This CVPR Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Application of DenseNet in Camera Model Identification and Post-processing Detection

Abdul Muntakim Rafi[1] Uday Kamal[1] Rakibul Hoque[1] Abid Abrar[1] Sowmitra Das[1]
Robert Laganière[2] Md. Kamrul Hasan[1*]
[1]Bangladesh University of Engineering & Technology [2]University of Ottowa
[2]laganier@uottawa.ca [1*]khasan@eee.buet.ac.bd

Abstract

Camera model identification has earned paramount importance in the field of image forensics with an upsurge of digitally altered images which are constantly being shared through websites, media, and social applications. But, the task of identification becomes quite challenging if metadata are absent from the image and/or if the image has been post-processed. In this paper, we present a DenseNet pipeline to solve the problem of identifying the source camera-model of an image. Our approach is to extract patches of 256×256 *from a labeled image dataset and apply augmentations, i.e., Empirical Mode Decomposition (EMD). We use this extended dataset to train a Neural Network with the DenseNet-201 architecture. We concatenate the output features for 3 different sizes (*$64 \times 64, 128 \times 128, 256 \times 256$*) and pass them to a secondary network to make the final prediction. This strategy proves to be very robust for identifying the source camera model, even when the original image is post-processed. Our model has been trained and tested on the Forensic Camera-Model Identification Dataset provided for the IEEE Signal Processing (SP) Cup 2018. During testing we achieved an overall accuracy of 98.37%, which is the current state-of-the-art on this dataset using a single model. We used transfer learning and tested our model on the Dresden Database for Camera Model Identification, with an overall test accuracy of over 99% for 19 models. In addition, we demonstrate that the proposed pipeline is suitable for other image-forensic classification tasks, such as, detecting the type of post-processing applied to an image with an accuracy of 96.66% – which indicates the generality of our approach.*

1. Introduction

In digital forensics, camera model identification is a distinguished field of research and has profound impact on crucial real-life applications, such as criminal investigations, authenticating evidence, detecting forgery, etc. Nowadays, professional image editing tools are readily available, making image-forgery quite commonplace. Thus, cyber-crimes via digital images are ever increasing and; so as, the need for a robust camera model identification scheme. But unfortunately, the task of identifying the camera-model is very challenging, especially when the metadata of the digital image is unavailable. As a result, a forensic analyst has to implement unique techniques to determine the source camera-model solely from an image.

In the literature, various methods have been proposed to perform this task. [28], [16] and [23] have perfectly described the present condition of camera model identification in their review. The initial approach was an infeasible idea of merging external features in an image for each and every device; like watermarks, device-specific-code, etc. As a result, focus has shifted towards detecting intrinsic camera features, such as the Color Filter Array (CFA) pattern ([3]), interpolation algorithms and Image Quality Metrics (IQM) used in the camera ([15], [10]). Device-specific camera-detection schemes have also been proposed, where noise-patterns like the Photo Response Non-Uniformity (PRNU) have been exploited to identify the device ([8], [18]), [9]. Although device specificity is an inherent feature of PRNU noise, forensic researchers have developed methods to make camera model identification device invariant ([30], [19]). Most of these works try to estimate the model-specific artifacts that are introduced into an image during image-capture, and then, correlate these features with a reference for the corresponding camera-model ([5]). In this approach, the second order statistics of the CFA pattern ([29]) and 3D co-occurrence matrices ([7], [21]) have been used as feature vectors to successfully detect camera-models with state-of-the-art accuracy.

Most of the methods stated so far have used traditional complex ensemble classifiers. Recently, researchers have adopted a data-driven approach and made an effort to solve this problem using Convolutional Neural Networks (CNN). This suggestion seems quite promising because, in the past

decade, Neural Networks have achieved phenomenal accuracy on image-classification benchmarks ([26]). To this end, [31] have trained a CNN on the Dresden database to solve this classification problem. Their work also includes the use of preprocessing using a custom built 2D high-pass filter. However, their overall accuracy is below the state-of-the-art accuracy reported in ([7]). A concept of Content Adaptive Fusion Network is introduced by [32], which is basically a cluster of CNNs with different kernel size, has been introduced to classify camera brand and device to achieve a moderate accuracy around 95%. In spite of the breadth of work performed in this field, little attention has been given to the detection of camera-model specifically from post-processed images (such as different JPEG Compression Rate, Resized, Gamma-Corrected images etc.) Though researchers have explored some of these cases discretely, not many have tried to bring them into the same framework. Image authentication from JPEG headers ([14]), forgery detection from intrinsic statistical fingerprints of images ([27]), detecting doubly compressed JPEG images using Discrete Cosine Transform (DCT) ([17]), and even the recent use of CNN to detect image manipulation in ([1] and [2]) are some examples of work done in detecting image manipulations and classifying them using this approach.

