3-10-2021

RemNet: Remnant Convolutional Neural Network for Camera Model Identification and Image Manipulation Detection

Abdul Muntakim Rafi
University of Windsor

RemNet: Remnant Convolutional Neural Network for Camera Model Identification and Image Manipulation Detection

By

Abdul Muntakim Rafi

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2021

RemNet: Remnant Convolutional Neural Network for Camera Model Identification and Image Manipulation Detection

by

Abdul Muntakim Rafi

APPROVED BY:

_____

R. Gras

School of Computer Science

_____

R. Razavi-Far

Department of Electrical and Computer Engineering

_____

J. Wu, Advisor

Department of Electrical and Computer Engineering

January 11, 2021

DECLARATION OF CO–AUTHORSHIP / PREVIOUS PUBLICATION

I. Co–Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows: This thesis incorporates the outcome of a research under the supervision of professor Jonathan Wu and collaboration with Dr.Md Kamrul Hasan, Thamidul Islam Tonmoy and Uday Kamal (Chapter 1- 6). In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author, and the contribution of the co-authors was primarily through the provision of proof reading and reviewing the research papers regarding the technical content.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis and have obtained written permission from each of the co-authors to include the above materials in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes two original papers that have been previously published in peer reviewed journals and conferences, as shown in the following table.

Thesis Chapter	Publication title/full citation	Publication Status
Chapters 1-6	Rafi, A.M., Tonmoy, T.I., Kamal, U. et al. RemNet: remnant convolutional neural network for camera model identification. Neural Comput & Applic (2020). https://doi.org/10.1007/s00521-020-05220-y	Published

Chapters 1-6	Rafi A.M., Wu J., Hasan M.K. (2020) L2-Constrained RemNet for Camera Model Identification and Image Manipulation Detection. In: Bartoli A., Fusiello A. (eds) Computer Vision – ECCV 2020 Workshops. ECCV 2020. Lecture Notes in Computer Science, vol 12538. Springer, Cham. https://doi.org/10.1007/978-3-030-66823-5_16	Published

I certify that I have obtained written permission from the copyright owners to include the above published materials in my thesis. I certify that the below material describes work completed during my registration as a graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material in excess of the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained written permission from the copyright owner(s) to include such material in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other university or institution.

ABSTRACT

Camera model identification (CMI) and image manipulation detection are of paramount importance in image forensics as digitally altered images are becoming increasingly commonplace. In this thesis, we propose a novel convolutional neural network (CNN) architecture for performing these two crucial tasks. Our proposed Remnant Convolutional Neural Network (RemNet) is designed with emphasis given on the preprocessing task considered to be inevitable for removing the scene content that heavily obscures the camera model fingerprints and image manipulation artifacts. 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, learn to suppress the unnecessary image contents dynamically. This helps the classification block extract more robust images forensics features from the remnant of the image. We also propose a variant of the network titled L2-constrained Remnant Convolutional Neural Network (L2-constrained RemNet), where 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. The whole network, 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 98.15% on test images from unseen devices and scenes, 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 95.49% on the IEEE Signal Processing Cup 2018 dataset, which indicates its generalizability. 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.

DEDICATION

To

my brother Abdul Muktadir Shafi

my friends Arup Chakrabarty Antar, Md Sirajul Islam, and Jie Huo

my parents

and to everyone else without whom it would never have been possible.

ACKNOWLEDGEMENTS

I would like to convey my heartfelt thanks to Dr. Jonathan Wu who guided me throughout my degree. He has been a kind mentor towards me. I would never be able to finish my thesis in time without his support. I am lucky to have him as my supervisor.

I am very grateful to Dr. Roozbeh Razavi-Far and Dr. Robin Gras for serving as my thesis committee members. I took courses under them during my degree and the knowledge I gathered from the courses helped me make tremendous progress in my thesis. Furthermore, I must thank our department graduate secretary Andria Ballo for her kind guidance.

I am forever in debt to my brother Abdul Muktadir Shafi and my friend Arup Chakrabarty Antar who provided me emotional support during the stressful period of the pandemic. I am thankful to my friend Md. Sirajul Islam who has been my constant companion in this foreign land. I would also like to thank my friends Nahid Hasan Ayon, Jie Huo, Xiexing Feng, Manya Mehta, and Shivang Rana for their sincere cooperation throughout my MASc program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS/SYMBOLS

2D	Two Dimensional
AvgPool	Average Pooling
BN	Batch Normalizaiton
CMI	Camera Model Identification
CNN	Convolutional Neural Network
Conv	Convolutional
Conv 2D	Two-Dimensional Convolution
L2	L2 Norm
PReLU	Parametric Rectified Linear Unit
ReLU	Rectified Linear Unit
RemNet	Remnant Convolutional Neural Network

CHAPTER 1
INTRODUCTION

1.1 Overview

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 [1]. 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 [2]. 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.

1.2 Motivation

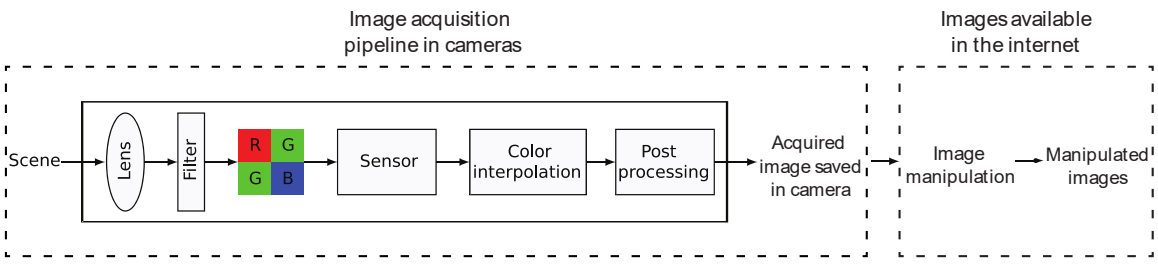

Figure 1: General acquisition pipeline of available digital images.

Digital images go through multiple operations from being captured by a digital camera to being available in different online platforms [1]. We first describe the image acquisition pipeline of digital cameras, as depicted in Figure *1*. In a typical digital camera, the light of a scene passes through a system of lenses and optical filters, which is then collected by an optical sensor. A color filter array (CFA) is used before the sensor to obtain RGB color images so that the individual sensor element records light of a certain color. The remaining color information is estimated from surrounding pixels through a process called CFA interpolation or demosaicing. After demosaicing, the image goes through a number of post-processing (e.g., color correction, edge enhancement, and compression) before it is saved on a storage device. As described in [1], most of these components leave certain `fingerprints' in the images, which can be utilized in different image forensic tasks. Manufacturers generally employ different lens systems in their different camera models, which causes lens distortion artifacts, such as radial lens distortion, chromatic aberration, and vignetting. The CFA layout and demosaicing process vary widely among different models and are generally considered as one of the most distinctive model-specific signatures. The sensor pattern noise (SPN) is the most unique characteristic of a digital camera, and it is used excessively in the literature for source identification. In addition to the camera-internal processing operations, digital images face different manipulations when they are edited by different image editing softwares. Moreover, they are resized or re-compressed when uploaded to photo-sharing websites or social media applications [3]. Therefore, image forensics techniques should be made robust to these common manipulation operations.

