overall accuracy of 95.11% on the IEEE Signal Processing Cup 2018 dataset, which indicates its generalizability.

Keywords Digital Image Forensics · Camera Model Identification · Convolutional Neural Networks · Remnant Block

Abdul Muntakim Rafi
E-mail: rafi11@uwindsor.ca

Thamidul Islam Tonmoy
E-mail: ttonm001@ucr.edu

Uday Kamal
E-mail: udday2014@gmail.com

Q. M. Jonathan Wu
E-mail: jwu@uwindsor.ca

*Md. Kamrul Hasan
E-mail: khasan@eee.buet.ac.bd

[1]Department of Electrical and Computer Engineering, University of Windsor, Windsor, Canada · [2]Department of Bioengineering, University of California, Riverside, USA · [3]Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, Bangladesh.
*Corresponding author.

1 Introduction

Camera model identification (CMI) has gained significant momentum in recent years for information forensics as digitally altered images are becoming more pervasive in electronic media [1]. The increased usage of digital images in our everyday-life for entertainment, social networking, and more importantly in legal and security issues is, therefore, raising authenticity concern regarding the source of an image and its content, especially when presented to a court as an evidence [2]. Furthermore, the available professional image editing tools, though intended for entertainment purposes, are also facilitating image manipulation for illegal acts, making the problem of CMI even crucial. Although the metadata of an image contains some information about the source, it is not a reliable metric to determine the source since this data can be forged [1]. Besides, the metadata of the digital images are mostly

unavailable when shared online in social media. Moreover, while sharing online, images go through various post-processing operations which destroy the trace of the source information to some extent, making the identification even more difficult [3]. As a result, the task of identifying the camera model is continually becoming more challenging. Therefore, a forensic analyst has to resort to image processing and analysis techniques to identify the camera model with which an image was taken.

A number of methods have been proposed in the literature for blind identification of a camera model. An extensive review of these methods can be found in [1, 4]. Initially, researchers have tried to merge external features, e.g., watermarks, device-specific-code, etc., present in an image for device identification [5]. However, adding different extrinsic features to every single camera being used has proved to be an unmanageable task [6]. As a result, the focus has shifted towards detecting intrinsic camera features, such as the color filter array (CFA) pattern [7], interpolation algorithms and image quality metrics (IQM) [8, 9]. Device-specific camera detection schemes have also been proposed, where noise patterns like the photo response non-uniformity (PRNU) have been exploited to identify the device [10–12]. At the same time, forensic researchers have developed device invariant CMI algorithms [13, 14]. Most of these methods attempt to estimate the model-specific artifacts that are introduced into an image during the image capturing process [15]. In this approach, the second-order statistics of the CFA pattern [16] and 3D co-occurrence matrices [17, 18] have been used as feature vectors to successfully detect camera models with state of the art accuracy.

Recently, researchers have adopted data-driven approaches and made efforts to solve the CMI problem using convolutional neural networks (CNNs). A common practice while using convolutional neural networks (CNNs) in digital image forensics is to perform some preprocessing on the input images to refrain the network from learning features related to the image content, at the same time, associate them to learn the camera models specific contents. Conventional median or high-pass filter has been used in some works prior to feeding images into CNNs [19, 20]. However, the reason behind using these fixed kernel coefficients or a particular kernel size is not well explained in those works, thereby requiring human intervention in designing these filters. What is more, these filters may not generalize on different datasets. In addition, as mentioned in [14], the signatures left by different components of the image acquisition pipeline have different frequency ranges because demosaicing and vignetting leave low-frequency patterns whereas the SPN introduces high-frequency components. Therefore, using a fixed high-pass filter may result in a loss of valuable camera model specific features. Similarly, a specific kernel size for median filtering may not serve the purpose optimally. To overcome such difficulties with the conventional fixed filters, Bayar and Stamm [21] have proposed a data-driven constrained convolutional layer which is shown to be superior in performance to both median and high-pass filters. However, the constrained convolution, originally proposed for image manipulation detection in [22], that extracts prediction error features in the preprocessing stage does not explain how these features retain camera model-specific features. Nevertheless, the improvement of results associated with the different preprocessing schemes has made it very clear that a customized preprocessing operation should be explored thoroughly in this field.

More recently, another category of works has emerged that does not employ any preprocessing stage that has been shown to facilitate feature extraction for CMI [23–26]. In [23], a concept of fusion residual network (FRN) is proposed which uses the idea of using multi-scale receptive fields on an input image. The FRN extracts intrinsic features from the input image through convolutions with different kernel sizes and then concatenates the extracted feature maps. Bondi et al. [24] have proposed a combination of CNN and support vector machine (SVM) to classify camera models, where they have used CNN to extract camera-specific artifacts. On the other hand, Yao et al. [25] have proposed a comparatively deeper CNN architecture for CMI. Although these approaches show promising performances, none of the authors have investigated if the performance of their networks can be ameliorated with the incorporation of dynamic preprocessing filters. In [26], the authors explore the performance of DenseNet [27] using three different image scales (64 X 64, 128 X 128, 256 X 256), which they find beneficial to the CNN model.

Despite the breadth of works performed in this field, little attention has been given to the identification of camera models from images of unseen devices– the devices whose images have not been used in training the neural networks. Kirchner and Gloe have emphasized on this issue by proposing an evaluation criterion that uses disjoint subsets of devices for training and testing CMI methods to replicate real-world scenarios [4]. Moreover, the performance of the existing CMI methods on post-processed images, e.g., JPEG compressed, resized, gamma-corrected, etc., is not well studied. Although some researchers have ex-

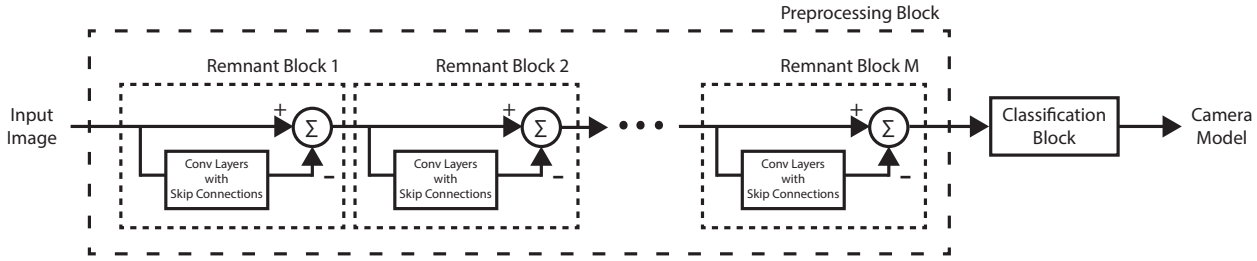

Fig. 1: Block diagram of our proposed RemNet.

plored the case of detecting image manipulation discretely [21, 22, 28–30], not many have tried to identify source camera model from post-processed images. In reality, a robust CMI network needs to correctly predict the source camera model of an image that may have gone through diverse *post-processing* and captured by an *unseen* device.

In this work, we propose RemNet, a novel CNN-based architecture to perform CMI task on extensively post-processed images acquired from unseen devices. The major constituent part of RemNet is the data-adaptive preprocessor that comprises of several remnant blocks. Unlike the conventional fixed-filter based approaches, our preprocessor can dynamically adapt its parameters to perform required preprocessing task suitable for the subsequent classification block. We also adopt a modular structure in our architecture which enables any CNN-based classifier to be cascaded with our proposed preprocessor. Our extensive experimentation shows that RemNet not only surpasses the state of the art CMI networks but also enhances their performance if used in cascade with our proposed preprocessor.

The rest of the paper is organized as follows. Section 2 presents a detailed description of our proposed method, along with the motivations and intuitions behind designing it. Section 3 provides a thorough discussion of the training and evaluation procedure, along with the experimental results obtained after testing the model with different datasets. Finally, we conclude in Section 4.