But still, the use of very deep networks is yet to be explored thoroughly for this task. In the absence of metadata and the presence of extensive post-processing in images, we believe that Deep Neural Networks (DNNs) have the potential to achieve a better classification-rate than existing methods. In the presence of these challenges, traditional feature-vectors such as the DCT-Residue([25]) and co-occurrence matrices(([7], [21])) are unarguably altered, often in ways that cannot be predicted in the general case. Thus, designing features that retain the camera-model information even from post-processed images is quite cumbersome, if not, extremely difficult ([19]). This provides the motivation to use Neural Networks to perform this task. Since, DNNs do not require explicit feature engineering, and can automatically learn the necessary features from the image, it makes the task of classification more tractable.

As stated earlier, forensic researchers have used a number of custom neural network architectures ([32], [2], [4], [33]). Besides, a number of deep architectures have been also proposed to perform the task of classification, such as the VGGNet, GoogLeNet, ResNet and most recently, the DenseNet ([20]). A major challenge in using such deep networks is to address the issues of over-fitting and feature-attenuation during training. The camera-model features existing within an image are extremely subtle, compared to other dominating features of the image. As such, while training a Deep Neural Network, these model-level features may be sharply attenuated as the input image is propagated through successive layers.

In this work, we choose to use the 201-layer DenseNet [12] as the core architecture of our network. In the DenseNet, the output of a certain layer is propagated to all the layers in front of it. Any layer in the network has direct access to the features generated by all the layers that came before it. As a result, if any of the image features are lost during forward propagation, they are re-generated at the input of latter layers through the dense connections. That is why, this architecture is quite suitable for detecting minute statistical features like those related to the source camera-model of the image. Again, experiments on image-classification benchmarks have shown that, using a secondary network to re-calibrate the learned features improves the representational power of a network. Motivated by these results, we feed the output features of our main network into a Squeeze-and-Excite block, introduced in ([11]). This boosts the final test-accuracy of our networks on the given benchmarks.

But, the problem of over-fitting still remains, as existing datasets are limited in their size. To overcome this setback and to ensure generalization of the features that are learned, we use a number of data-augmentation schemes such as, gamma correction, JPEG compression, re-scaling, extracting patches for training, randomly cropping and flipping the training image to extend the dataset. These prevent the network from becoming dependent on the specific device from which the training images are taken and help the network learn more robust features. Additionally, we have also trained the network for image manipulation detection, to see whether or not- it can be used as a general purpose network for image forensics.
The following sections of this paper explain image acquisition model, the outline of our model, detailed architecture of the network, training procedure, and the detailed comparative results.

2. Materials

2.1. Description of Datasets

In order to train our network, we have used the Camera-Model Identification Dataset provided for the IEEE Signal Processing (SP) Cup 2018. The initial dataset consisted of images captured by 10 different camera models having 275 images for each, all of which were provided by the IEEE Signal Processing Society. In addition to this, external data for each camera-model is collected from Flickr during the open competition phase of SP Cup 2018. This one contained varying number of images for each camera model. Dataset-I is formed by combining both of these sets of data. A brief summary of the dataset-I is given in Table 1.

The test data for the SP Cup dataset is provided separately on the Kaggle platform without any labels. It includes

Camera Model	SP Cup Data (No. of Images)	Flickr Data (No. of Images)
HTC-1-M7	275 × 10	746
iPhone-4s		499
iPhone-6		548
LG-Nexus-5x		405
Motorola-Droid-Maxx		549
Motorola-Nexus-6		650
Motorola-X		344
Samsung-Galaxy-Note3		274
Samsung-Galaxy-S4		1137
Sony-NEX-7		557
Sub-Total	2750	5709
Grand-Total	8459	

Table 1. SP Cup data & Flickr data (Dataset I)

2640 images of size 512×512, among which 1320 are unaltered and the rest are manipulated externally. The details of the manipulation scheme used to generate these images are discussed in subsequent sections.

In addition to Dataset-I, we have also performed experiments on the Dresden Image Database. This dataset include varying number of images for 27 different camera models. We denote these images as Dataset-II.

2.2. Data Augmentation

Additional data has been generated by post-processing the original images given in the dataset. It is a common practice in deep learning to deliberately alter the input data to help the network learn more robust features. A total of 8 types of post-processing have been performed on the images of Dataset-I. These are JPEG-Compression with quality factor 90% and 70%; Resizing by a factor of 0.5, 0.8, 1.5 and 2.0; Gamma-Correction using γ as 0.8 and 1.2. Also, EMD has been performed as an augmentation which is discussed in section 2.2.2. Moreover, the input image is randomly rotated by $0°, \pm 90°$, and $180°$ during training. Because of this, the network can extract the camera model-features irrespective of whether the image was taken in landscape mode or portrait mode.