To explain the motivation of our proposed method, we first describe the image acquisition pipeline of digital cameras. In a typical digital camera, light from a scene passes through a system of lenses and optical filters, which is then collected by an optical sensor. A color filter array (CFA) is used before the sensor to obtain RGB color images so that an individual sensor element records light of a certain color. The remaining color information is estimated from surrounding pixels through a process called CFA interpolation or demosaicing. After demosaicing, the image goes through a number of

post-processing schemes (e.g., color correction, edge enhancement, and compression) before it is saved on a storage device. As described in [1], most of these components leave certain `fingerprints' in the image which can be utilized in different image forensic tasks. Manufacturers generally employ different lens systems in their different camera models, which causes lens distortion artifacts, such as radial lens distortion, chromatic aberration, and vignetting. The CFA layout and demosaicing process vary widely among different models and are generally considered as one of the most distinctive model-specific signatures. The sensor pattern noise (SPN) is a unique characteristic of a digital camera, and it is often used in the literature for source identification.

In designing CNNs for image forensic tasks, it has been therefore a common practice to use a pre-processing scheme to suppress the image contents and intensify the minute signatures induced by the image acquisition pipeline [5–7]. However, these methods suffer from their own drawbacks of using either fixed kernels or constraints as described earlier. Our main goal is, therefore, to introduce a preprocessing scheme that is completely data-driven but without any imposed constraints or fixed kernels. The weights of the preprocessor block are dynamically extracted from end-to-end training with the classifier by minimizing the loss function for the task. The benefit of designing such a preprocessing block is that it can dynamically adapt itself to different classification blocks in cascade with it. It can also adapt itself well on different datasets. This strategy proves to be crucial for extracting rich camera model-specific higher-level features for our classification task as evident from our experimental results (see Chapter 4).

1.3 Challenges Addressed

Despite the numerous researches conducted in this field, most researchers have explored CMI and image manipulation detection problems discretely. Bayar and Stamm [6] 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 [1]. 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* [7] during testing and cannot overperform by learning features specific to the device or scene. Besides, 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.

1.4 Objective

In this thesis, we propose a general-purpose novel CNN architecture, called Remnant Convolutional Neural Network (RemNet) for performing two crucial tasks in image forensics, CMI and image manipulation 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.

1.5 Structure of the Thesis

We organize the rest of the thesis as follows. Chapter 2 contains a brief description about the relevant researches that has been conducted in the field. We describe our proposed network and loss function in Chapter 3. We discuss our training and evaluation criteria,

along with the experimental results in Chapter 4. We explore the significance of our proposed remnant blocks in Chapter 5. Finally, we conclude in Chapter 6.

CHAPTER 2
BACKGROUND

2.1 Related Work

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 [8-9], 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. [9]. However, using separate external features for each camera model is an unmanageable task [10]. Consequently, researchers have focused on utilizing the intrinsic features, such as the Color Filter Array (CFA) pattern [11], interpolation algorithms [12], and Image Quality Metrics (IQM) [13]. Utilizing Photo Response Non-Uniformity (PRNU) noise patterns have been proposed for device-level identification [14-15]. Although sensor noise carries device-specific noise artifacts, researchers have developed methods to perform CMI using sensor noise patterns [16-17].

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 [18]. In the case of image manipulation, traces are found in the image according to the type of processing it has gone through [19]. Following this theory, researchers have used distinct forensic approaches for identifying different kinds of image manipulation, such as resizing [20-21], contrast enhancement [22-23], and multiple jpeg compression [24-25], 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 [26]. For that reason, more recent researches have focused on becoming data-driven, such as utilizing local pixel dependencies used in steganalysis [27-28] to perform CMI [4], [29] and detect image manipulation [30]. In

[31], 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 [2].

2.2 Recent Trend of Research

Recently, researchers have started applying Convolutional Neural Networks (CNNs) for image forensic tasks [32]. It is expected as CNNs have performed extremely well in different image classification tasks [33]. 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 [6]. 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. [34] have proposed using a median filter, whereas Tuama et al. [5] 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 [17]. Therefore, fixed filters as preprocessor may lose forensics-related features. Bayar and Stamm [6] have proposed a data-driven constrained convolutional layer which has performed better than the above-mentioned fixed filters. Bayar and Stamm [6] 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 [35]. In [36], the authors use CNN and support vector machine (SVM) for CMI, where they use the CNN part as a feature extractor. In [37], explores the performance of DenseNet [38] in both CMI and image manipulation detection. In [2], the authors investigate the performance of densely connected CNNs in image manipulation 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 researches exist in the literature that use auxiliary loss function to enhance the

discrimination between learned features [39-41]. There is a scope of utilizing such auxiliary loss functions in the modular CNN architectures for image forensics.

2.3 Convolutional Neural Networks

Convolutional neural networks are particular types of deep neural networks that have gained attention from research community and industry, achieving empirical successes in tasks such as object recognition, object detection, speech recognition, and natural language processing [42]. They automatically extract discriminatory features from raw input information which are very difficult to obtain through traditional hand-crafted feature engineering [42].

In a typical CNN, the input information is passed through several convolution layers where they are convolved with the filters to generate output feature maps. If x_m^l is the m-th input feature in the l-th layer and $w_{\{n,m\}}^l$ is the kernel weight parameter of the l-th layer, then the n-th output feature in that layer y_n^l is computed as

$$y_n^l = \sum_m^{M^{l-1}} w_{n,m}^l * x_m^l + b_n^l$$

where M^{l-1} is the number of input maps, $*$ denotes convolution operation, and b_n^l is the bias of the n-th output map in the l-th level.

The convolution operations are usually followed by activation functions. The purpose of these functions are to introduce nonlinearity in the network. In computer vision tasks, ReLU [43] is the most popular choice for activation which is defined as

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ 0, & \text{if } y_i \leq 0 \end{cases}$$

However, ReLU activation applies a constraint on feature generation by passing only positive values while all negative values are set to zero. As a result, a number of modifications of the ReLU function have been proposed in the literature of which Parametric ReLU (PReLU) [44] has gained popularity in image recognition tasks in

recent years. Instead of setting the negative values to zero, PReLU incorporates a learnable parameter a_i as

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

While training neural networks, internal covariance shift causes the distribution of each layer's inputs to change which subsequently slows down the training process. To mitigate this problem, researchers have proposed various normalizing schemes of which batch normalization (BN) [45] is used extensively in recent works. If x_i is the i-th input feature in a mini-batch B with m input features, then the output y_i with BN is calculated as

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$$
$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$$
$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$
$$y_i = \gamma \hat{x}_i + \beta$$

where μ_B and σ_B^2 are the mean and the variance of the mini-batch, respectively, and β and γ are two learnable parameters. The variable ϵ represents a very small value that is used to prevent possible division by zero cases.

To reduce the dimensionality of the feature maps, various pooling operations are performed, such as, max-pooling, average-pooling, etc. The max-pooling operation takes a window of $q \times q$ and keeps only the maximum value of the selected window whereas average-pooling keeps only the average value. Pooling layers perform subsampling on the feature space in such a way that the most dominant features are retained.

The input is passed through successive convolutional layers along with activation, BN and pooling layers. Eventually, the feature space is gradually reduced to the number of classes N to get $y = [y_1, y_2, \ldots, y_N]$, where y_i represents the score of the i-th class. Finally, a softmax activation is applied on the output layer mapping the N class scores to N probability values $p = [p_1, p_2, \ldots, p_N]$ for each class which sum up to 1:

$$p_i = \frac{\exp(y_i)}{\sum_{n=1}^{N} \exp\,(y_n)}$$

The training of a neural network is conducted through successive forward and backward propagations of the data. During each forward pass, we get a probability output score for each input data. A loss is then calculated based on the predicted output and the ground truth. For multi-class classification problems, categorical crossentropy loss function is mostly used which is given by

$$L = \sum_{k=1}^{N} y_i^{*(k)} \log\left(y_i^{(k)}\right)$$

where $y_i^{*(k)}$ and $y_i^{(k)}$ are, respectively, the true label and the network output of the i-th image at the k-th class among the N classes. This loss is backpropagated to update the weights of the network parameters by using various optimization algorithms, such as, stochastic gradient descent (SGD) [46] and adaptive momentum (Adam) [47].