2 Proposed CNN Model

In designing CNNs for image forensic tasks, it has been a common practice to use a preprocessing scheme to suppress the image contents and intensify the minute signatures induced by the image acquisition pipeline [19–21]. However, the methods reported so far suffer from their own drawbacks of using either fixed kernels or constraints as described earlier. The main objective of this work is, therefore, to introduce a preprocessing scheme that is completely data-driven but without any imposed constraints or fixed kernels. To this end, we design a novel CNN architecture called RemNet.

RemNet is comprised of two major building blocks– a data-driven preprocessing block used at the beginning of the network which is followed by a classification block (see Fig. 1). These blocks are trained end-to-end so that the preprocessing block acts as a data-driven custom preprocessing scheme on the input image that learns to suppress image contents to some extent as required for better minimization of the loss function and intensifies camera model-specific feature-rich portions of the image at its output. The details of our proposed network architecture are presented in the following.

2.1 Preprocessing Block

The preprocessing block consists of several remnant blocks. The detailed architecture of the remnant block is shown in Fig. 2(b). Each block consists of 3 convolutional layers with kernel size 3×3 followed by BN. Inside each block, the feature space is widened from $64 \times 64 \times 3$ to $64 \times 64 \times f_i$ in the first 2 convolutional layers and then reduced to $64 \times 64 \times 3$ again in the last convolutional layer. The choices for f_i in the consecutive remnant blocks are 64, 128, and 256, respectively. Finally, to generate the residue, the output of the final convolutional layer in a block is subtracted from the input in a pixel-wise manner. As the convolutional layers are followed by batch normalization (BN) layer, in spite of directly using the input, we use the batch normalized version of it. Our intuition behind such architectural choice is to enable a remnant block to learn the required transformation that would disintegrate the undesired contents so that the subsequent subtraction operation can suppress them and generate forensic feature enriched residue. But there is still a possibility of losing some important forensic information after

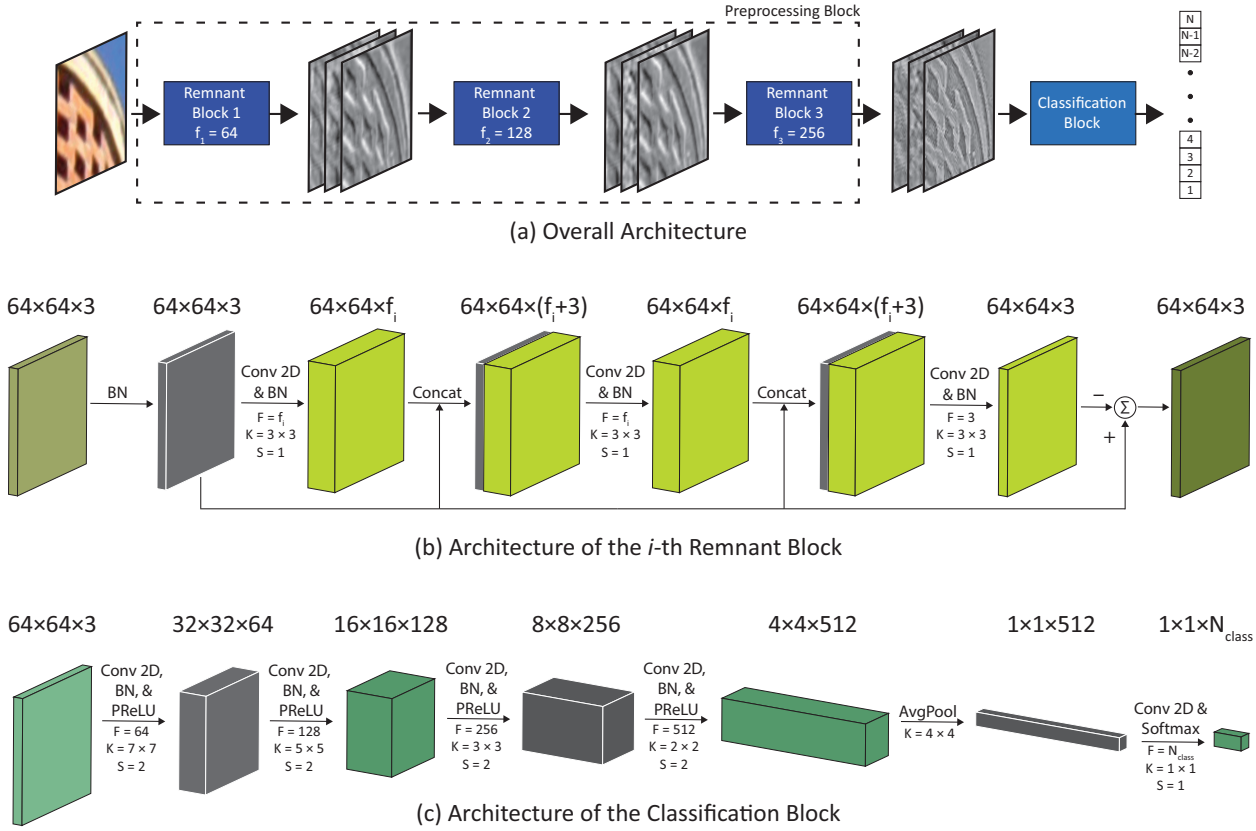

Fig. 2: The architecture of our proposed RemNet. (a) Illustrates the overall architecture with three remnant blocks with one classification block. The architectures of the remnant and classification blocks are depicted in (b) and (c), respectively. In (b) and (c), AvgPool, BN, and Conv2D represent average pooling, batch normalization, and 2D convolution, respectively. The letters F, K, and S represent the number of filters, their kernel sizes, and strides, respectively, in the corresponding convolution layers. The letter N_{class} represents the number of camera models.

such intermediate convolution operations. As the subsequent blocks operate on the residue generated by the previous block, such information loss would gradually build up, causing considerable degradation of the model's performance. The input information must be preserved as much as possible throughout the block to alleviate this problem. In order to ensure this, we include several skip connections so that the input to a remnant block is propagated to every convolutional layer inside that block. Even if some of the minute features are lost in a layer, it is regenerated through the skip connections (see Fig. 2(b)). This also prevents the vanishing of gradient-flow during training. We do not use any activation function in our remnant blocks because we prefer to build the remnant blocks as linear filters that will act as optimal preprocessors for CMI. The contribution of the remnant blocks is experimentally verified in our experimental results section (see Table 3).

There are several hyperparameter choices in the final structure of our preprocessing scheme: the number of remnant blocks, the depth of a single block, the number of filters in each layer, and kernel size– all of these are set using cross-validation.

The remnant blocks are somewhat influenced by the highway networks proposed by Srivastava et al. in [31]. A plain convolutional layer applies a linear transformation H (parameterized by $\mathbf{W_H}$) on its input $\mathbf{x}$ to produce its output $\mathbf{y}$:

$$\mathbf{y} = H\left(\mathbf{x}, \mathbf{W_H}\right), \tag{1}$$

where H is usually an affine transformation followed by a nonlinear activation function, but it may take different forms for different tasks.