3. Methods

3.1. Model Proposal

The complete structure of our model is shown in Fig. 1. Different parts of our model are outlined as follows:

- First, we select patches of size 256×256– from the generated images based on their quality.
- After extracting patches, we use them to train Dense Convolutional Networks (DenseNets), specifically the DenseNet-201 architecture, with patches of size 256×256.
- Next, using the DenseNet-201 trained on 256×256 patches only, we extract features from second to the last layer for the size 256×256 and all non-overlapping patches of size 128×128 and 64×64 from each training image. Thus, at the end, we essentially have 3 feature vectors for 3 different patch size.
- Then, we concatenate the feature vectors produced by this network and use them to train a secondary network consisting of a Squeeze-and-Excitation (SE) block and a classification block. The output of the SE block is passed to the classification block.
- During testing, we similarly generate feature vectors for each 256×256 patch using the DenseNet-201 trained on 256×256 patches only. These features are concatenated and passed to the secondary network to generate the final prediction for the entire image.

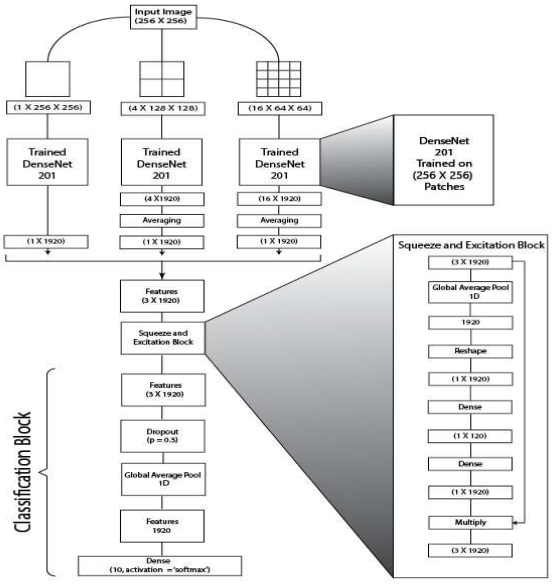

Figure 1. Overview of the network.

3.2. Training Data Generation

3.2.1 Selecting and Extracting Patches

The first step of the proposed pipeline is to generate patches from the input images—both processed and unprocessed. The idea of extracting patches is motivated by 3 reasons: (i) it results in more data to train our neural network, thus making the training process more generalized; (ii) it enables us to generate multiple predictions for a given test image. Averaging over all of those predictions will ensure a more

accurate classification, and (iii) training our network with patches of small size relative to the image prevents our network from learning dominant spacial features of the image. As a result, the network can better learn inherent statistical features related to the source camera model. The patch sizes that we opt to use is 256×256.

But, it is apparent that, all the patches are not suitable for training. In particular, saturated patches are not likely to contain enough statistical information about the used camera model. Therefore, before extracting patches, we determine their quality and only use patches of good quality to train our network.

We compute the quality value of a patch as outlined in [4]. For each patch $\mathcal{P}$ in an image, its quality $Q(\mathcal{P})$ is computed as:

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

where α, β and γ are empirically set constants (set to 0.7, 4 and $\ln(0.01)$ in our experiments, respectively), whereas μ_c and σ_c, $c \in [R, G, B]$ are the mean and standard deviation of the red, green and blue components (normalized by 255 to the range [0,1]) of patch $\mathcal{P}$, respectively. This quality measure tends to be lower for overly saturated or flat patches, whereas it is higher for textured patches showing some statistical variance. For each image, we select 20 patches of size 256×256 with the highest Q values.

3.2.2 Empirical Mode Decomposition

The training data has been augmented further by performing EMD [13]. In EMD, an input signal is decomposed into the so-called Intrinsic Mode Functions (IMFs) and a Residue (see in Fig. 2). Mathematically, for 2D EMD the deocomposition can be expressed as

$$I(m,n) = \sum_{j=1}^{L} \text{IMF}_j(m,n) + \text{Res}_L(m,n) \tag{2}$$

where $I(m,n)$ is the 2D image, $\text{IMF}_j(m,n)$ is the $j-$th Intrinsic Mode Function, and $\text{Res}_L(m,n)$ represents the Residue corresponding to L intrinsic modes.
In our experiments, we have $m = n = 256$. The most commonly used algorithm for 2D-EMD is implemented using FastRBF [6]. At first, a set of discrete nodes denoted by $\mathbf{X} = \{\mathbf{x_i}\}_{i=1}^{N} \in I(m,n)$ are selected, which are either local minima or local maxima points for $I(m,n)$. Here, $\mathbf{x_i}$ can be described as (x_i, y_i) points on a 2D plane. These coordinates are used as centers for RBF or Radial Basis Functions. An RBF or Radial Basis Function [22] is mathematically expressed as

$$s(\mathbf{x}) = \mathrm{p}_m(\mathbf{x}) + \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{x} - \mathbf{x_i}\|) \tag{3}$$

where, $s(\mathbf{x})$ is the Radial Basis Function or RBF, $\mathrm{p}_m(\mathbf{x})$ is a low-degree polynomial with degree m, λ_i are the RBF coefficients, ϕ is a real valued function (the spline function is used in our case) and $\mathbf{x}$ denotes variable point (x, y) on 2D space and $\mathbf{x_i}$ are the RBF centers. Here, $\|\cdot\|$ denotes the Euclidean norm.