CHAPTER 3

PROPOSED NETWORK

In this paper, we propose a CNN-based patch-level method for CMI and image manipulation detection. A block diagram of our proposed method is shown in Figure *2*.

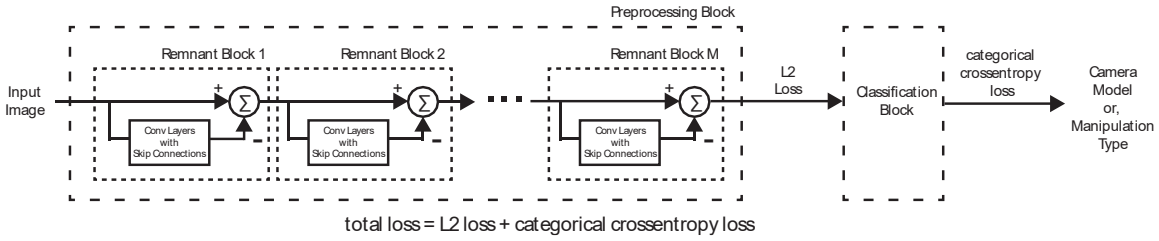

Figure 2: Block diagram of our proposed method.

As shown in Figure 3, we first extract high quality clusters of size 256×256 from an input image. From each cluster, patches of size 64×64 are taken and fed to the network. 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.

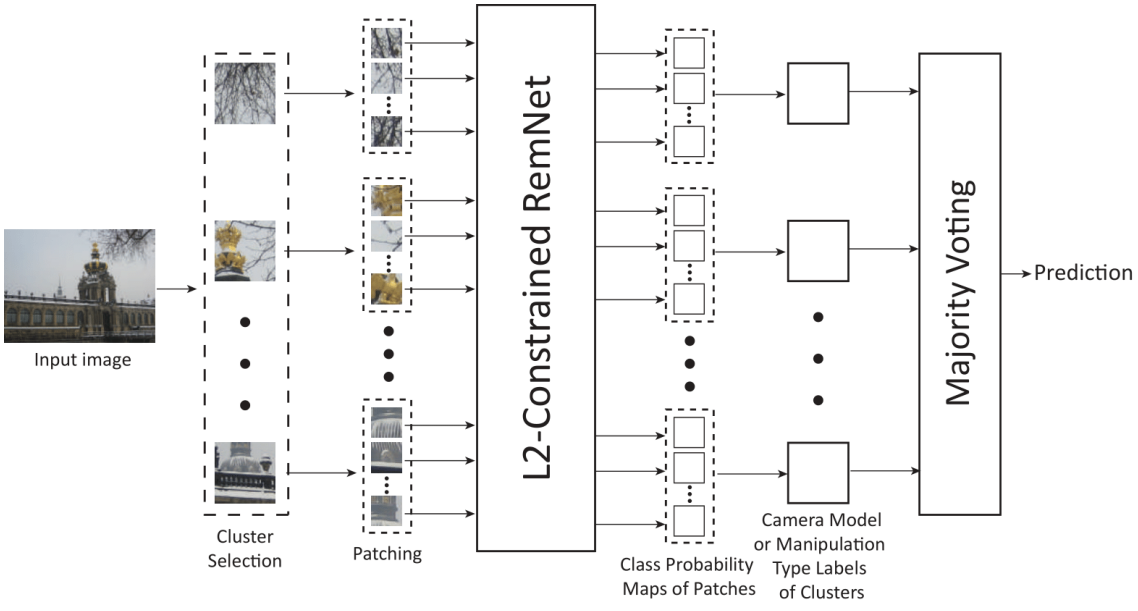

Figure 3: Schematic representation of the proposed method for CMI and image manipulation.

3.1 Network Architecture

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 Figure *4*). 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.

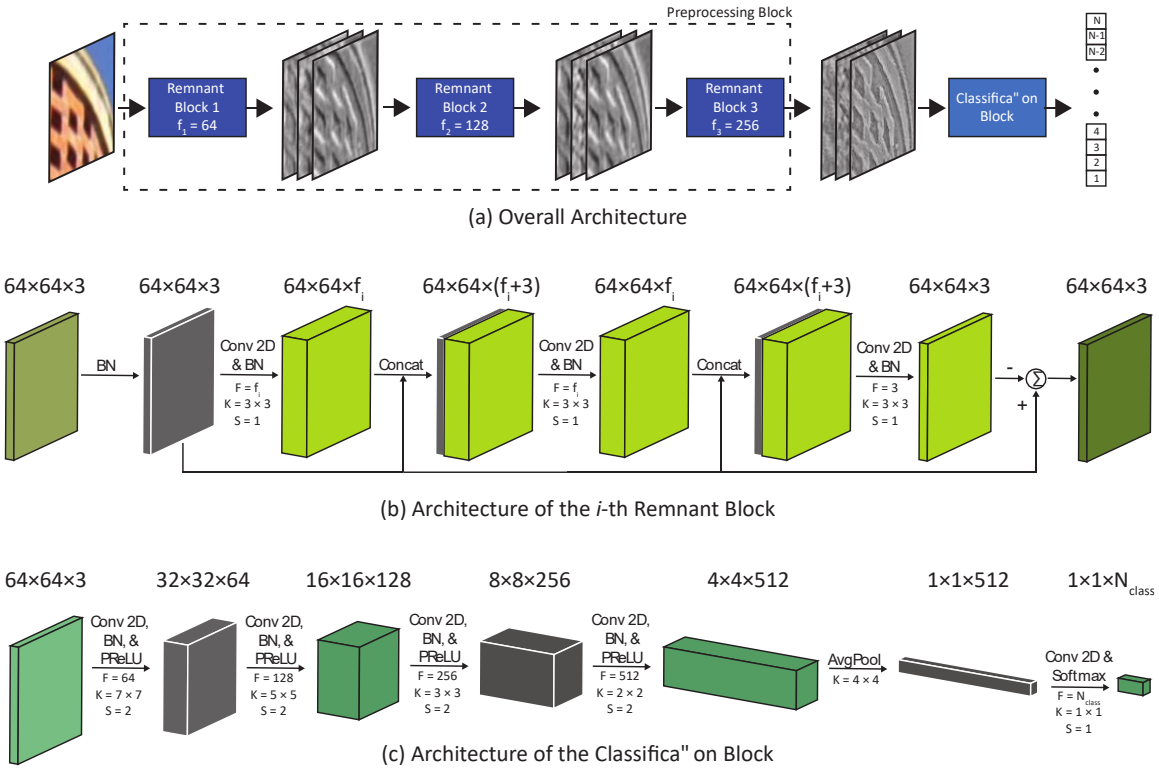

Figure 4: The architecture of our proposed RemNet. (a) Illustrates the overall architecture with three remnant blocks with one classification block. The architectures of the remnant and classification blocks are depicted in (b) and (c), respectively. In (b) and (c), AvgPool, BN, and Conv2D represent 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.

3.1.1 Preprocessing Block

The preprocessing block consists of several remnant blocks. The detailed architecture of the remnant block is shown in Figure *4*(b) and Table 1. 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 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 Figure *4*(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 chapter.

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.