For a highway network, two nonlinear transforms $T(\mathbf{x}, \mathbf{W_T})$ and $C(\mathbf{x}, \mathbf{W_C})$ are defined such that

$$\mathbf{y} = H\left(\mathbf{x}, \mathbf{W_H}\right) \cdot T\left(\mathbf{x}, \mathbf{W_T}\right) + \mathbf{x} \cdot C\left(\mathbf{x}, \mathbf{W_C}\right), \tag{2}$$

where T is the transform gate and C is the carry gate. T controls how much of the activation is passed through and C controls how much of the unmodified input is passed through. Our remnant blocks are motivated by

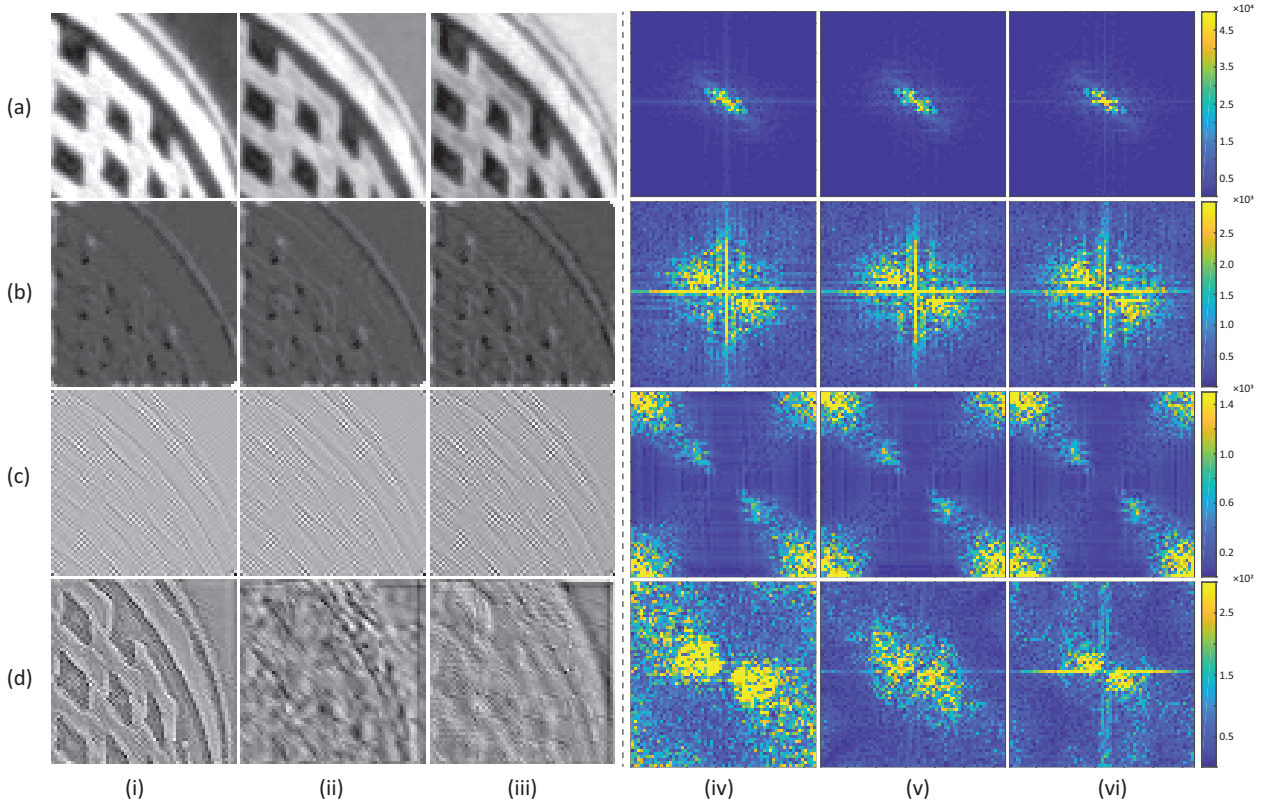

Fig. 3: Comparison of outputs of various preprocessing schemes. (a) Input image, (b) median filter residue, (c) high-pass filter output, and (d) output of the third remnant block of our proposed RemNet. Columns (i), (ii), and (iii) correspond to different output channels, whereas columns (iv), (v), and (vi) depict their frequency responses, respectively.

these two gating units. We make significant modifications in our transformation function H because of the nature of the operation we want to perform. As the remnant blocks are intended to be designed as a linear preprocessor, as stated before, we avoid the use of nonlinear activation functions. Also, we make use of multiple intra-block skip connections in our remnant block to preserve input information throughout a block. We use a pixel-wise subtraction operation that regulates the flow of information and alleviates the loss of information through successive convolutional operations. For the above-mentioned reasons, our transform and carry gate are linear in nature and we set T and C as -1 and 1, respectively. As a result, (2) becomes

$$\mathbf{y} = \mathbf{x} - H\left(\mathbf{x}, \mathbf{W_H}\right). \tag{3}$$

The residual network (ResNet) [32] is also a variant of the highway network [33] where the choices for both T and C are 1 for the residual blocks. However, the transformation H used in [32] works as a nonlinear feature extractor whereas the H of our remnant blocks performs linear filtering operation. In addition, ResNet does not use any skip connections.

To demonstrate that the dynamically designed remnant blocks truly performs the desired preprocessing task, we show in Fig. 3 the outputs of the final remnant block along with their frequency characteristics for a randomly selected image. We also make a spatial and frequency domain comparison of the conventional filters, e.g., median and high-pass filters used in [19, 20], respectively. Fig. 3(a) shows the RGB image, Figs. 3(b)-3(d) show the median filtered residue, high-pass filtered output, and the output of the last remnant block, respectively. If we observe the frequency domain representation of the outputs, we notice that conventional fixed filters are constrained in the frequency domain as compared to our remnant blocks since the conventional filters apply the same frequency domain transformation on all the channels equally. However, it is well known that the sensor pattern noise is not uniformly distributed throughout all three channels [34], and Lukas et al. [14] have explicitly stated that both low and high-frequency information are required for CMI. We, therefore, claim that our data-adaptive preprocessing performs better filtering operation, pre-

serving the camera signature from a wide range of frequencies, which is empirically justified by our experimental results presented in Section 3.

2.2 Classification Block

The output of the final remnant block, of size $64 \times 64 \times 3$, is passed to a classification block which is outlined in Table 1. The aim of this module is to extract higher-level camera model-specific features, reduce the dimensions of the feature vectors, and eventually generate a class probability of the source camera model of the input image. The classification block is trained end-to-end with the remnant blocks. Therefore, it forces the remnant blocks to suppress unnecessary contents, enhance the useful ones, and then generate a remnant of the image which contains rich camera model fingerprints for better minimization of the classification loss function.

The classification block has four consecutive convolution layers at the beginning. Each of the convolutional layers is followed by a BN layer and a PReLU activation. The output of the fourth convolutional layer, of size $4 \times 4 \times 512$, is followed by an average-pooling operation, which reduces the feature vector to a size of $1 \times 1 \times 512$. Finally, we pass the average-pooled feature vector to a final convolution layer with softmax activation to generate probabilities for the N_{class} number of camera models.

Instead of using max-pool operation, we use strided convolution to reduce the feature space in the first four convolution layers. This makes the feature reduction process learnable and much less aggressive compared to max-pool [35]. As per the design principles introduced in [21], we gradually decrease the kernel size in the first convolution layers. The BN layer is included for regularization and faster convergence.

Previously CNNs used the ReLU as the activation function [36]. But here we want to emphasize on extracting camera model fingerprints which are statistical in nature. They do not necessarily have to be positive. As we do not want to put any constraint on the feature generation, we use the PReLU activation function in our classification block. Also, when CNNs used with a PReLU activation function, it has experimentally demonstrated higher accuracy [22]. We have also experimentally verified this in our experimental results section (see Table 3).

The average-pool operation is used as per the conventional design structure of CNNs [27, 32, 37] to reduce the dimensionality of the feature space before making the final decision. We do not use fully connected layers in the classification block to keep the number of parameters lower, which in turn makes the network less prone to overfitting. This also helps the network to train faster.

Table 1: Architecture of our proposed RemNet

Layers	Output Size	Kernels*
Preprocessing Block		
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	$1 \times 1 \times N_{class}$	F = N_{class}, K = 1×1, S = 1
Softmax	N_{class}	–

* The letters F, K, and S represent the number of filters, their kernel size, and strides, respectively, in the corresponding convolution layers. The letter N_{class} represents the number of camera models.