The algorithm [24] uses FastRBF to interpolate upper and lower envelopes of scattered local maxima and minima from $I(m,n)$. The mean of the envelopes is then subtracted from the image to get the IMF.

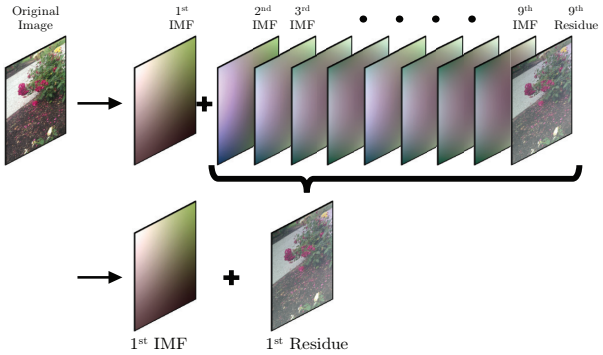

Figure 2. EMD of an Image– showing the Intrinsic Mode Functions (IMFs) and Residue.

In this work, we have used 2-dimensional EMD to remove the 1st IMF from each channel of the input images separately and retain the residue obtained after the 1st stage decomposition. This residue serves as additional data to train our networks. Applied in this manner, EMD essentially works as a denoising scheme by removing random high-frequency noise-components from the image data. Thus, using EMD more than once may prove to be detrimental, as the intrinsic camera-model features embedded in the image may be removed upon successive decompositions.

We have used Python's PyEMD library to apply the decomposition to all of the 256×256 patches extracted from the SP-Cup Data.

3.3. Architecture

3.3.1 Densenet

After extracting patches from the images, we train a deep Convolutional Neural Network (CNN) to perform the task of Source Camera-Model Identification. The CNN model that we opt to use is the Dense Convolutional Network

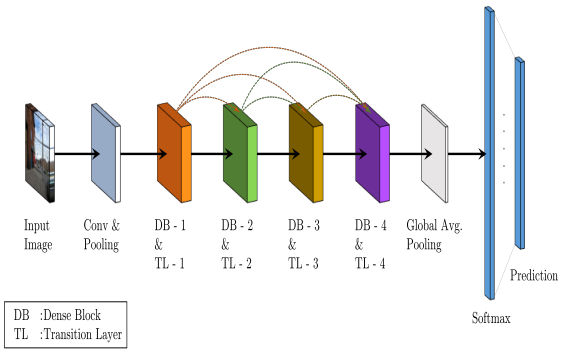

Figure 3. Illustration of Dense-Connections and Transition-Layers implemented in DenseNet.

(DenseNet). The details of the DenseNet architecture that we use for Camera-Model Identification is summarized in Table 2.

Table 2. Architecture of DenseNet-201

Layers	DenseNet-201
Convolution	7×7 conv, stride 2
Pooling	3×3 max pool, stride 2
Dense Block (1)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	1×1 conv
	3×3 max pool, stride 2
Dense Block (2)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	1×1 conv
	2×2 average pool, stride 2
Dense Block (3)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Transition Layer (3)	1×1 conv
	2×2 average pool, stride 2
Dense Block (4)	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Layer	Global Average Pooling
Classification	Softmax

The model that we use is the 201-layer DenseNet introduced in [12]. It consists of 4 dense blocks, each with a growth-rate of 32. Transition layers have been used between successive dense blocks. These consist of a convolution layer and a max-pooling layer. No reduction and dropout layers have been used in the network. The dimensionality of the output feature vector is reduced by using Global Average Pooling, and the features are finally classified by using a Fully-Connected layer with Softmax as the activation function. This layer outputs the probabilities of classification for each class.

The intuition behind using this architecture is the nature of the classification that we wish to accomplish. The camera-model features inherent in an image are very subtle and minute features of the image [28]. Detecting and classifying these features are difficult in itself. But, the task is made even more challenging by the constraints posed for the task. In addition to the model-level features, the image also contains device-level features such as the Photo Response Non Uniformity (PRNU) sensor noise [9] . To detect the source camera-model effectively, we need to take care that the network does not become dependent on this type of sensor noise. In addition to this, post-processing has also been introduced in the dataset which alters the spacial structure of the model-features in an unpredictable manner. Therefore, a network that can detect the model-features under all of these constraints needs to be sufficiently deep and have a large number of parameters. But, training such a deep network to detect the subtle model-features proves to be very difficult. The network invariably becomes dependent on the image content or the device specific noise, as all of the minute statistical information is lost when the image is propagated through consecutive layers.