Table 1: Architecture of the i-th remnant block

Layers	Output Size	Kernels*
BN	$64 \times 64 \times 3$	
Conv 2D & BN	$64 \times 64 \times f_1$	$F = f_1, K = 3 \times 3, S = 1$
Concatenate	$64 \times 64 \times (f_1 + 3)$	
Conv 2D&BN	$64 \times 64 \times f_1$	$F = f_1, K = 3 \times 3, S = 1$
Concatenate	$64 \times 64 \times (f_1 + 3)$	
Conv 2D&BN	$64 \times 64 \times 3$	$F = 3, K = 3 \times 3, S = 1$
Subtract	$64 \times 64 \times 3$	

* The letters F, K, and S represent the number of filters, their kernel size, and strides, respectively, in the corresponding convolution layers.

The remnant blocks are somewhat influenced by the highway networks proposed by Srivastava et al. in [48]. A plain convolutional layer applies a linear transformation H(parameterized by $\boldsymbol{W_H}$) on its input $\boldsymbol{x}$ to produce its output $\boldsymbol{y}$:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H})$$

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 $T(\boldsymbol{x}, \boldsymbol{W_T})$ and $C(\boldsymbol{x}, \boldsymbol{W_C})$ are defined such that

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

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 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, the equation of highway network becomes

$$\mathbf{y} = \mathbf{x} - H(\mathbf{x}, \mathbf{W_H})$$

The residual network (ResNet) [49] is also a variant of the highway network [50] where the choices for both T and C are 1 for the residual blocks. Besides, the transformation H used in [49] works as a nonlinear feature extractor whereas the H of our remnant blocks performs linear filtering operation. Also, ResNet does not use any intra-block skip connections. Most importantly, the remnant blocks are used at the beginning part of the network. In Chapter 4, we provide a comparison of our proposed network with ResNet [49] in camera model identification and image manipulation detection tasks. In Chapter 5, we show how the performance of ResNet [49] can be improved by adding remnant blocks at the beginning of the network.

3.1.2 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 2. 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.

Table 2: Architecture of our Proposed RemNet

Layers	Output Size	Kernels
Remnant Block 1	$64 \times 64 \times 3$	$f_1 = 64$
Remnant Block 2	$64 \times 64 \times 3$	$f_2 = 128$
Remnant Block 3	$64 \times 64 \times 3$	$f_3 = 256$
Classification Block		
Conv 2D, BN, & PReLU	$32 \times 32 \times 64$	$F = 64, K = 7 \times 7, S = 2$
Conv 2D, BN, & PReLU	$16 \times 16 \times 128$	$F = 128, K = 5 \times 5, S = 2$
Conv 2D, BN, & PReLU	$8 \times 8 \times 256$	$F = 256, K = 3 \times 3, S = 2$
Conv 2D, BN, & PReLU	$4 \times 4 \times 512$	$F = 512, K = 2 \times 2, S = 2$
Average Pool	$1 \times 1 \times 512$	$K = 4 \times 4$
Conv 2D	$1 \times 1 \times 18$	$F = N, K = 1 \times 1, S = 1$
Softmax	N	$-$

* The letters F, K, and S represent the number of filters, their kernel size, and strides, respectively, in the corresponding convolution layers.

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 [51]. As per the design principles introduced in [6], 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 [52]. 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 [53]. We have also experimentally verified this in our experimental results section.

The average-pool operation is used as per the conventional design structure of CNNs [38], [49], [54] 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.

3.2 Loss Function

The preprocessing block contains M remnant blocks. The i-th remnant block applies a transformation H_i on its input $\boldsymbol{x_i}$ (which is also the output of the $(i-1)$-th remnant block) and subtracts it from its input to produce the output y_{p_i}:

$$\mathrm{y_{Pi}} = \mathbf{x_i} - H(\mathbf{x_i}, \mathbf{W_{Pi}})$$

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

$$L_2 = \sum_{l=1}^{N_{\mathrm{param}}} \boldsymbol{y}_{\boldsymbol{p}_{\boldsymbol{M}_l}}^{\boldsymbol{2}}$$

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

$$\mathrm{y_c} = G\left(\mathrm{y_{pM}}, \mathbf{W_c}\right)$$

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

$$L_{xent} = \sum_{k=1}^{N_{class}} y_{c_i}^{*(k)} \log\left(y_{c_i}^{(k)}\right)$$

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 total loss L is defined using the following equation:

$$L = \alpha * L_2 + L_{xent}$$

Here, α indicates how much weight we want to put in the suppression of the residue from the preprocessor block. A larger choice for α may cause the vanishing gradient problem for the classifier [55]. We empirically set the value of α as 0.5. During backpropagation, the gradient of L_2 is used to update the weights of the preprocessing 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.

CHAPTER 4
EXPERIMENTAL RESULTS

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.

4.1 Camera Model Identification

4.1.1 Results on Dresden Dataset

We comprehensively evaluate our overall approach on the Dresden dataset [56]. 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 [56]. 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 [1]. 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 3.

4.1.1.1 Training and testing strategy

Training a CMI network is challenging because of the existence of device-specific features such as PRNU noise [15], [17] 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 3). We refer to these sets as *unaltered* train, validation, and test sets. This splitting policy, proposed in [36], 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.

Table 3: 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 Nikon D70S	373	1 1	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		

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 contain enough statistical information about the camera model [36]. Therefore, different authors have used different cluster selection strategies in the literature. In [35], 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 [36], 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 [36]. For each cluster P in an image, its quality $Q(P)$ is computed as

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

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 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 Figure 5).

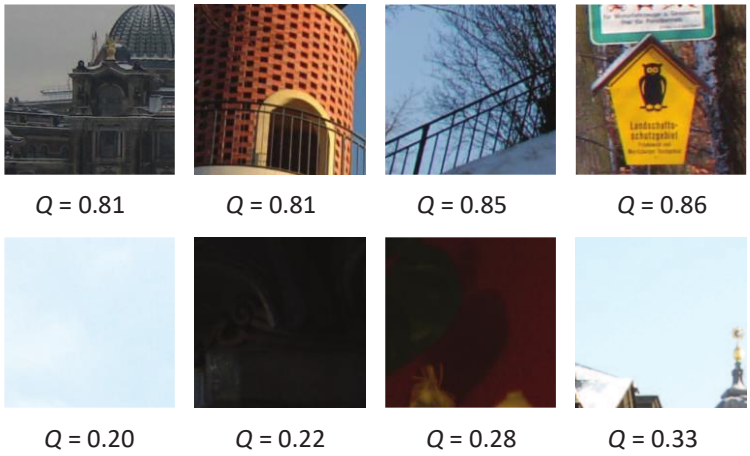

Figure 5: Examples of clusters of different qualities with their quality indices.

We found that this quality assessment is consistent with the *others* category mentioned in [35]. According to the definition in [35], 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. [35].

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 [35], [36], [57]. 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 introduce 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 Figure *5*. 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 [58]. We use categorical cross-entropy as the loss function and Adam [47] 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 decreased by a factor of 0.5 if the softmax classification loss (L_{xent}) does not decrease in three 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 70 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:

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

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.

4.1.1.2 Comparison of Design Choices

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 4. 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%.

Table 4: 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

4.1.1.3 Comparison with state-of-the-art networks on unaltered images

We compare our results with the established methods in CMI-- constrained-convolutional network [6], fusion residual network [35] and first steps toward the camera model identification with convolutional neural networks [36]. The reason behind choosing [6] and [35] 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 [35], we implement the fusion residual network for the *others* category only, instead of each of the three different categories mentioned in [35]. We also include [36] in our comparison as we adopt their cluster selection strategy. Recently, several works such as [59–61] 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 [49] and DenseNet [38]. 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 5: Accuracy of different methods trained and tested on the unaltered train and test sets of the Dresden database

Method	Accuracy (%)

Bayar and Stamm [6]	95.56
Yang et al. [35]	94.81
Bondi et al. [36]	90.55
ResNet [49]	92.40
ResNeXt [62]	93.33
DenseNet [38]	93.33
RemNet	97.03
L2-Constrained RemNet	97.59

The results presented in Table 5 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.