2.3 Loss Function and Training

The preprocessing block consists of M remnant blocks. The i-th remnant block performs a transformation H_i (parameterized by $\mathbf{W_{p_i}}$) on its input $\mathbf{x_i}$ (that is the output of the $(i-1)$-th remnant block) and subtracts it from its input to produce its output $\mathbf{y_{p_i}}$:

$$\mathbf{y_{p_i}} = \mathbf{x_i} - H\left(\mathbf{x_i}, \mathbf{W_{p_i}}\right), \tag{4}$$

The output of the last remnant block, $\mathbf{y_{p_M}}$, then becomes the input of the classifier block that applies another transformation G (parameterized by $\mathbf{W_c}$) to produce the final output $\mathbf{y_c}$:

$$\mathbf{y_c} = G\left(\mathbf{y_{p_M}}, \mathbf{W_c}\right), \tag{5}$$

Finally, multiclass categorical crossentropy loss is calculated based on this output and the ground truth using the following equation:

$$L = \sum_{k=1}^{N_{class}} y_{c_i}^{*(k)} \log\left(y_{c_i}^{(k)}\right), \tag{6}$$

where $y_{c_i}^{*(k)}$ and $y_{c_i}^{(k)}$ are the true label and the network output of the i-th image at the k-th class among the N_{class} classes, respectively,. The gradient of this loss is backpropagated to update the weights of both the preprocessing block and the classifier block of the

network. Since the preprocessing block generates a residue of the input signal and the subsequent classifier will have to extract useful features from this residue alone as well as the whole network being trained in an end to end manner, the minimization of the loss function ensures that the preprocessing block learns to suppress the image contents that are irrelevant for CMI and the residue generated by it contains rich camera fingerprints.

3 Experimental Results

To demonstrate the effectiveness of the RemNet and the remnant blocks separately, we conduct a number of experiments. In this section, we discuss those experimental results in detail. All of the experiments regarding training and implementation of the model are performed in a hardware environment which includes Intel Core-i7 8700K, 3.70 GHz CPUs and Nvidia GeForce GTX 1080 Ti (11 GB Memory) GPU. The necessary codes are written in Python and the neural network models are implemented using the Keras API (version 2.1.6) with TensorFlow-GPU (version 1.8.0) in the backend.

3.1 Results on Dresden Dataset

We comprehensively evaluate our overall approach on the Dresden dataset [38]. These images are captured with 73 devices of 27 different camera models. Multiple shots have been taken from several locations (e.g., office, public square, etc.) for each device. Different pictures are acquired from different viewpoints (e.g., looking on the right, on the left, etc.) for each location. We refer to different combinations of locations and viewpoints as different *scenes*. The acquisition process is explained in detail in [38]. In our work, we choose only those camera models which have more than one device so that we can keep one device separate for testing purpose. This results in discarding 8 camera models. Of the rest 19 devices, we consider two camera models, Nikon D70 and Nikon D70s, as a single model based on the work of Kirchner and Gloe. [4]. Consequently, we train and test our models using the images of these 18 camera models. A brief description of the dataset used is presented in Table 2.

3.1.1 Training and testing strategy

Training a CMI network is challenging because of the existence of device-specific features such as PRNU noise [11, 14] along with model-specific features in the image. Therefore, a network that can detect the model-specific features needs to be trained in such a way that it excludes the device-specific features as much as possible and is able to focus on the model-specific features. We solve this problem by using images from multiple devices to train our network for most camera models.

We first split the dataset into train, validation, and test sets in such a way that the camera device and scenes used during testing are never used for training or validation. This results in 7938, 1353 and 540 images in the train, validation and test set, respectively (see Table 2). We refer to these sets as *unaltered* train, validation, and test sets. This splitting policy, proposed in [24], is of paramount importance so that we can be sure that the neural network does not overfit on the training data and the testing accuracy is not biased by device-specific features or the natural content of the scenes.

After splitting the dataset, we extract 256×256 sized clusters of pixels from the original images. However, it is to be noted that all clusters of pixels from an image are not rich in camera model-specific features. In particular, saturated and flat regions are not likely to

Table 2: Camera models of the Dresden database used in our experiments

Serial No.	Camera Model	No. of Images	No. of Devices	
			Train and Val.	Test
1	Canon IXUS 70	363	2	1
2	Casio EX-Z150	692	4	1
3	FujiFilm FinePix J50	385	2	1
4	Kodak M1063	1698	4	1
5	Nikon Coolpix S710	695	4	1
6	Nikon D200	373	1	1
7	Nikon D70	373	1	1
	Nikon D70S		1	1
8	Olympus μ1050SW	782	4	1
9	Panasonic DMC-FZ50	564	2	1
10	Pentax Optio A40	405	3	1
11	Praktica DCZ 5.9	766	4	1
12	Ricoh Capilo GX100	559	4	1
13	Rollei RCP-7325XS	377	2	1
14	Samsung L74wide	441	2	1
15	Samsung NV15	412	2	1
16	Sony DSC-H50	253	1	1
17	Sony DSC-T77	492	3	1
18	Sony DSC-W170	201	1	1
	Total	9831		

contain enough statistical information about the camera model [24]. Therefore, different authors have used different cluster selection strategies in the literature. In [23], the authors propose a new metric to classify the image clusters into three categories: i) Smooth, ii) Saturated and iii) Others. After that, they train their network on these three categories separately and get three different networks (same architecture but different weights) on which they report the performance results for the respective categories of image clusters. On the other hand, in [24], the authors propose a metric that gives a higher score to the image cluster with more texture, and train and test their network with these high-scoring clusters only. Since our target is to propose a single CMI network for solving the task, we need to train and test it with clusters that contain enough statistical information about the camera model. That is why we compute the quality value of a cluster as outlined in [24]. For each cluster $\mathcal{P}$ in an image, its quality $Q(\mathcal{P})$ is computed as

$$Q(\mathcal{P}) = \frac{1}{3} \sum_{c \in [R,G,B]} \left[\alpha \cdot \beta \cdot (\mu_c - \mu_c^2) + (1 - \alpha) \cdot (1 - e^{\gamma \sigma_c}) \right] \tag{7}$$

where α, β, and γ are empirically set constants (set to 0.7, 4 and ln(0.01), respectively), μ_c and σ_c, $c \in [R, G, B]$ are the mean and standard deviation of the red, green, and blue components of cluster $\mathcal{P}$, respectively. For a cluster of pixels with texture, this quality measure tends to be higher than for the overly saturated or flat clusters (see Fig. 4). We found that this quality assessment is consistent with the '*others*' category mentioned in [23]. According to the definition in [23], 99.32% of our high-quality clusters fall into *others* category while 0.63% are *smooth*, and the rest 0.03% are *saturated*. Therefore, we can consider that our cluster selection strategy is almost identical to choosing the 'others' category patches of Yang et al. [23].

Although we extract 256×256 sized rich quality clusters from the main image, the input patch size that we opt to use for our network is 64×64, as suggested in [23–25]. During training, we select a patch of size 64×64 randomly from a cluster of 256×256 in each epoch. The idea of small input patch of 64×64 is motivated by three reasons: (i) it results in more data to train our proposed network; (ii) during the test, it enables us to generate multiple predictions for a given image and averaging over all of those predictions may ensure a more accurate classification; (iii) training our network with patches of smaller size relative to the image prevents our network from learning dominant spatial features of the image affixed directly to its contents, subsequently enabling the network to learn inherent model-specific statistical features. Also, training a network with bigger input patch size poses hardware constraints and requires more training time.

Our cluster and patch selection strategy introduces statistical variations during training. The network cannot rely on seeing the same patch of size 64×64 more than once since they are randomly extracted from the 256×256 clusters in each epoch. This has a regularizing effect and forces the network to learn more robust features that generalize better across multiple samples of the input data. Our proposed cluster selection method also ensures that the input patches of 64×64 to the network are a mix of good and bad patches where good patches are dominant in number. Some of the rich quality clusters of 256×256 may contain a few bad patches of 64×64 as illustrated in Fig. 4. Therefore, during training, the network learns to extract features from saturated regions as well. This, in turn, helps our network perform well in poor quality clusters extracted from the main image, which is demonstrated in the experimental results.