This problem is alleviated in the DenseNet through the use of dense connections. To preserve image information throughout the network, the output of each layer is propagated to all of the layers in front of it. Even if some of the minute features are lost due to some operation, it is regenerated from the output of the previous layers at the input of the subsequent layers through these dense connections (see Fig. 3). This prevents the gradient-flow from vanishing during training in such a deep network and allows us to extract features which are very difficult - if not impossible to detect using conventional CNN architectures.

3.3.2 Squeeze and Excitation Block

The output after the 4 dense blocks is passed to another module called a "*Squeeze-and-Excitation*" (SE) block. This module has been introduced by Hu, Shen and Sun [11]. The aim of this module is to improve the representational power of a network by explicitly modelling the interdependencies between the channels of its output. To achieve this, the SE block performs feature recalibration, through which it can learn to use global information to selectively emphasize informative features and suppress less useful ones, without changing the dimensions of the feature vector.

The internal layers of the SE block and corresponding shapes are given in Figure 4. We construct the SE block to perform feature recalibration as follows. The input features are first passed through a *squeeze* operation, which aggregates the feature maps across spatial dimensions to produce a channel descriptor. This descriptor embeds the global distribution of channel-wise feature responses, enabling information from the global receptive field of the network to be

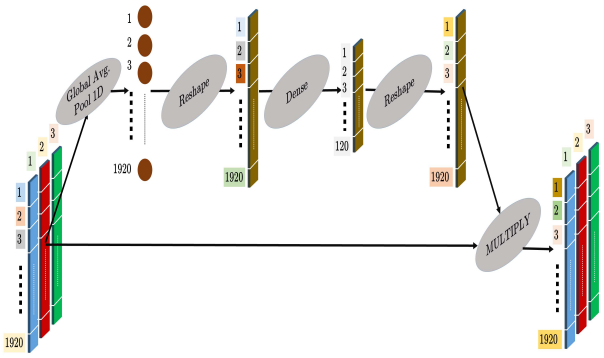

Figure 4. Illustration of a Squeeze-and-Excitation Block.

leveraged by its lower layers. This is followed by an *excitation* operation, in which sample-specific activations, learned for each channel by a self-gating mechanism based on channel dependence, govern the excitation of each channel. The feature maps are then reweighted to generate the output of the SE block which can then be fed directly to the classification layers.

3.3.3 Classification block

The modified features, of size (3×1920), produced at the output of the SE block is then passed through a Dropout layer with a dropout-rate of 30%. This is followed by a Global Average Pooling operation to reduce the feature vector to a size of (1×1920). Finally, we pass the pooled feature vector to a Dense Layer with Softmax as the activation function to generate probabilities for the 10 classes which represent the 10 camera models that we need to classify.

4. Experiments

In this section, we discuss the training procedure in detail. Before training, the DenseNet-201 model was initialized using weights pre-trained on the ImageNet database. This ensured a better and faster convergence of the weights during training.

4.1. Phase-I

In Phase-I, we train our model using Dataset-I. We take 20 patches of size 256×256 from each image to train our network. During training, 85% of the total number of patches are used for training and the rest are used for validation. We have used Stochastic Gradient Descent as the Optimizer in our network with a momentum of 0.9 and initial learning rate of 10^{-3}. The learning rate is decreased by a factor of 10^{-1} if the validation loss have not decreased in 2 successive epochs. In this way, when the learning rate is reduced to 10^{-7}, training is stopped.

After training the DenseNet, we extract features from the second to last layer for input size of 256×256. The output features are of size 1×1920. We also extract features for all the non-overlapping patches of size 128×128 and 64×64 that the 256×256 patch contains as visualized in Figure 1. We receive 4×1920 feature vector from the 4 non-overlapping patches of 128×128 and 16×1920 feature vector from the 16 non-overlapping patches of size 64×64. We reduce both the feature vectors of size 4×1920 and 16×1920 to 1×1920 each by averaging. Lastly, we concatenate the feature vectors for 3 different input sizes to form a resultant feature vector of size 3×1920. It should be noted that the same model is used for extracting features.

We then use the output feature vectors generated by the DenseNet to train our secondary network. These features are passed to the Squeeze-and-Excite and classification network. The classification network outputs a final class vector of size 1×10 which represent the 10 Camera Models that we need to detect (see Figure 1).

Although our proposed architecture is a single model trained on 256×256 patches only, for investigation purposes, we have separately trained three separate DenseNet-201 networks for three input sizes, the outcomes of which shall be discussed in the next section.