4.1.1.4 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 [63]. 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 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.

Table 6: 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 [6]	93.89
Yang et al. [35]	95.19
Bondi et al. [36]	92.59
ResNet [49]	95.19
ResNeXt [62]	95.55
DenseNet [38]	95.05
RemNet	97.59
L2-Constrained RemNet	98.15

After training on the augmented train set, evaluation is carried out on the unaltered test set. The results are presented in Table *6*. If we compare the results of Table *6* with that of Table *5*, we observe that these post-processing schemes, as a form of data-augmentation, indeed improve the performance of all the networks except that in [6]. 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

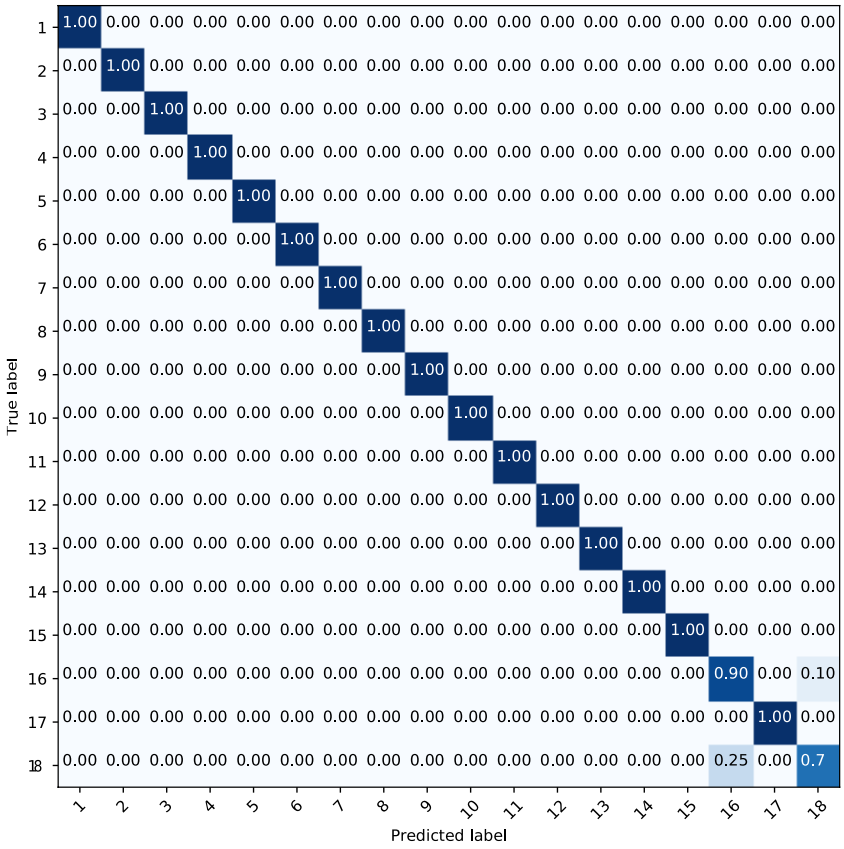

Figure 6: Confusion Matrix of our proposed RemNet trained on the augmented train set and tested on the unaltered test set of the Dresden database. The input and predicted label correspond to the Serial No. used in Table 3.

matrix in Figure *6*. 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 [5]. As mentioned in [1], images captured with camera models of the same manufacturer are likely to share some components which makes it harder to separate them.

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 7. The highest result for each manipulation factor is made

bold (see Table 7). 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 [36] achieves better performance in two. For Resize manipulation, we see that ResNeXt [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.

Table 7: 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 [6]	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. [35]	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. [36]	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 [38]	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 [49]	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 [62]	94.25	95.55	93.88	95.18	95.18	**94.82**	94.25	94.07	95.00	95.00	**96.11**	**95.55**
RemNet	96.11	97.22	96.11	95.56	**97.59**	**94.82**	92.59	92.78	95.00	93.33	92.04	92.41
L2-Constrained RemNet	**96.29**	**98.14**	**97.59**	**97.96**	92.96	93.33	**96.11**	**97.03**	**96.67**	**96.67**	90.74	91.66

4.1.1.5 Justification of Using the L2 Loss

L2-constrained RemNet achieves an overall accuracy of 98.15%, which is better than all other approaches we compare with (see Table *6*). It should be noted that we set the value for in our custom loss function (5) empirically. We have achieved accuracy of 97.77%, 98.15%, and 97.77%, when α 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 pre-processing block in our network. First, we train the RemNet without any pre-processing 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 pre-processing block justifies the use of the preprocessing step (see Chapter 5). Similarly, the lower accuracy of RemNet without any additional loss justifies the use of the auxiliary loss (see Table *6*). 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.

4.1.2 Results on the SP 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 [64]. 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 *8*.

Table 8: 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 *8* 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:

$$\text{Accuracy} = 0.7 \times (\text{Accuracy of Unaltered Images}) + 0.3 \times (\text{Akcuracy of Manipulated Images}).$$

Table *9* 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. This satisfactory performance is evidence of the generalizability of our approach. Among the other networks, wider [35] and deeper ([38], [49]) networks perform comparatively better than the shallower ones.

Table 9: Accuracy of different methods on the IEEE SP Cup 2018 testing dataset

Method	Accuracy (%)
Bayar and Stamm [6]	90.97

Yang et al. [35]	94.83
Bondi et al. [36]	90.07
ResNet [49]	93.92
DenseNet [38]	93.70
RemNet	95.11
L2-Constrained RemNet	95.49

4.1.3 Data Imbalance Problem

If the data used to train a network is not evenly distributed into different classes, then supervised machine learning algorithms can become biased or skewed to specific classes. A machine learning algorithm should be trained with a somewhat equal number of images in each category in an ideal situation. Data imbalance can lead to poor performance, particularly for the classes with fewer samples available during training. We can see in Table 3 that the number of images available for different camera models is quite imbalanced. Therefore, we have the possibility of facing a data imbalance problem in our experiments. However, we can see in Figure 6 that it is not the case in our experiments. In CMI, RemNet achieves 100% accuracy for 16 camera models of the 18 camera models in the Dresden database. Despite having an unequal number of training images for different camera models, it does not affect the network's performance. The alleviation of the data imbalance problem can be attributed to our patch selection strategy, the data augmentation methods, and the proposed network's better performance.

4.2 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 (64×64). 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.

Table 10: Accuracy (in %) of different methods in image manipulation detection

Method	Accuracy (%)
Yang et al. [35]	91.74
Bayar and Stamm [6]	87.28
RemNet	98.27
L2-Constrained RemNet	99.68

Table 11: 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	80	0.8	0.9	1.1	1.2
Yang et al. [35]	99.07	98.52	97.04	**98.70**	49.44	100	100	100	100	97.40	60.74	100
Bayar and Stamm [6]	94.44	83.33	77.22	90.56	11.30	100	100	100	100	100	90.93	99.63
RemNet	100	**99.81**	**99.63**	**100**	81.48	98.33	100	100	100	100	100	100
L2-Constrained RemNet	**100**	99.63	99.26	**98.70**	**100**	98.7	**100**	**100**	**100**	**100**	**100**	**100**

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 γ = 0.5, 0.75, 1.25, and 1.5; $540 \times 3 = 1620$ JPEG compressed images compressed with factors of 85%, 90%, 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 *10*. We achieve an overall accuracy of 99.68% in this task whereas RemNet, Bayar and Stamm [6], and Yang et al. [35] achieve 98.27%, 87.28% and 91.74%, respectively. We demonstrate the detection accuracy for different factors of manipulation in Table *11*. For gamma-corrected images, the performances of [35], RemNet and L2-Constrained RemNet are substantially better than that of [6]. In the case of JPEG compression, all four networks perform almost the same except at the compression factor of 95, where [6] and [35] fail miserably by misclassifying most of the compressed images as unaltered images. There is a significant drop in the detection accuracy for RemNet 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 performs the same by attaining a 100% accuracy. Of the other two networks, [6] performs better than [35].