During training, we extract 20 rich quality clusters of size 256×256 from each image which results in 158760 and 27060 clusters for the unaltered train and validation set, respectively. We then randomly crop a 64×64 size patch from each cluster in each epoch and feed it to the network. Since we are experimenting with 18 camera models, we set $N_{class} = 18$ for our classification block. The weights of the network kernels are initialized randomly with the uniform distribution proposed by Glorot and Bengio [39]. We use categorical cross-entropy as the loss function and Adam [40] as the optimizer with the exponential decay rate factors $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size we opt to use is 64. The initial learning rate is set to 10^{-3} and is

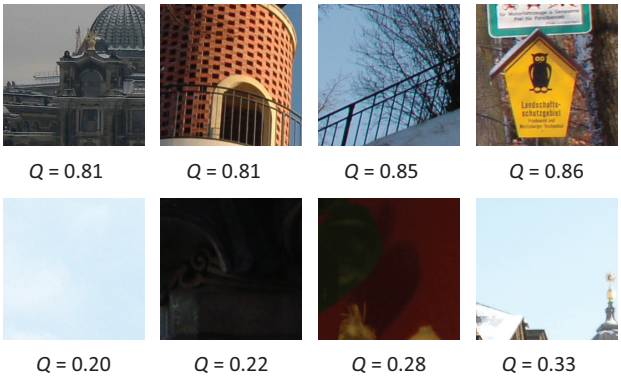

Fig. 4: Examples of clusters of different qualities with their quality indices. The top row represents rich quality clusters and the bottom row represents poor quality clusters.

decreased by a factor of 0.5 if the validation loss does not decrease in two successive epochs. When the learning rate is reduced to 10^{-7}, the training is stopped. In this way, we train our network for a maximum of 50 epochs and save the weight with the least validation loss for evaluation.

After training, we test our network on the unaltered test set comprised of 540 images from unseen devices of 18 different camera models of the Dresden database. During testing, we select N number of rich quality clusters of size 256×256 from each test image according to our quality assessment. To make a prediction for each cluster, we take the average of the predictions on all non-overlapping patches of size 64×64 it contains and assigns a camera model label $\hat{l_j}$ to it. The final prediction for the image is obtained through majority voting on $\hat{l_n}$ for $n \in [1, N]$. In all the subsequent experiments, we use $N = 20$ unless otherwise stated. Finally, the accuracy of the network is obtained using the following equation:

$$\text{Accuracy} = \frac{N_{corr}}{N_{tot}} \times 100\%, \tag{8}$$

where N_{corr} is the number of images correctly predicted and N_{tot} is the total number of images, which in this case, is 540.

3.1.2 Comparison of design choices

First, we experiment with several architectural design choices of our proposed RemNet. We train and test these various designs on the unaltered dataset. The results of these experiments are presented in Table 3. It is evident from the table that our proposed RemNet with 3 remnant blocks followed by a classification block with PReLU activation results in a better accuracy. The detection accuracy it achieves is 97.03%.

3.1.3 Comparison with state of the art networks on unaltered images

We compare our results with the established methods in CMI– constrained-convolutional network [21], fusion residual network [23] and first steps toward the camera model identification with convolutional neural networks [24]. The reason behind choosing [21] and [23] is that both of these works incorporate their own preprocessing scheme that agrees to our main intuition in this work. Since our rich quality clusters commensurate with the 'others' category of [23], we implement the fusion residual network for the 'others' category only, instead of each of the three different categories mentioned in [23]. We also include [24] in our comparison as we adopt their cluster selection strategy. Recently, several works such as [41], [42], and [43] confirm the superior performance of very deep neural networks in different camera forensic applications. As a result, we also compare the performance of the RemNet with two CNN based deeper architectures namely ResNet [32] and DenseNet [44]. For a fair comparison, we use the same input patch size, 64×64, for all the networks and the implementation of each method is made under careful scrutiny.

Table 3: Accuracy of different design choices of RemNet trained and tested on the unaltered train and test sets of the Dresden database

Design Choice	Accuracy (%)
Remnant Blocks + Classifier (ReLU)	96.48
Remnant Blocks with Activation (PReLU) + Classifier (PReLU)	96.67
Remnant Blocks + Classifier (PReLU)	**97.03**

The results presented in Table 4 show that networks with preprocessing schemes perform substantially better than the other networks and our proposed RemNet outperforms all the networks with a significant margin. This observation, therefore, establishes our claim that a preprocessor is indeed necessary in CMI even for deeper architectures.

3.1.4 Effects of Data Augmentation

Deep neural networks have a tendency to overfit due to their large number of learnable parameters. Since these methods require a large amount of data to avoid overfitting, data augmentation is a commonly used method in training CNNs [45]. Also, our goal is to design a robust network that can perform CMI even if the image is post-processed. To address these challenges, we use different types of post-processing methods as a form of data augmentation to increase the volume of

Table 4: Accuracy of different methods trained and tested on the unaltered train and test sets of the Dresden database

Method	Accuracy (%)
Bayar and Stamm [21]	95.56
Yang et al. [23]	94.81
Bondi et al. [24]	90.55
ResNet [32]	92.40
DenseNet [44]	93.33
Proposed Method	**97.03**

training data. The types of augmentation that we use in this work are:

- JPEG-Compression with quality factor of 70%, 80%, and 90%
- Rescaling by a factor of 0.5, 0.8, 1.5, and 2.0
- Gamma-Correction using γ = 0.8 and 1.2

We perform the aforementioned post-processing methods on the train and validation sets which increase the volume of data by 9 folds. We refer to these increased datasets as *augmented* train and validation sets. The augmented datasets contain both unaltered and manipulated images.

After training on the augmented train set, evaluation is carried out on the unaltered test set. The results are presented in Table 5. If we compare the results of Table 5 with that of Table 4, we observe that these post-processing schemes, as a form of data-augmentation, indeed improve the performance of all the networks except that in [21]. Our proposed RemNet achieves the best accuracy of 97.59% among all the models . It is worthwhile to mention that RemNet attains 100% accuracy on identifying 16 camera models, as shown in the corresponding confusion matrix in Fig. 5. For the rest of the two camera models, Sony DSC-H50 and Sony DSC-W170, RemNet attains accuracy of 90% and 75%, respectively. The decrease in the identification accuracy for these two exact models has also been observed in [20]. As mentioned in [4], images captured with camera models of the same manufacturer are likely to share some components which makes it harder to separate them.

Table 5: Accuracy of different methods trained on the augmented train set and tested on the unaltered test set of the Dresden database

Method	Accuracy (%)
Bayar and Stamm [21]	93.89
Yang et al. [23]	95.19
Bondi et al. [24]	92.59
ResNet [32]	95.18
DenseNet [44]	95.05
Proposed Method	**97.59**

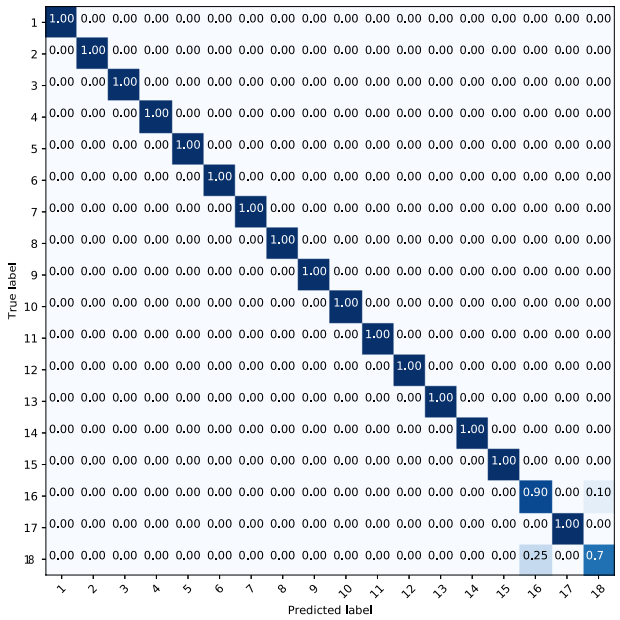

Fig. 5: Confusion Matrix of our proposed RemNet trained on the augmented train set and tested on the unaltered test set of the Dresden database. The input and predicted label correspond to the Serial No. used in Table 2.

In Fig. 6, we observe the effect of the voting number, the number of clusters on which the prediction is made during testing, on the performance of different networks. For the rich quality clusters (see Fig. 6(a)), our network shows a somewhat steady trend, whereas the other networks show oscillatory behavior. This indicates that the performance of our network is nearly independent of the voting number of clusters, whereas an optimum voting number has to be selected for other networks. On the other hand, for prediction on poor quality clusters of an image, the accuracy gradually increases with the increment of voting number for all of the networks, as is evident from Fig. 6(b). In both of these two cases, our proposed RemNet outperforms the other networks in comparison.