4.2. Phase-II

For phase-II, we have used the images of Dataset-II and extracted the best 20 non-overlapping patches of size 256×256 depending on the quality we outlined before. For training, we have used transfer learning on our previously trained model from Phase-I. We load the weights of the network from Phase-I to initialize the DenseNet and train the network for input size of 256×256. We do not implement our full proposed pipeline in this phase. The output feature of the DenseNet of size 1×1920 for 256×256 input patches have been directly used in classification. The classification block receives the 1×1920 feature vector from DenseNet and is trained for the 27 classes. Both the classification block and the DenseNet runs through the same backpropagation. Hyper-parameters for training have been kept the same as in Phase-I.

4.3. Phase-III

In Phase-III, We have used all images from Phase-I. However, EMD-data has not been included in this case. We sub-divided these data into 4 classes (Unaltered, Resized, JPEG-Compressed and Gamma-Corrected) irrespective of their camera models. Similar to phase-II, DenseNet has been initialized using the network from phase-I and the classification block is trained to detect the presence of manipulation in the data. It must be mentioned that, dur-

Table 3. Detection Accuracy of Camera-Models for different Input Sizes

Network	Accuracy		
	Unaltered (70%)	Manipulated (30%)	Total (100%)
DenseNet-201 (64×64)	67.16%	27.43%	94.59%
DenseNet-201 (128×128)	68.33%	28.61%	96.94%
DenseNet-201 (256×256)	68.75%	28.82%	97.57%
DenseNet-201 (Final Layer Prediction Average)	69.12%	28.84%	97.96%
Full Pipeline	69.33%	29.04%	98.37%

ing training, our dataset have been reduced to some extent ($150000 \times 4 = 600000$) to make the training data evenly distributed among 4 classes. Also, in this case, we have used 128×128 sized patches for training due to much higher prediction accuracy compared to other sizes.

The accuracies obtained from all of these networks during training and testing are included in the result section.

5. Results and Discussion

In this section, we shall discuss our experimental procedure in details. The following subsections will present the outcomes of our experiments.

5.1. Phase-I

This is the core result of our work. The test dataset of Phase-I is completely from an unseen device and contains 2640 images of size 512×512 with equal numbers of unaltered and manipulated images. We have tested the results generated by our networks in Kaggle. According to the competition rules of IEEE Signal Processing Cup 2018, Kaggle provides a score on the test-results based on the following formula:

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

In this work, whenever we mention overall accuracy, we refer to this score. We can calculate individual accuracies from the above scoring equation by submitting predictions for unaltered or manipulated images separately. The test-accuracies of Phase-I are summarized in Table-3.

In Table 3, we can clearly see the impact of input image size on the test-results. Despite being trained on the same DenseNet-201 architecture, higher accuracy is produced for larger input sizes. It may be the consequence of lower quality of the 64×64 patches. The residual camera-model information left after cropping an image to this size are minimal. This may have caused difficulties for the network to predict accurately for inputs of this size.

Nonetheless, a better result may be obtained by averaging the predictions of the 3 networks, with separate weights for the classification layer. This illustrates that some of the camera-model features may vary depending on the size of the input. As a result, ensembles over multiple networks trained on different input sizes are likely to have improved performance. However, our aim in this work is to maximize the detection-accuracy using a single network. So, we used the weights for the 256×256 input-size in all the networks of our model. We have generated the output features for all the input sizes using this single weight. And we have used the SE network to automatically adjust the weights of these features. This full pipeline achieved an overall accuracy of 98.37%.

Besides, the performance of our network on different parts of the dataset are shown in Figure 5.

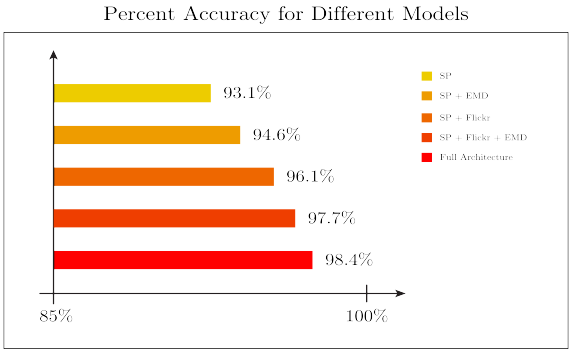

Figure 5. Difference in accuracy with the variation of Dataset. Here we can clearly see the effects of adding EMD augmented images in increasing the accuracy.