CHAPTER 5

SIGNIFICANCE OF THE REMNANT BLOCKS

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.

5.1 With and Without the Remnant Blocks

In order to validate the significance of our proposed preprocessor, we train and test our proposed classifier network without the remnant blocks. We also train and test the network proposed in [36], ResNet [49], and DenseNet [38] 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

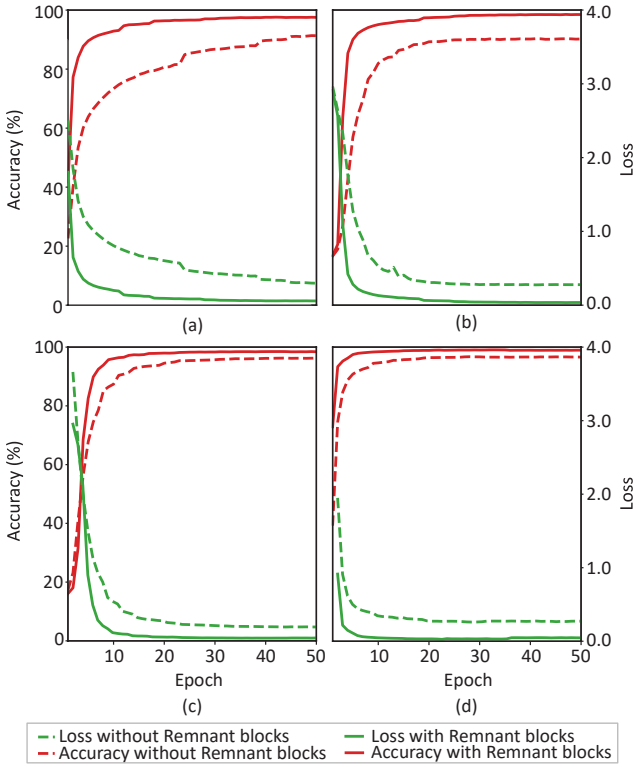

Figure 7: Training history of (a) Bondi et al., (b) DenseNet, (c) ResNet, and (d) Our Proposed Classifier, with and without remnant blocks, for training with the augmented train set of the Dresden database.

database. It is to be mentioned that we do not perform similar experiments on [35] and [6] since these networks already consist of their own preprocessing schemes.

The training histories of the models are presented in Figure *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.

Table 12: 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.	90.55	95.92	92.59	96.29
ResNet	92.40	96.85	95.18	98.33
DenseNet	93.33	96.29	95.01	98.14
RemNet	93.31	97.03	95.74	97.59

From the experimental results presented in Table *12*, 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.

Table 13: 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				Rescale			
Factor	0.5	0.75	1.25	1.5	95	90	85	80	0.8	0.9	1.1	1.2
Bondi et al.	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	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	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

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 [36], ResNet [49], and DenseNet [38] with remnant blocks on the manipulated test dataset. The experimental results shown in Table *13* 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 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.

To verify the effect of remnant blocks on different networks for the IEEE SP Cup 2018 dataset, we train the networks [36], [49], [59] in cascade with remnant blocks. The experimental results are presented in Table *14*. It is clear from the table that the addition of the remnant blocks improves the performances of the aforementioned networks. Therefore, our presumption that the remnant blocks can improve the performance of any classification network in CMI is further verified in different datasets.

Table 14: 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

5.2 Frequency Analysis

To demonstrate that the dynamically designed remnant blocks truly performs the desired pre-processing task, we show in Figure 8 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 [5], [34], respectively. Figure 8(a) shows the RGB image, Figure 8(b)- Figure 8(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 [65], and Lukas et al. [17] have explicitly stated that both low and high

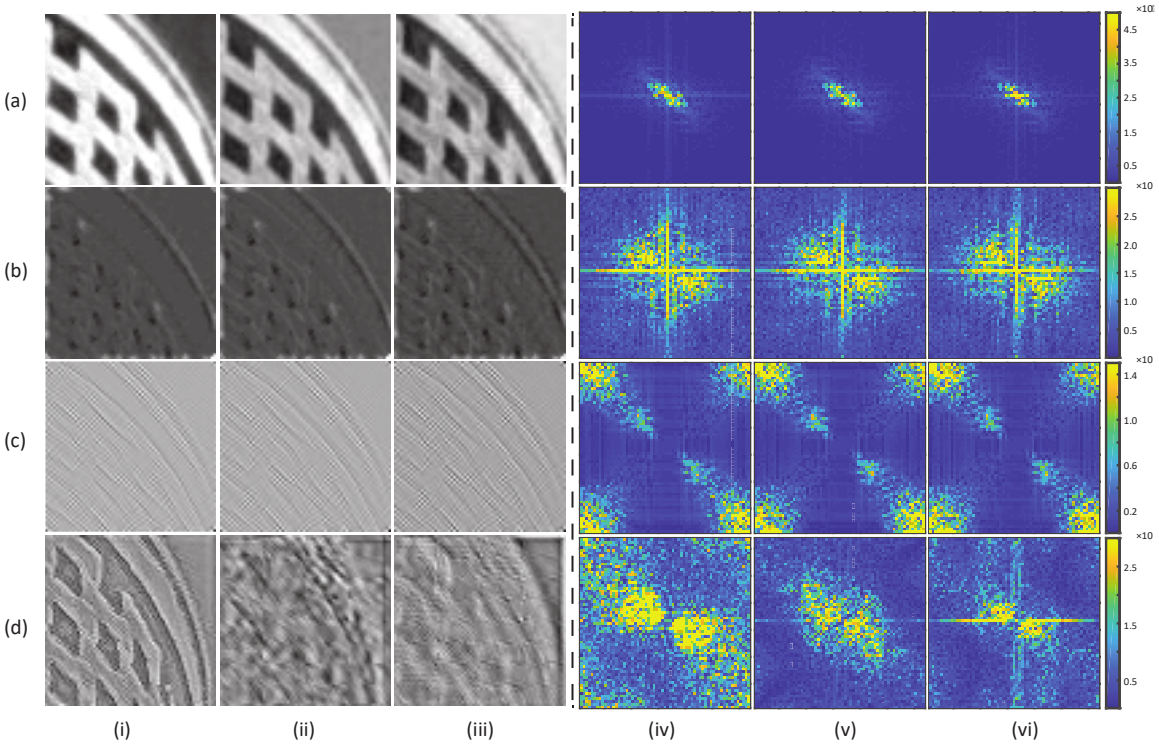

Figure 8: Comparison of outputs of various pre-processing schemes. (a) Input image, (b) median filter residue, (c) high-pass filter output, and (d) output of the third remnant block of our proposed RemNet. Columns (i), (ii), and (iii) correspond to different output channels, whereas columns (iv), (v), and (vi) depict their frequency responses, respectively.

frequency information are required for CMI. We, therefore, claim that our data-adaptive preprocessing performs better filtering operation, preserving the camera signature from a wide range of frequencies.