To further ensure that the networks are not biased toward the augmented train set, we perform post-processing on test images with such factors that are not necessarily used in the augmented train and validation set. We process the test images using gamma correction with γ = 0.5, 0.75, 1.25, and 1.5; JPEG compression quality factors (QFs) 95%, 90%, 85%, and 80%; and rescaling factor of 0.8, 0.9, 1.1, 1.2. The results of this study are presented in Table 6. These results show that our proposed RemNet outperforms the other networks with a significant margin in gamma correction. In rescale, the deeper models, specially ResNet [32], perform substantially better than all other networks. In JPEG compression, ResNet [32] and our proposed RemNet both achieve better performances in totality.

3.1.5 Significance of the Remnant Blocks

In order to validate the significance of our proposed preprocessor, we train and test our proposed classi-

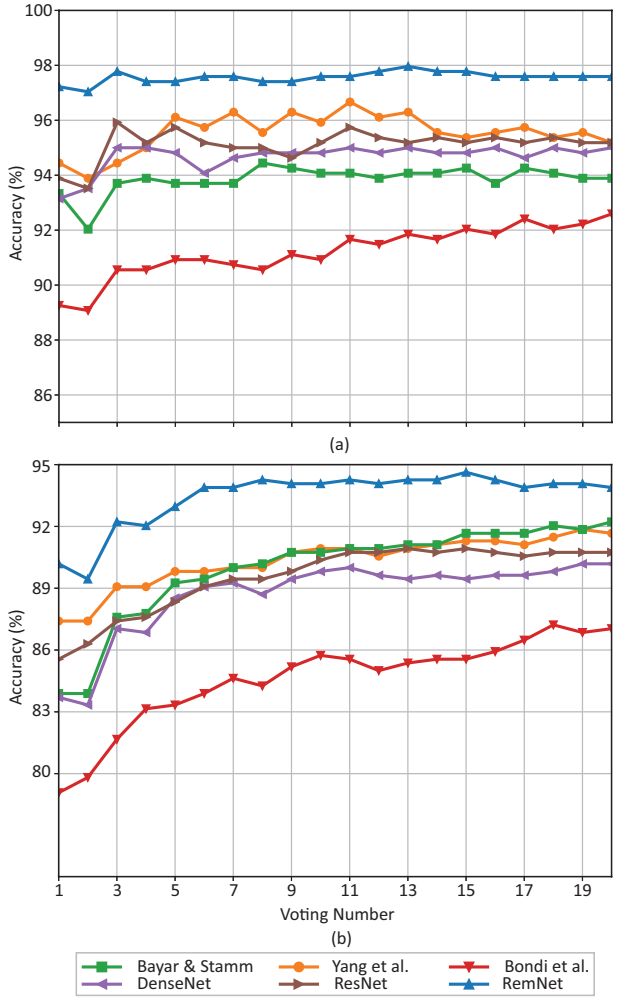

Fig. 6: Results of varying voting number for (a) rich quality clusters and (b) poor quality clusters of different methods, trained on the augmented train set, for testing with the unaltered test set of the Dresden database.

fier network without the remnant blocks. We also train and test the network proposed in [24], ResNet [32], and DenseNet [44] together with the remnant blocks to demonstrate its generalizability to any classification network and its positive impact on their performances. All these networks are trained end-to-end on the Dresden database. It is to be mentioned that we do not perform similar experiments on [21] and [23] since these networks already consist of their own preprocessing schemes.

The training histories of the models are presented in Fig. 7. As we can see, the addition of the remnant blocks not only improve the performances but also helps the models converge faster. The credit for these improvements can be attributed to the remnant blocks. When raw input images are fed directly to these classification networks, they are required to perform two tasks at the same time that is, to suppress the image content and to extract the required camera model fingerprints. Our proposed preprocessing scheme makes the later task easier as it suppresses the unnecessary content of the image and provides the classification block with inputs which are rich in camera model-specific features. Therefore, it becomes easier for these classification networks to identify camera models and update their weights faster during training compared to when they are trained with raw input images.

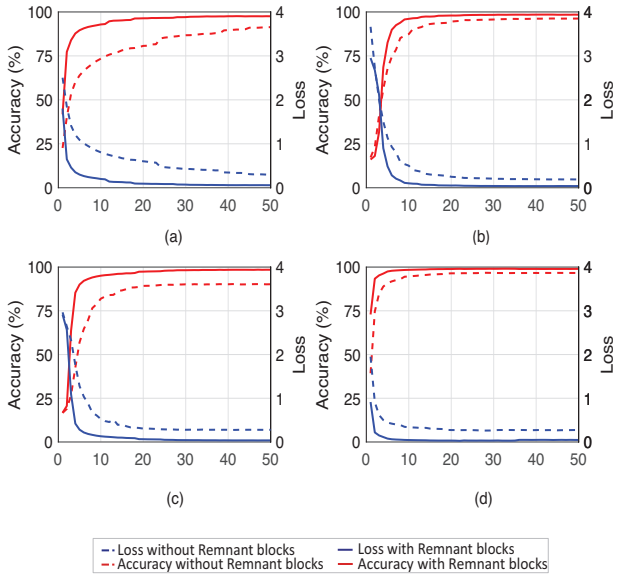

Fig. 7: Training history of (a) Bondi et al., (b) DenseNet, (c) ResNet, and (d) Our Proposed Classifier, with and without remnant blocks, for training with the augmented train set of the Dresden database.

From the experimental results presented in Table 7, it is clearly evident that our proposed preprocessing scheme improves the performance of all the aforementioned methods with a significant margin. The addition of our remnant blocks in cascade with these models helps them achieve substantially better performance even when they are trained with unaltered images only. Their performances further improve when they are trained with augmented data.

Also, in order to verify the effect of remnant blocks on the robustness of the networks trained with the augmented dataset, we further evaluate the performance of [24], ResNet [32], and DenseNet [44] with remnant blocks on the manipulated test dataset. The experimental results shown in Table 8 demonstrate that with the addition of the remnant blocks, all three models have a performance gain in most of the cases and also in totality. Also, due to the adaptive nature

Table 6: Comparative results of our proposed network with different methods, trained on the augmented train set, in identifying camera models from manipulated test images of the Dresden database (Accuracy in %)

Manipulation	Gamma Correction				JPEG Compression				Rescale			
Factor	0.5	0.75	1.25	1.5	95	90	85	80	0.8	0.9	1.1	1.2
Bayar and Stamm [21]	93.52	94.44	94.44	94.63	92.59	94.81	88.15	85.74	88.15	87.04	64.44	59.07
Yang et al. [23]	94.26	95.37	95.00	92.78	94.07	94.07	92.59	92.59	94.26	92.59	90.93	90.56
Bondi et al. [24]	85.92	91.85	89.07	92.03	84.07	85.92	91.48	90.74	92.56	92.77	91.48	89.44
ResNet. [32]	91.85	95.18	92.77	94.81	93.88	94.82	95.55	95.00	95.18	95.18	95.00	95.18
DenseNet. [44]	91.66	95.18	92.03	94.62	92.77	92.96	94.26	94.81	95.00	94.81	94.44	94.26
Proposed Method	96.11	97.22	96.11	95.56	97.59	94.81	92.59	92.78	95.00	93.33	92.04	92.41

Table 7: Results of different models, with and without remnant blocks, tested on the unaltered test set of the Dresden dataset (Accuracy in %)

Method	Trained on Unaltered Train Set		Trained on Augmented Train Set	
	without remnant blocks	with remnant blocks	without remnant blocks	with remnant blocks
Bondi et al. [24]	90.55	95.92	92.59	96.29
ResNet [32]	92.40	96.85	95.18	98.33
DenseNet [44]	93.33	96.29	95.01	98.14
Proposed Classifier	93.31	97.03	95.74	97.59