Table 4. Predictions of detected manipulations

	Unaltered	JPEG-Compr.	Gamma-Corrected	Resized
Unaltered	**90.07%**	3.49%	6.06%	0.38%
JPEG-Compr.	0.15%	**99.85%**	0%	0%
Gamma-Corrected	3.18%	0%	**96.75%**	0.07%
Resized	0%	0%	0%	**100%**

Using patches of size 256×256 only, extracted from the SP Cup Data and their manipulated versions, we achieved a modest accuracy of 93.1%. However, since these images are collected from only one device for each camera-model, the low result is expected. The network inevitably learns

device-level features, which degrades its performance. After adding EMD versions of these images, the accuracy significantly improves to 94.6%. Since, the 1st IMF of the input image was removed in our EMD versions, this IMF is likely to have some correlations to the device-level features. Using the entire Dataset-I during training, which included the data from Flickr, the accuracy is further improved to 96.1%. This can be mainly attributed to the presence of images captured by different devices in the Flickr Data for each camera-model. Because of the presence of these variations in device-level features, the network could learn the model-specific features more accurately– thus causing the increase in accuracy. Adding EMD versions of all these images boosted the accuracy to 97.7%. This strengthens our previous assumption of EMD being an effective augmentation technique in this task. To recall, all of these results are achieved by using patches of only one size– 256×256. Using all of the patches and our full pipeline, we have achieved a final accuracy of 98.37%. This is our final result on Dataset-I, which is the current state-of-the-art on this dataset using a single network.

5.2. Phase-II

For Phase-II, we have tested our network on the images from 27 different camera models of the Dresden Database. Although we have not used our full pipeline for this dataset, an overall accuracy of over 99% is achieved for 19 camera models by the 1st network of Phase-I. The camera-models for which accuracy dropped are CANON_IXUS-55, CANON_A640, NIKON_D200, NIKON_D70, NIKON_D70S, SONY_H50, SONY_T77, SONY_W170. However, the false detection of images is confined within the models of the manufacturing company. It means, this network is able to detect the manufacturing company of the source camera-model with an accuracy of 100%. Also, we have another very important thing to notice in this case. Though the training dataset is very small ($16961 \times 20 = 339220$) compared to the dataset of Phase-I, but still, DenseNet-201 is able to detect the camera models very accurately because of the learnt features from Phase-I. This indicates that our network can be fine-tuned to detect further camera models. In case of wrongly detected camera models, such as Nikon_D200 or Sony_W170, there is a high chance that these models does have almost similar interpolation method or CFA pattern corresponding to other camera models from the same manufacturer. That is why the test-results show some mismatch for these models. Our findings are in commensurate with the work of Kirchner et al. [16] where Nikon D70 and Nikon D70s, have been found out to be the same.

5.3. Phase-III

In this phase of experiments, we have tried to identify the 4 types of image-manipulations used on the images of Dataset-I: Unaltered, JPEG Compressed, Gamma Corrected and Resized. In testing, we have used the unaltered images from the test data of size 512×512 provided by Kaggle and generated a total of $1320 \times 9 = 11880$ test images, which include 1320 unaltered, 2640 JPEG compressed, 2640 gamma corrected and 5280 resized images. Details of the result are given in Table 4. We have achieved an overall accuracy of 96.66% in this task. It is instructive to mention that, these results have been obtained by using only the DenseNet-201 architecture and 128×128 patch size.

The results show that, the features learned by our proposed model, have some sort of orthogonality among them, depending on the type of manipulation present in the image. As a result, these features may be used in other image-forensic tasks outside the premise of our current work.

6. Conclusion

In this paper, we have proposed a DenseNet-oriented pipeline for identifying the source camera-model of an image. We have used DenseNet-201 as well as a Squeeze and Excitation (SE) network for our model architecture and trained our model on the IEEE Forensic Camera-Model Identification Dataset. This pipeline shows an overall accuracy of 98.37% on the test data provided for the IEEE Signal Processing Cup 2018 Camera Model Identification Challenge. A number of Data-Augmentation techniques have been used in our work to extend the dataset, among which EMD is a novel addition to the repertoire of techniques used in Camera-Model Identification. Besides, we have also used transfer learning and evaluated our model on the Dresden Image Database, which showed an accuracy of over 99% for 19 camera models, where we have been able to detect the manufacturing company of the camera-model with an accuracy of 100%. However, there is an issue that needs to be addressed regarding the experiment on Dresden image Database. The test images used in this experiment are not from a separate device than the devices used to capture the training images because of the unavailability of multiple devices for all 27 camera models. This may have resulted in higher accuracy than the experiment on SP Cup database. Moreover, the features learned by the DenseNet have also been used to classify the manipulations that have been applied to an image, with an accuracy of 96.66%. This demonstrates the generalization of our training procedure, for detecting camera-model features across varying datasets and the suitability of using these features in multiple image-forensic tasks.