5.3 Advantage of Data-Adaptive Filters

The advantage of using data-adaptive dynamic filters of different frequency bands for different image channels is demonstrated in Table 15. Here, we first train our proposed classification block without the remnant blocks. Then we constrain the last remnant block of our proposed RemNet to look at the same frequencies in all three channels. Lastly, we replace the remnant blocks with the fixed highpass filter proposed in [5]. The high-pass filter is followed by our proposed classification block. All the networks are trained on the augmented training set and tested on the unaltered test set. As evident, the performance of the proposed RemNet is better than using only the classification block or the constrained RemNet. It can be also observed that using a fixed high-pass filter with the classification block significantly deteriorates the performance of the network as compared to any other configuration as demonstrated in Table 15. These results suggest that the dynamic filters are superior to the constrained or fixed filters.

Table 15: Accuracy of different constrained models and our proposed model trained on the augmented train set and tested on the unaltered test set of the Dresden database

Method	Accuracy (%)
Proposed classification block	95.74
Constrained RemNet	96.48
High-pass filter [5] followed by our proposed classification block	92.14
Proposed RemNet	97.59

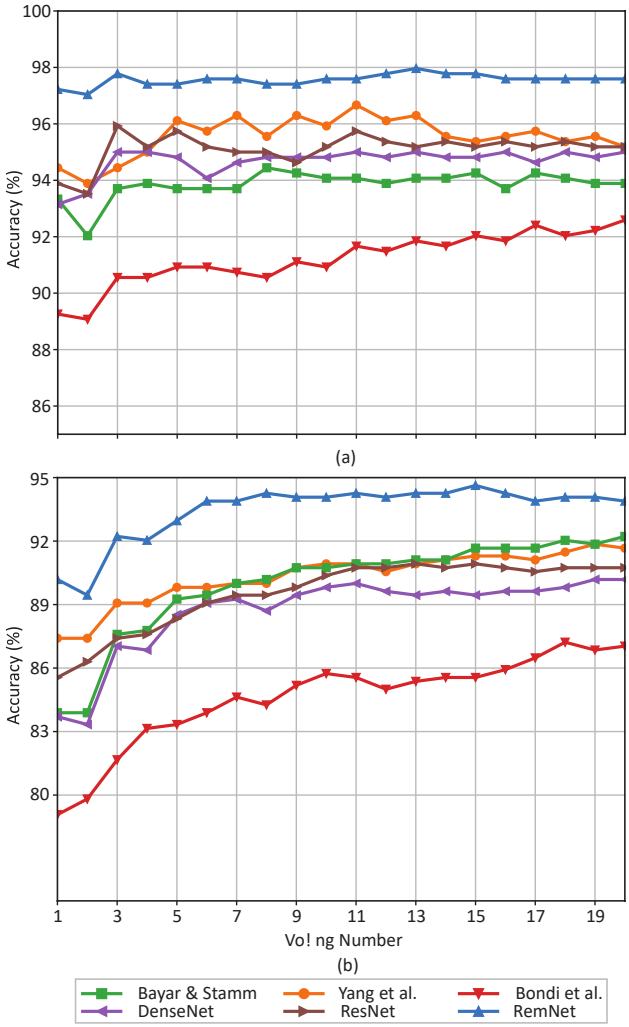

Figure 9: Results of varying voting number for (a) rich quality clusters and (b) poor quality clusters of different methods, trained on the augmented train set, for testing with the unaltered test set of the Dresden database.

5.4 Performance on Good and Bad Patches

In Figure 9, 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 Figure 5), 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 Figure 9(b). In both of these two cases, our proposed RemNet outperforms the other networks in comparison.

5.5 Visualizing the Models Class Activation

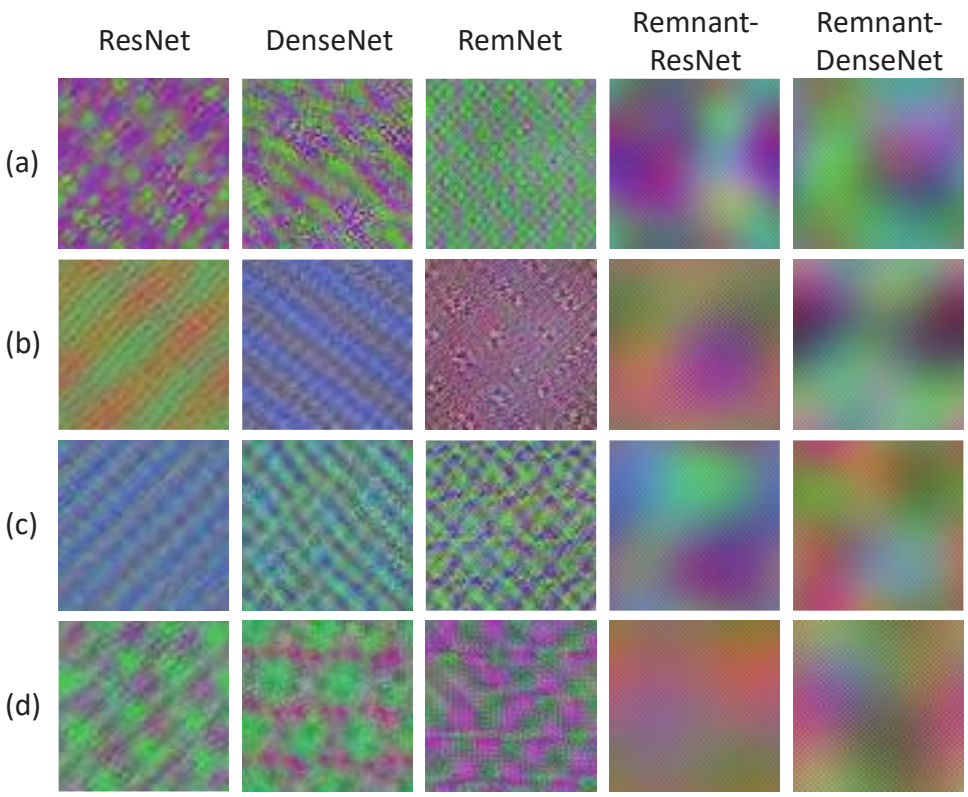

Figure 10: Visualization of input activation of (a) Canon IXUS 70, (b) CanonEX-Z150, and (c) FujiFilm FinePix J50 for different networks trained on the Dresden database.

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. [66] 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 [49], DenseNet [38], and our proposed network in Figure 10. From this figure, it is evident that deep networks 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.

CHAPTER 6
CONCLUSION

In this thesis, a novel CNN model has been proposed for performing two important image forensics tasks, namely, CMI and image manipulation detection. To address the problem effectively, a dynamic CNN-based preprocessing block has been placed in cascade with the shallow CNN-based classifier for enhancing the intrinsic image forensics 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 closed 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 indicate that they are suitable as a general-purpose preprocessing scheme for varieties of CMI networks. 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.

In future works, we wish to explore the possibility of coming up with an improved loss function to facilitate the training of the RemNet better. We can further study the image acquisition pipeline to investigate appropriate loss functions, which will help suppress the image's image contents at the preprocessing step. It should be noted that the accompanying design choices of the RemNet may also change for different loss functions. We wish to explore the potential of such a preprocessing scheme in other image forensic tasks as well.

REFERENCES/BIBLIOGRAPHY

[1] M. Kirchner and T. Gloe, “Forensic camera model identification,” *Proc. WOL Handb. Digit. Forensics Multimed. Data Devices*, pp. 329–374, 2015.