Table 8: Comparative results of different models with and without remnant blocks, trained on the augmented train set, in identifying camera models from manipulated test images of the Dresden database (Accuracy in %)

Manipulation	Gamma Correction				JPEG Compression				Resize Scale			
Factor	0.5	0.75	1.25	1.5	95	90	85	80	0.8	0.9	1.1	1.2
Bondi et al. [24]	85.92	91.85	89.07	92.03	84.07	85.92	91.48	90.74	92.56	92.77	91.48	89.44
Remnant-Bondi et al.	94.07	95.74	95.37	95.92	88.88	89.07	93.52	92.22	91.66	91.85	90.00	88.14
ResNet. [32]	91.85	95.18	92.77	94.81	93.88	94.82	95.55	95.00	95.18	95.18	95.00	95.18
Remnant-ResNet	98.33	98.33	97.59	97.59	93.33	93.33	95.18	95.92	95.37	95.18	92.40	95.00
DenseNet. [44]	91.66	95.18	92.03	94.62	92.77	92.96	94.26	94.81	95.00	94.81	94.44	94.26
Remnant-DenseNet.	96.85	97.59	97.96	97.59	93.70	93.88	94.81	95.92	95.37	94.81	93.52	95.18

of our preprocessing scheme and end-to-end training, the remnant blocks can learn to produce the optimum output as required by the subsequent classifier block. Such adaptive nature of our preprocessing scheme makes it a promising approach to further improve the CMI performance of the existing DNN based approaches without changing their configuration.

3.2 Results on the IEEE Signal Processing Cup 2018 Dataset

To test the generalizability of our approach, we have also trained and tested the aforementioned networks on the CMI Dataset provided for the IEEE Signal Processing (SP) Cup 2018 [46]. The training dataset provided by the IEEE Signal Processing Society consists of images captured by 10 different camera models having 275 images for each model. Since only one device is used to capture these images for each camera model, we collect external data from multiple devices from Flickr under the creative commons license. All these images are used for training and validation purposes only. A brief summary of the dataset is given in Table 9.

Table 9: IEEE SP Cup 2018 data and Flickr data

Camera Model	No. of Images	
	SP Cup Data	Flickr Data
HTC-1-M7	275	746
iPhone-4s	275	499
iPhone-6	275	548
LG-Nexus-5x	275	405
Motorola-Droid-Maxx	275	549
Motorola-Nexus-6	275	650
Motorola-X	275	344
Samsung-Note3	275	274
Samsung-Galaxy-S4	275	1137
Sony-NEX-7	275	557
Sub-Total	2750	5709
Grand-Total	8459	

The dataset described in Table 9 is split into train and validation data by a 3:1 ratio. The test dataset is provided separately, which includes 2640 images of size 512 × 512, among which 1320 are unaltered, and the rest are augmented, i.e., resized, gamma-corrected, or JPEG compressed. All the test images are acquired with a separate device other than the ones used for capturing training and validation images.

The training and testing is done by following the same procedures as mentioned in the earlier experiments. This time, we train our network for 10 classes. The testing is done on the test set which contains images from completely separate devices that are used for training. Since the size of the test images is 512 × 512, we extract the best clusters of size 256 × 256 and generate result following the testing procedure mentioned previously. According to the competition rules of IEEE SP Cup 2018, the score on the test-results are calculated based on the following formula:

$$\begin{aligned}\text{Accuracy} = 0.7 \times (\text{Accuracy of Unaltered Images}) + \\ 0.3 \times (\text{Accuracy of Manipulated Images})\end{aligned} \tag{9}$$

Table 10 summarizes the result of our model on the SP cup dataset along with comparing it with different networks. From the table, it is clear that our proposed RemNet outperforms the other networks with an accuracy of 95.11%. This satisfactory performance is evidence of the generalizability of our approach. Among the other networks, wider ([23]) and deeper ([32, 44]) networks perform comparatively better than the shallower ones.

To verify the effect of remnant blocks on different networks for the IEEE SP Cup 2018 dataset, we train the networks [24], ResNet [32], and DenseNet [44] in cascade with remnant blocks. The experimental results are presented in Table 11. It is clear from the table that the addition of the remnant blocks improves the performances of the aforementioned networks. Therefore, our hypothesis that the remnant blocks can improve the performance of any classification network in CMI is further verified in different datasets.

Table 10: Accuracy of different methods on the IEEE SP Cup 2018 testing dataset

Method	Accuracy (%)
Bayar and Stamm [21]	90.97
Yang et al. [23]	94.83
Bondi et al. [24]	90.07
ResNet [32]	93.92
DenseNet [44]	93.70
Proposed Method	**95.11**

Table 11: Comparative results of different models, in cascade with remnant blocks, tested on the IEEE SP Cup 2018 testing dataset

Method	Accuracy (%)
Remnant-Bondi et al.	92.15
Remnant-ResNet	93.98
Remnant-DenseNet	94.68

3.3 Visualizing the Models Class Activation

Due to a large number of parameters, the CNNs can easily get biased to the image content, rather than the intrinsic camera fingerprint. It has been, therefore, a topic of great interest among the camera-forensic experts about what type of forensic features such deep models learn for CMI. To explore this, we adopt the class activation maximization method proposed by Erhan et al [47] at the highest level of feature representation of the networks, i.e., on the output neuron to understand what type of input patterns activate the final class. The main goal of such an experiment is to observe and explore the hidden patterns present in the image that the networks have learned to extract for CMI. Due to the paper size limit, we show the generated patterns for 3 different camera models for ResNet [32], DenseNet [44], and our proposed network in Fig. 8. From this figure, it is evident that deep networks

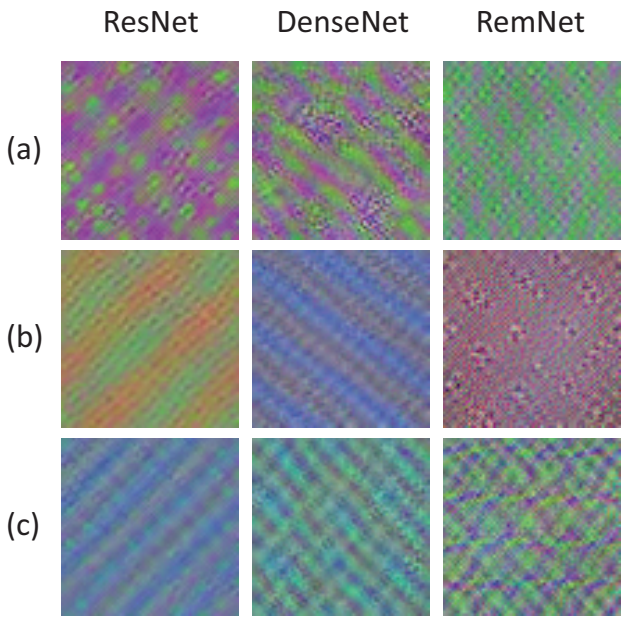

Fig. 8: Visualization of input activation of (a) Canon IXUS 70, (b) CanonEX-Z150, and (c) FujiFilm FinePix J50 for different networks trained on the Dresden database.

trained for CMI do not focus on the visible image content. The noticeable difference among the patterns of different networks can be explained by the fact that different network architecture can be thought of different transformation function to be applied to the same input, which on the other hand, may result in such a difference.

4 Conclusion

In this paper, a novel CNN model has been proposed for the identification of the source camera model of an image for digital image forensics. To address the CMI problem effectively, a dynamic CNN-based preprocessing block has been placed in cascade with the shallow CNN-based classifier for enhancing the intrinsic camera model-specific fingerprints at its output by suppressing the undesired contents of the input image. Unlike the conventional fixed filter-based approaches for preprocessing in image forensics, the remnant blocks of the proposed preprocessing unit are completely data-driven. The experimental results on the Dresden and the IEEE SP Cup 2018 Camera Model Identification datasets, focusing on the unseen devices of close-set camera models and post-processed images, have demonstrated improved performance and generalizability of the proposed modular RemNet for real-world CMI application. Furthermore, the demonstrated ability of the remnant blocks to improve the CMI performance along with the speed of convergence of the well-known CNN-based approaches indicates that they are suitable as a general-purpose preprocessing scheme for varieties of CMI networks. In future works, we wish to explore the potential of such a preprocessing scheme in other image forensic tasks such as forgery detection and post-processing classification.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interests.