References

[1] Belhassen Bayar and Matthew C Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 5–10. ACM, 2016. 2

[2] Belhassen Bayar and Matthew C Stamm. Design principles of convolutional neural networks for multimedia forensics. *Electronic Imaging*, 2017(7):77–86, 2017. 2

[3] Sevinc Bayram, Husrev Sencar, Nasir Memon, and Ismail Avcibas. Source camera identification based on cfa interpolation. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–69. IEEE, 2005. 1

[4] Luca Bondi, Luca Baroffio, David Güera, Paolo Bestagini, Edward J Delp, and Stefano Tubaro. First steps toward camera model identification with convolutional neural networks. *IEEE Signal Processing Letters*, 24(3):259–263, 2017. 2, 4

[5] Hong Cao and Alex C Kot. Accurate detection of demosaicing regularity for digital image forensics. *IEEE Transactions on Information Forensics and Security*, 4(4):899–910, 2009. 1

[6] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001. 4

[7] Chen Chen and Matthew C Stamm. Camera model identification framework using an ensemble of demosaicing features. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, pages 1–6. IEEE, 2015. 1, 2

[8] A Emir Dirik, Husrev T Sencar, and Nasir Memon. Source camera identification based on sensor dust characteristics. In *Signal Processing Applications for Public Security and Forensics, 2007. SAFE'07. IEEE Workshop on*, pages 1–6. IEEE, 2007. 1

[9] Tomás Filler, Jessica Fridrich, and Miroslav Goljan. Using sensor pattern noise for camera model identification. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 1296–1299. IEEE, 2008. 1, 5

[10] Thomas Gloe. Feature-based forensic camera model identification. In *Transactions on Data Hiding and Multimedia Security VIII*, pages 42–62. Springer, 2012. 1

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. 2, 5

[12] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017. 2, 5

[13] Norden E Huang, Zheng Shen, Steven R Long, Manli C Wu, Hsing H Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. In *Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences*, volume 454, pages 903–995. The Royal Society, 1998. 4

[14] Eric Kee, Micah K Johnson, and Hany Farid. Digital image authentication from jpeg headers. *IEEE transactions on information forensics and security*, 6(3):1066–1075, 2011. 2

[15] Mehdi Kharrazi, Husrev T Sencar, and Nasir Memon. Blind source camera identification. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 1, pages 709–712. IEEE, 2004. 1

[16] Matthias Kirchner and Thomas Gloe. Forensic camera model identification. *Handbook of Digital Forensics of Multimedia Data and Devices*, pages 329–374, 2015. 1, 8

[17] Bin Li, Yun Q Shi, and Jiwu Huang. Detecting doubly compressed jpeg images by using mode based first digit features. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pages 730–735. IEEE, 2008. 2

[18] Jan Lukáš, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, 2006. 1

[19] Jan Lukas, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, 2006. 1, 2

[20] Francesco Marra, Diego Gragnaniello, and Luisa Verdoliva. On the vulnerability of deep learning to adversarial attacks for camera model identification. *Signal Processing: Image Communication*, 65:240–248, 2018. 2

[21] Francesco Marra, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. A study of co-occurrence based local features for camera model identification. *Multimedia Tools and Applications*, 76(4):4765–4781, 2017. 1, 2

[22] Jean Claude Nunes, Steve Guyot, and Eric Deléchelle. Texture analysis based on local analysis of the bidimensional empirical mode decomposition. *Machine Vision and applications*, 16(3):177–188, 2005. 4

[23] Alessandro Piva. An overview on image forensics. *ISRN Signal Processing*, 2013, 2013. 1

[24] Li-Hong Qiao, Li-Zhong Peng, Wei Guo, and Wei-Tao Yuan. A novel image fusion algorithm based on 2d emd and ihs. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 7, pages 4040–4044. IEEE, 2008. 4

[25] Aniket Roy, Rajat Subhra Chakraborty, Udaya Sameer, and Ruchira Naskar. Camera source identification using discrete cosine transform residue features and ensemble classifier. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1848–1854. IEEE, 2017. 2

[26] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 2

[27] Matthew C. Stamm and K.J. Ray Liu. Forensic detection of image manipulation using statistical intrinsic fingerprints. *IEEE Transactions on Information Forensics and Security*, 5(3):492–506, 2010. 2

[28] Matthew C Stamm, Min Wu, and KJ Ray Liu. Information forensics: An overview of the first decade. *IEEE Access*, 1:167–200, 2013. 1, 5

[29] Ashwin Swaminathan, Min Wu, and KJ Ray Liu. Nonintrusive component forensics of visual sensors using output images. *IEEE Transactions on Information Forensics and Security*, 2(1):91–106, 2007. 1

[30] Thanh Hai Thai, Remi Cogranne, and Florent Retraint. Camera model identification based on the heteroscedastic noise model. *IEEE Transactions on Image Processing*, 23(1):250–263, 2014. 1

[31] Amel Tuama, Frédéric Comby, and Marc Chaumont. Camera model identification with the use of deep convolutional neural networks. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016. 2

[32] Pengpeng Yang, Wei Zhao, Rongrong Ni, and Yao Zhao. Source camera identification based on content-adaptive fusion network. *arXiv preprint arXiv:1703.04856*, 2017. 2

[33] Hongwei Yao, Tong Qiao, Ming Xu, and Ning Zheng. Robust multi-classifier for camera model identification based on convolution neural network. *IEEE Access*, 6:24973–24982, 2018. 2