[2] Y. Chen, X. Kang, Y. Q. Shi, and Z. J. Wang, “A multi-purpose image forensic method using densely connected convolutional neural networks,” *J. Real-Time Image Process.*, vol. 16, no. 3, pp. 725–740, 2019.

[3] B. Bayar and M. C. Stamm, “Augmented convolutional feature maps for robust cnn-based camera model identification,” in *Proc. IEEE Int. Conf. on Image Process., (ICIP)*, 2017, pp. 4098–4102.

[4] C. Chen and M. C. Stamm, “Camera model identification framework using an ensemble of demosaicing features,” in *Proc. IEEE Int. Works. Infor. (WIFS)*, 2015, pp. 1–6.

[5] 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)*, 2016, pp. 1–6.

[6] B. Bayar and M. C. Stamm, “Design principles of convolutional neural networks for multimedia forensics,” *Electron. Imaging*, vol. 2017, no. 7, pp. 77–86, 2017.

[7] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, “A survey on data leakage prevention systems,” *J. Netw. Comput. Appl.*, vol. 62, pp. 137–152, 2016.

[8] M. C. Stamm, M. Wu, and K. J. R. Liu, “Information forensics: An overview of the first decade,” *{IEEE} Access*, vol. 1, pp. 167–200, 2013.

[9] A. Piva, “An overview on image forensics,” *Proc. ISRN Signal Process.*, vol. 2013, 2013.

[10] H. Farid, “Image forgery detection,” *{IEEE} Signal Process. Mag.*, vol. 26, no. 2, pp. 16–25, 2009.

[11] 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)*, 2005, vol. 3, pp. III--69.

[12] M. Kharrazi, H. T. Sencar, and N. Memon, “Blind source camera identification,” in *Proc. IEEE Int. Conf. on Image Process., (ICIP)*, 2004, vol. 1, pp. 709–712.

[13] 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.

[14] 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*, 2007, pp. 1–6.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] B. Bayar and M. C. Stamm, “Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection,” *{IEEE} Trans. Inf. Forensics Secur.*, vol. 13, no. 11, pp. 2691–2706, 2018.

[20] A. C. Popescu and H. Farid, “Exposing digital forgeries by detecting traces of resampling,” *IEEE Trans. signal Process.*, vol. 53, no. 2, pp. 758–767, 2005.

[21] X. Feng, I. J. Cox, and G. Doerr, “Normalized energy density-based forensic detection of resampled images,” *IEEE Trans. Multimed.*, vol. 14, no. 3, pp. 536–545, 2012.

[22] M. C. Stamm and K. J. R. Liu, “Forensic detection of image manipulation using statistical intrinsic fingerprints,” *{IEEE} Trans. Inf. Forensics Secur.*, vol. 5, no. 3, pp. 492–506, 2010.

[23] H. Yao, S. Wang, and X. Zhang, “Detect piecewise linear contrast enhancement and estimate parameters using spectral analysis of image histogram,” 2009.

[24] T. Bianchi and A. Piva, “Image forgery localization via block-grained analysis of JPEG artifacts,” *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 3, pp. 1003–1017, 2012.

[25] R. Neelamani, R. De Queiroz, Z. Fan, S. Dash, and R. G. Baraniuk, “JPEG compression history estimation for color images,” *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1365–1378, 2006.

[26] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, “Determining image origin and integrity using sensor noise,” *{IEEE} Trans. Inf. Forensics Secur.*, vol. 3, no. 1, pp. 74–90, 2008.

[27] J. Fridrich and J. Kodovsky, “Rich models for steganalysis of digital images,” *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 3, pp. 868–882, 2012.

[28] T. Pevny, P. Bas, and J. Fridrich, “Steganalysis by subtractive pixel adjacency matrix,” *IEEE Trans. Inf. Forensics Secur.*, vol. 5, no. 2, pp. 215–224, 2010.

[29] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, “A study of co-occurrence based local features for camera model identification,” *Multimed. Tools Appl.*, vol. 76, no. 4, pp. 4765–4781, 2017.

[30] X. Qiu, H. Li, W. Luo, and J. Huang, “A universal image forensic strategy based on steganalytic model,” in *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, 2014, pp. 165–170.

[31] W. Fan, K. Wang, and F. Cayre, “General-purpose image forensics using patch likelihood under image statistical models,” in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2015, pp. 1–6.

[32] P. Yang, D. Baracchi, R. Ni, Y. Zhao, F. Argenti, and A. Piva, “A Survey of Deep Learning-Based Source Image Forensics,” *J. Imaging*, vol. 6, no. 3, p. 9, 2020.

[33] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.

[34] 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.

[35] 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.

[36] 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.

[37] A. M. Rafi *et al.*, “Application of DenseNet in Camera Model Identification and Post-processing Detection.,” in *CVPR Workshops*, 2019, pp. 19–28.

[38] 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)*, 2017, vol. 1, no. 2, p. 3.

[39] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in *European conference on computer vision*, 2016, pp. 499–515.

[40] Y. Wen, Z. Li, and Y. Qiao, “Latent factor guided convolutional neural networks for age-invariant face recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4893–4901.

[41] Y. Sun, X. Wang, and X. Tang, “Deeply learned face representations are sparse, selective, and robust,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2892–2900.

[42] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, p. 436, 2015.

[43] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” 2010.

[44] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[45] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv Prepr. arXiv1502.03167*, 2015.

[46] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*, Springer, 2010, pp. 177–186.

[47] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv Prepr. arXiv1412.6980*, 2014.

[48] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *arXiv Prepr. arXiv1505.00387*, 2015.

[49] 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.

[50] K. Greff, R. K. Srivastava, and J. Schmidhuber, “Highway and residual networks learn unrolled iterative estimation,” *arXiv Prepr. arXiv1612.07771*, 2016.

[51] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *arXiv Prepr. arXiv1412.6806*, 2014.

[52] 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.

[53] B. Bayar and M. C. Stamm, "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.*, 2016, pp. 5–10.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv Prepr. arXiv1409.1556*, 2014.

[55] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4799–4807.

[56] T. Gloe and R. Böhme, "The Dresden Image Database for Benchmarking Digital Image Forensics.," *J. Digit. Forensic Pract.*, vol. 3, pp. 150–159, 2010.

[57] 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. 24973–24982, 2018.

[58] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010, pp. 249–256.

[59] 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.*, 2018, pp. 91–96, doi: 10.1145/3206004.3206013.

[60] 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, doi: 10.1109/ICIP.2018.8451698.

[61] F. J. Boroumand Mehdi, "Deep Learning for Detecting Processing History of Images," *Electron. Imaging*, 2018.

[62] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[63] 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)*, 2016, pp. 1–6.

[64] 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.

[65] 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.

[66] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, “Visualizing higher-layer features of a deep network,” *Univ. Montr.*, vol. 1341, no. 3, p. 1, 2009.

APPENDIX

Springer Permission to Reprint

In reference to Springer copyrighted material which is used with permission in this thesis, the Springer does not endorse any of University of Windsor's products or services. Internal or personal use of this material is permitted. If interested in reprinting/ republishing Springer copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to https://www.springer.com/gp/rights-permissions/obtaining-permissions/882 to learn how to obtain a License from RightsLink.

VITA AUCTORIS

NAME:	Abdul Muntakim Rafi
PLACE OF BIRTH:	Sylhet, Bangladesh
YEAR OF BIRTH:	1995
EDUCATION:	Blue Bird School & College, Sylhet, Bangladesh MC College, Sylhet, Bangladesh Bangladesh University of Engineering and Technology, Dhaka, Bangladesh