References

1. M. C. Stamm, M. Wu, and K. R. Liu, "Information forensics: An overview of the first decade," *IEEE Access*, vol. 1, pp. 167–200, 2013.
2. K. San Choi, E. Y. Lam, and K. K. Wong, "Source camera identification by jpeg compression statistics for image forensics," in *TENCON IEEE Region*. IEEE, 2006, pp. 1–4.
3. A. Castiglione, G. Cattaneo, M. Cembalo, and U. F. Petrillo, "Experimentations with source camera identification and online social networks," *J. Amb. Intel. Hum. Comp*, vol. 4, no. 2, pp. 265–274, 2013.
4. M. Kirchner and T. Gloe, "Forensic camera model identification," *Proc. WOL Handbook of Digital Forensics of Multimedia Data and Devices*, pp. 329–374, 2015.
5. A. Piva, "An overview on image forensics," *Proc. ISRN Signal Process.*, vol. 2013, 2013.
6. H. Farid, "Image forgery detection," *IEEE Signal Process. Mag.*, vol. 26, no. 2, pp. 16–25, 2009.
7. S. Bayram, H. Sencar, N. Memon, and I. Avcibas, "Source camera identification based on cfa interpolation," in *Proc. IEEE Int. Conf. on Image Process., (ICIP)*, vol. 3. IEEE, 2005, pp. III–69.
8. M. Kharrazi, H. T. Sencar, and N. Memon, "Blind source camera identification," in *Proc. IEEE Int. Conf. on Image Process., (ICIP)*, vol. 1. IEEE, 2004, pp. 709–712.
9. T. Gloe, "Feature-based forensic camera model identification," in *LNCS Trans. Data Hiding and Multimed. Secur. VIII , Vol. 7228 of Lect. Notes Comput. Sc.* Springer, 2012, pp. 42–62.
10. A. E. Dirik, H. T. Sencar, and N. Memon, "Source camera identification based on sensor dust characteristics," in *Proc. IEEE Workshop Signal Process. Appl. Public Secur. Forensics*. IEEE, 2007, pp. 1–6.
11. J. Fridrich, J. Lukas, and M. Goljan, "Digital camera identification from sensor noise," *IEEE Trans. Inf. Forensics Secur.*, vol. 1, no. 2, pp. 205–214, 2006.
12. T. Filler, J. Fridrich, and M. Goljan, "Using sensor pattern noise for camera model identification," in *Proc. IEEE Int. Conf. on Image Process., (ICIP)*. IEEE, 2008, pp. 1296–1299.
13. T. H. Thai, R. Cogranne, and F. Retraint, "Camera model identification based on the heteroscedastic noise model," *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 250–263, 2014.
14. J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Trans. Inf. Forensics Secur.*, vol. 1, no. 2, pp. 205–214, 2006.
15. H. Cao and A. C. Kot, "Accurate detection of demosaicing regularity for digital image forensics," *IEEE Trans. Inf. Forensics Secur.*, vol. 4, no. 4, pp. 899–910, 2009.
16. A. Swaminathan, M. Wu, and K. R. Liu, "Nonintrusive component forensics of visual sensors using output images," *IEEE Trans. Inf. Forensics Secur.*, vol. 2, no. 1, pp. 91–106, 2007.
17. C. Chen and M. C. Stamm, "Camera model identification framework using an ensemble of demosaicing features," in *Proc. IEEE Int. Works. Infor. (WIFS)*. IEEE, 2015, pp. 1–6.
18. F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, "A study of co-occurrence based local features for camera model identification," *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 4765–4781, 2017.
19. J. Chen, X. Kang, Y. Liu, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1849–1853, 2015.
20. A. Tuama, F. Comby, and M. Chaumont, "Camera model identification with the use of deep convolutional neural networks," in *Proc. IEEE Int. Workshop on inf. Forensics and Secur. (WIFS)*. IEEE, 2016, pp. 1–6.
21. B. Bayar and M. C. Stamm, "Design principles of convolutional neural networks for multimedia forensics," *Electronic Imaging*, vol. 2017, no. 7, pp. 77–86, 2017.
22. ——, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proc. 4th-ACM Workshop on inf. Hiding and Multimedia Secur.* ACM, 2016, pp. 5–10.
23. P. Yang, W. Zhao, R. Ni, and Y. Zhao, "Source camera identification based on content-adaptive fusion network," *Pattern Recog. Lett.*, vol. 119, pp. 195–204, 2019.
24. L. Bondi, L. Baroffio, D. Güera, P. Bestagini, E. J. Delp, and S. Tubaro, "First steps toward camera model identification with convolutional neural networks," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 259–263, 2017.

25. H. Yao, T. Qiao, M. Xu, and N. Zheng, "Robust multi-classifier for camera model identification based on convolution neural network," *IEEE Access*, vol. 6, pp. 24 973–24 982, 2018.
26. A. M. Rafi, U. Kamal, R. Hoque, A. Abrar, S. Das, R. Laganière, and M. K. Hasan, "Application of densenet in camera model identification and post-processing detection." in *CVPR Workshops*, 2019, pp. 19–28.
27. G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, vol. 1, no. 2, 2017, p. 3.
28. B. Li, Y. Q. Shi, and J. Huang, "Detecting doubly compressed jpeg images by using mode based first digit features," in *Proc. IEEE 10th Workshop on Multimedia Signal Process.* IEEE, 2008, pp. 730–735.
29. M. C. Stamm and K. R. Liu, "Forensic detection of image manipulation using statistical intrinsic fingerprints," *IEEE Trans. Inf. Forensics Secur.*, vol. 5, no. 3, pp. 492–506, 2010.
30. E. Kee, M. K. Johnson, and H. Farid, "Digital image authentication from jpeg headers," *IEEE Trans. Inf. Forensics Secur.*, vol. 6, no. 3, pp. 1066–1075, 2011.
31. R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.
32. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2016, pp. 770–778.
33. K. Greff, R. K. Srivastava, and J. Schmidhuber, "Highway and residual networks learn unrolled iterative estimation," *arXiv preprint arXiv:1612.07771*, 2016.
34. Ashref, Lawgaly, Fouad, and Khelifi, "Sensor pattern noise estimation based on improved locally adaptive dct filtering and weighted averaging for source camera identification and verification," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, 2017.
35. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.
36. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. in Neural Inf. Process. Systems*, 2012, pp. 1097–1105.
37. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
38. T. Gloe and R. Böhme, "The dresden image database for benchmarking digital image forensics." *J. Digital Forensic Practice*, vol. 3, pp. 150–159, 01 2010.
39. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010, pp. 249–256.
40. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
41. Y. Chen, X. Kang, Z. J. Wang, and Q. Zhang, "Densely connected convolutional neural network for multi-purpose image forensics under anti-forensic attacks," in *Proc. 6th ACM Workshop Inf. Hiding Multimedia Secur.* New York, NY, USA: ACM, 2018, pp. 91–96.
42. M. Barni, A. Costanzo, E. Nowroozi, and B. Tondi, "Cnn-based detection of generic contrast adjustment with jpeg post-processing," in *Proc. IEEE Int. Conf. on Image Process. (ICIP)*, Oct 2018, pp. 3803–3807.
43. F. J. Boroumand, Mehdi, "Deep learning for detecting processing history of images," *Electronic Imaging*, 2018.
44. G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, July 2017, pp. 2261–2269.
45. S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *Int. Conf. Digit. Image Comput.: Tech. and Appl. (DICTA)*. IEEE, 2016, pp. 1–6.
46. M. Stamm, P. Bestagini, L. Marcenaro, and P. Campisi, "Forensic camera model identification: Highlights from the ieee signal processing cup 2018 student competition [sp competitions]," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 168–174, 2018.
47. D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, p. 1, 2009.